PX-8
OS Reference Manual

Trademark Acknowledgments

CP/M™ is a trademark of Digital Research, Inc.

MICROCASSETTE™ is a trademark of OLYMPUS OPTICAL CO., LTD.

NOTICE

CONTENTS

\

Chapter 1　Introduction

1.1　Purpose of This Manual

This manual describes the functions of the operating

system for the EPSON PX-8, HC-80, and HC-88 series

(referred to as MAPLE) microcomputer systems. It is

intended for system house users who are to develop

applications programs which make the best of the MAPLE's

capabilities.

The reader is assumed to be familiar with the following:

-　　　Basic knowledge about the CP/M operating system

-　　　General knowledge about machine-language programming

-　　　Z80 instructions

1.2 Before Reading This Manual

This manual uses the following notational conventions:

(1) Data representation

This manual discusses binary, decimal, and hexadecimal numbers. They are represented in the formats:

Binary: 00100011B (Numbers are followed by 'B')

Decimal: 35 (only numerals)

Hexadecimal: 23H (Numbers are followed by 'H')

Character constants are enclosed in apostrophes (').

Example:

    'ABC'

(2) Operating system types

The MAPLE runs in three types of operating systems (OS).

In this manual, these operating systems are

distinguished as follows:

  ASCII (OS): ASCII ver. OS (PX-8)

  JIS (OS): Japanese-language JIS Keyboard OS (HC-80, -88)

  TXT*(OS): Japanese-language TXT Keyboard OS (HC-80, -88T)

    * TXT stands for the Touch-16 Japanese language

     input methods originally developped by EPSON.

  Japanese-language (OS): Japanese-language JIS and TXT

     Keyboard OSs

Chapter 2   General Description and System Configuration

2.1 MAPLE System Configuration

The MAPLE is a successor of the worldly-accepted EPSON HC-20 hand-held computer. It is a new generation hand-held computer which incorporates in its compact body much more functions than ordinary desktop microcomputers. With its battery-driven power supply, the user can user the MAPLE any time, any place, even outside the office.

To further augment this outstanding portability feature, EPSON supplies a wide variety of peripheral devices and options. For example, the MAPLE employs a large (80 columns by 8 lines) LCD screen. With the virtual screen support, the MAPLE allows the user to create display images larger than those the conventional CRT devices can provide. The MAPLE is furnished as standard microcassette drives which are completely controlled by the distribution operating system so the user can handle them as easy as floppy disk units. Another standard device is an RS-232C interface which enables the MAPLE to communicate with other computers directly or via a telephone lines. When combined with an optional microfloppy disk drives, P-80 printer, or CP-20

acoustic coupler, all are battery driven!, the MAPLE
provides a full computing environment even in locations
where no commercial AC source. The main unit proper
will meet most of daily business needs.

The MAPLE employs as its operating system the industry
standard CP/M version 2.2 operating system implemented
in ROM. This allows the user to implement an abundance
of commercial CP/M application programs on the MAPLE.
In addition to the supports for all MAPLE peripheral
devices, the MAPLE CP/M has many extended functions
which will help the user develop application programs
for the MAPLE.

The MAPLE with the Japanese-language OS and Japanese-
language processor unit supports kanji processing so that
the user can easily construct application programs using
kanji characters.

The ideal combination of the MAPLE with the software
that make the best of the MAPLE's portability and
capability will explore new computer uses that no one
ever imagine.

2.2 Hardware Configuration

2.2.1 Hardware Configuration (see block diagram)

(1) CPU

The MAPLE uses three processors: Z80, 6301, and 7508. The 6301 and 7508 processors are used mainly to control I/O operations to reduce the burden of the Z80 central processing unit.

1) Z80

- Main CPU

- CMOS version

- 2.46 MHz clock

2) 6301

-      8-bit CPU

-      CMOS version

-      614 KHz clock

-      Contains 4K-byte program

The 6301 CPU controls the following I/O devices:

-      Screen (LCD)

-      Serial Interface

-      Microcassette

-      ROM capsule

-      Speaker

3) 7508

-      4-bit CPU

-      CMOS version

-      200 KHz clock

-      Contains 4k-byte program

The 7508 CPU controls the following I/O devices

-      Keyboard

-      Power supply to main CPU

-      RESET SW

-      Battery voltage port

- Temperature data port

- Calendar ports

- Alarm port

- 1-second software timers

(2) Memory

OS ROM: 32K bytes (CMOS mask ROM)

Main ROM: 64K bytes (CMOS DRAM)

VRAM: 6K bytes (CMOS DRAM)

- The OS ROM and main RAM are bank-switched.

- VRAM is controlled by the 6301 processor.

- The main RAM and VRAM are battery backed up and their data are sustained even when power switch is turned off.

(3) Battery

Two types of rechargeable Ni-Cd batteries are used:

Main battery capacity: 1100 mAH

Subbattery capacity: 90 mAH

Normally, the main battery is held on. When the power voltage falls down to 4.7 volts, power is switched from the main battery to subbattery and the subbattery maintain only power to the RAM. Recharging (tricle

recharging) is accomplished using the attached AC adapter. Eight hours after tricle recharging is started with the AC adapter, recharging is stopped to prevent overcharging from damaging the battery. The main battery charges the subbattery while it is in operation.

(4) Interrupt handling

Z80 mode 2 interrupts are used for interrupt to Z80. Six interrupt levels are available. They are listed below in the descending order of priority:

1)   Interrupts from the 7508

2)   RS-232C receive interrupt from 8251

3)   CD (Carrier Detect) interrupt from RS-232C interface

4)   FRC (Free Running Converter) overflow interrupt

5)   ICF (Interrupt Catch Flag) interrupt from the bar code reader.

6)   External interrupt

2.2.2 Built-in I/O devices

(1) Keyboard

- The keyboard consists of 66 keys and six switches (66 keys and seven switches for Japanese-language version).
- The keyboard supports N-key rollover feature.
- The keyboard also supports auto repeat feature.

(2) LCD

- 480 dots (wide) x 64 dots (long)

  Dot size: 0.41 mm (wide) x 0.45 mm (long)

  Dot spacing: 0.46 mm (wide) x 0.50 mm (long)
- 80 characters by 8 lines (30 characters by 3 lines for kanji characters)
- The LCD panel swivels in the range of 180° in 13 intervals.
- The LCD view angle can be controlled by a slide switch.
- 1/64 duty

(3) Microcassette drive

- The microcassette drive is controlled by software.

- Allows Dame recording and playback. Only playback
  is possible with voice information.

- The sound from the microcassette drive can be
  monitored using the internal or external
  speaker.

- The tape speed is 2.4 cm/second.

(4) ROM capsule

- 28-pin 2764/27128, 27256, or equivalent.

- NMOS or CMOS mask ROM or PROM is possible.

- A ROM capsule can contain up to two ROM chips.
  They may be used single or in combination.

- Power to the ROM is supplied only when it is
  accessed, which is controlled by software.

- The ROM capsule allows easy installation or removal
  by the user.

(5) Built-in dynamic speaker

-     Compact dynamic speaker

-     The frequency and duration can be controlled by software.

-     The volume can be adjusted with a volume control.

-     The output can be connected to an external speaker interface.

2.2.3 External Interfaces

(1) RS-232C

- The RS-232C interface uses a CMOS 8251 controller chip (compatible with Intel 8251A).

- The output level is $\pm$ 8 volts.

- The power to the driver is controlled by software.

- 8-pin mini-DIN connector is used.

- Bit rates (bps)

  TX: 110, 150, 200, 300, 600, 1200

  RX: 110, 150, 200, 300, 600, 1200

  TX: 240, 4800, 8600, 19200

  RX: 240, 4800, 8600, 19200

  TX: 1200, 75

  RX: 75, 1200

- Number of start bits: 1

- Number of stop bits: 1, 2

- Data length: 7, 8

- Parity: Even, odd, none

- Full duplex/half duplex

(2) Serial interface

- The output level is $\pm$ 8 volts.

- The power to the driver is controlled by software.

- The driver is shared with the RS-232C interface.

- 8-pin mini-DIN connector is used.

- Bit rates (bps)

    TX: 110, 600, 4800, 38400

    RX: 110, 600, 4800, 38400

- Number of start bits: 1

- Number of stop bits: 1

- Data length: 8

- Parity: None

- Full duplex/half duplex

(3) Bar code reader

- 3-pole connector

- Power is controlled by software.

(4) Analog input ports

- 2 channels

- Input level: 0 to 2 volts

- Resolution: 6 bits ($2 v / 2^6 \fallingdotseq 0.03$ v)

(5) External speaker

- The output to the built-in speaker can be switched
  to the external speaker by plugging in a plug into
  the speaker jack.

(6) System bus

- A total of 50 lines including the 16 address bus
  lines and 8 data bus lines are available.

Peripheral Devices Connectable to the External Interface

External Interface	Peripheral	Cable	Options
RS-232C	Printer	#723	P40 P80 Series
		#725	EPSON printers with serial I/F MP(X) series FP(X) series RP(X) series
	Acoustic coupler	#724	CP-20 CX-20 CX-21
	Computer	#726	MAPLE PINE
		#725	QC(X)-20 QC(X)-10
		#738	HC(X)-20

External Interface	Peripheral	Cable	Options
Serial I/F	Minifloppy disk drive	#723	TF-10 TF-15 TF-20
	Microfloppy disk drive	#726	PF-10
	Printer	#723	P40 P80 series
		#725	EPSON printers with serial I/F MP(X) series FP(X) series RP(X) series
Bar code reader I/F	Bar code reader (Wand)		HØØBR code JA (low resolution) HØØBR code HA (high resolution)

External Interface	Peripheral	Cable	Options
System bus	Expansion unit	#727	RAM disk unit Japanese-language unit Modem unit Multi-unit 64 Universal unit

Block Diagram
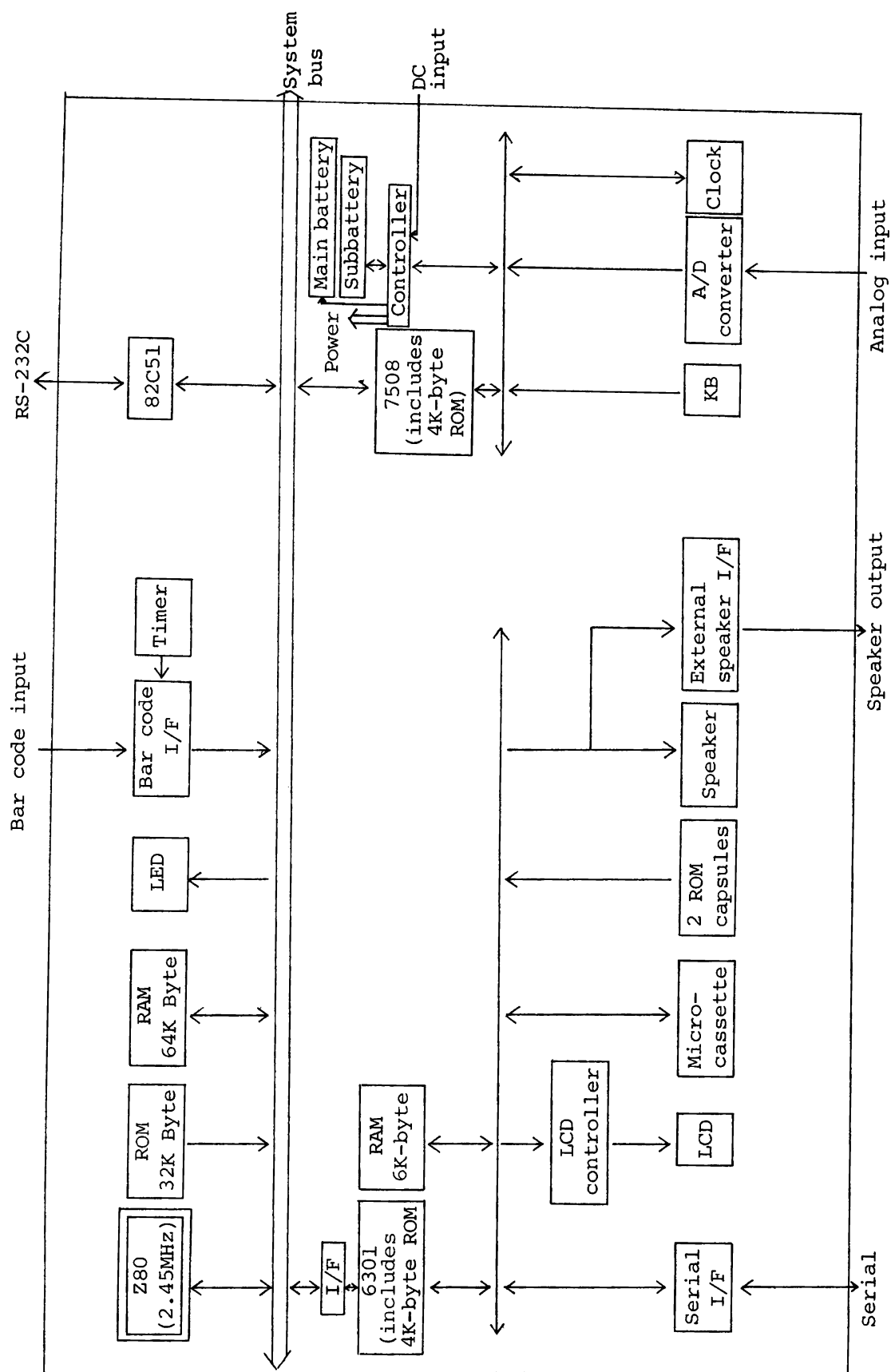
2.3 Software Features and Organization

2.3.1 Software Features

This subsection lists the features of the MAPLE software.

(1) Industry-standard CP/M 2.2

This allows the user to transport an abundance of commercial CP/M application programs to the MAPLE with relatively little effort.

(2) A variety of peripheral devices supported by OS.

The peripheral devices that MAPLE CP/M 2.2 supports are:

RAM disk

ROM capsule

Mini- and micro-FD

Microcassette

Speaker

Analog input

RS-232C

Power

Clock (calendar)

To support these devices, 25 BIOS entries have been included into the standard CP/M BIOS. Consequently, the user can develop application programs handling these devices with great ease.

The OS, however, supports no bar code equipment. It
must be handled by application programs. These programs
are also supplied from EPSON.

(3) Many CP/M drives

The table below lists the peripheral devices that are supported as CP/M drivers.

Drive	Peripheral	Capacity		Direc-tories	Maximum tracks/sector
		Total	Data area		
A:	RAM disk	When main RAM is used: Ø - 23K bytes	Ø - 23K bytes	16	3 TRK/7 SCT
		When RAM disk is used: 6ØK bytes 64K bytes 12ØK bytes 128K bytes	 59K bytes 63K bytes 119K bytes 127K bytes	 32 32 32 32	 7 TRK/31 SCT 7 TRK/63 SCT 14 TRK/63 SCT 15 TRK/63 SCT
B: C:	ROM capsule	Depends on ROM type 8K bytes 16K bytes 32K bytes Sum of the above capacities when drives B: and C: are used as a contiguous drive.	 8K bytes 16K bytes 32K bytes ←	 31 31 31	 Ø TRK/63 SCT 1 TRK/63 SCT 3 TRK/63 SCT
D: E: F: G:	FD	320K bytes	278K bytes	64	39 TRK/64 SCT

Drive	Peripheral	Capacity		Direc-tories	Maximum tracks/sector
		Total	Data area		
H:	Microcassette	Approx. 3ØK bytes with 3Ø-minute tape	Approx. 3ØK bytes with 3Ø-minute tape	12	Only sequential access in file units is allowed.
I:	ROM capsule in extension unit Supported by a combination of OS ASCII version B or later and a Multi-unit 64.	128K bytes	128K bytes	31	15 TRK/63 SCT
	ROM capsule in extension unit Supported by a combination of Japanese-language OS and a Japanese-language unit.	32K bytes	32K bytes	31	3 TRK/63 SCT

(4) RAM disk features

-    Allows both reads and writes.

-    High-speed access.

-    Provides a storage capacity of 23K bytes maximum
     when main memory is submitted as RAM disk and a
     capacity of 128K bytes when an extension unit is
     installed.

-    Data is maintained even when power is turned off.

-    The main memory RAM disk is disabled when the
     extension unit RAM disk is used.

(5) ROM capsule features

-    Allows only reads.

-    High-speed access

-    Provides a storage capacity of 8K bytes (one 2764
     chip) to 64K bytes (two 27256 chips).

-    Easily installed and removed.

(6) FD features

-    Allows both reads and writes.

-    High-speed access.

-    Provides a large capacity of removable storage.

-    Can handle both mini- and micro-floppy disk drives.

-    The micro-FD drives (PF-10) is battery driven.

(7) Microcassette

-    Allows both reads and writes.

-    High-speed access.

-    Customized OS allows the user to handle
     microcassette in the same easy way as FD files.

-    Only sequential access is supported.

-    Only one file can be open at a time.

(8) Extended unit ROM capsule

-    Allows only reads.

-    High-speed access

-    ASCII OS supports larger capacity than internal ROM
     capsules.

(9) Devices for software exchange

The user can select any of the following devices for

exchanging storage media of different sizes and

formats:

- ROM capsule

- FD

- Microcassette

(10) Screen

The MAPLE is provided with a large (80 columns by 8 lines) LCD. Its OS also supports virtual screens as large as 80 columns by 48 lines. The user can switch between four screens, namely, i.e., three text only screens (the 80-column text screen, the 39-column Split screen, and the Dual screen) and one graphics screen, all under software control. In addition to these screens, the Japanese-language OS supports two types of kanji screens. It also permits switching of virtual screens and control of screen scrolling with function keys.

(11) Clock

The MAPLE has a clock which indicates the year (lowest two digits, month, day, minutes, and second). The clock is battery backed up and performs automatic leap year adjustment.

(12) Password function

The password function protects the MAPLE programs and
data from unauthorized accesses. Once a password is
defined, this function defers any attempt for a
power-on sequence until the operator enters the defined
parameter.

(13) Alarm function

The alarm function sounds an alarm at the preset time,
whether the MAPLE is in use or not, and displays the
predefined messages on the screen. This function can be
used for schedule management.

(14) Wake function

The wake function automatically powers on the MAPLE and
executes programs in the preprogrammed sequence when the
preset time (month, day, and hour) has reached. If the
MAPLE is already in on state when the preset time is
reached, this function sounds an alarm and displays
messages indicating the operating procedure for the
programs (alarm function). This function can be used
for automatically starting the MAPLE in instrumentation
and data gathering applications.

(15) Auto start function

The auto start function performs the steps or programs
predefined by the user automatically at power on time.
This function will be useful when the MAPLE is used as a
dedicated machine.

(16) Menu function

The menu function displays a directory of executable
programs on the screen in a menu format at power on or
warm boot time. The user can select the program with
cursor movement keys and start the selected program by
pressing the RETURN key. This function is highly
convenient for users who are unfamiliar with operator
operations. When a program is already in the TPA, this
function causes the program in memory to be immediately
executed, thus eliminating the time-consuming program
load step.

(17) System display function

The system display function is started by pressing the
CTRL and HELP keys simultaneously and displays the
system status on the screen. The user can define the
parameters for the password, alarm wake, auto start, and
menu functions from the screen. The function also
allows the user to manually control the microcassette
drive.

(18) Hard copy function

The user can take a hardcopy of the current contents of

the LCD screen on the printer in one of the

following methods:

(1)   Pressing the CTRL/PF5 key.

(2)   Calling the BIOS hardcopy routine.

Some OS versions do support the hard copy function.

(19) Power off state in restart and continue modes

The MAPLE can be in one of the two power off modes,

i.e., restart and continue modes, depending on how the

MAPLE is powered off.

- Restart mode: Execution starts at CCP or a menu is

                displayed when the MAPLE is powered on.

- Continue mode: The processing that were being executed

                when the MAPLE was powered off is resumed

                when the MAPLE is powered on again.

(20) Power on/off

The MAPLE can be powered on and off not only through the
POWER switch but under program control. The MAPLE can
be turned on by the wake function and turned off by a
BIOS routine. The user can also set the restart or
continue mode. The contents of MAPLE memory are
maintained when MAPLE power is turned off.

(21) Auto power off function

The auto power off function automatically turns MAPLE
power off in the continue mode when no key entry is made
for a predetermined time, thus saving battery power.
When the MAPLE is powered on again, executions resumes
at the point when the auto power off function is
executed.

(22) Voltage drop warning

When the battery voltage drops below approximately 4.7
volts, the OS displays a message "CHARGE BATTERY" on the
screen and, in approximately 20 seconds later,
automatically turns MAPLE power off. This precludes the
contents of the RAM from being completely destroyed or
the CPU from hanging up due to the reduced battery
voltage level. When this occurs, the active battery is
automatically switched to the subbattery which only
maintains the power to RAM.

2.3.2 Software Organization

The MAPLE OS resides in the 32K-byte ROM. The OS runs
while switching between the RAM and ROM banks. The OS
contains the modules listed below.

Module	Function
STARTER	Resides in ROM and performs the following: - System initialize - RESET switch processing - POWER switch processing - Processing of alarm interrupts in power off state.
INTROM	Resides in ROM and processes interrupts from the 7508 and 8251.
MENU	Resides in ROM and controls menu processing.
SYSCRN	Resides in ROM and controls system display processing.
RELOC	Resides in ROM and relocates RAM resident modules from ROM.

Module	Function
BDOS	Resides in ROM and processes CP/M BDOS calls.
PREBIOS	Resides in ROM and perform preprocessing for CP/M BIOS calls.
PSTBIOS	Resides in ROM and perform postprocessing for CP/M BIOS calls.
BIOS1 BIOS2 BIOS3	Resides in ROM and processes CP/M BIOS calls. The BIOS module is divided into three submodules.
SCREEN	Resides in ROM and controls CONOUT BIOS call processing.
MCT	Resides in ROM and controls the microcassette drive.

Module	Function
CCPD	The CCP portion of CP/M in a relocatable format and is relocated into RAM at the beginning of execution.
RBDOSB	The BDOS entry portion of CP/M (main BDOS body resides in ROM) in a relocatable format and is relocated into RAM at the beginning of execution.
RSYSPR	The part of the CP/M BIOS entry portion (main BIOS body resides in ROM) in a relocatable format and is relocated into RAM at the beginning of execution. Includes interrupt handling routines and other system routines.
SYSAR1 SYSAR2 SYSAR3	Copied into RAM and initialize the system work area. There are three modules which are invoked at different timings depending on when the work area is to be initialized.
ROMID	Contains the OS ROM identification.

2.4 MAPLE State Transition

The MAPLE can be in eight states when viewed from the software standpoint. The interrelationship between these eight states is illustrated in the figure on the next page.

MAPLE states

(1) Restart mode power off state

(2) Continue mode power off state

(3) Password entry screen display state

(4) Menu screen display state

(5) System display screen display state

(6) Alarm/wake screen display state

The MAPLE performs no special wake function except it displays messages (strings) indicating operating procedures in the same way as the alarm function when a wake time is reached in the power on state.

(7) Charge battery screen display state

(8) CCP or application program running state

States 1 through 7 are unique to the MAPLE and only supported by the MAPLE OS.

Note: Power failure refers to a drop in the battery voltage below a specified level.

MAPLE State Transition Diagram
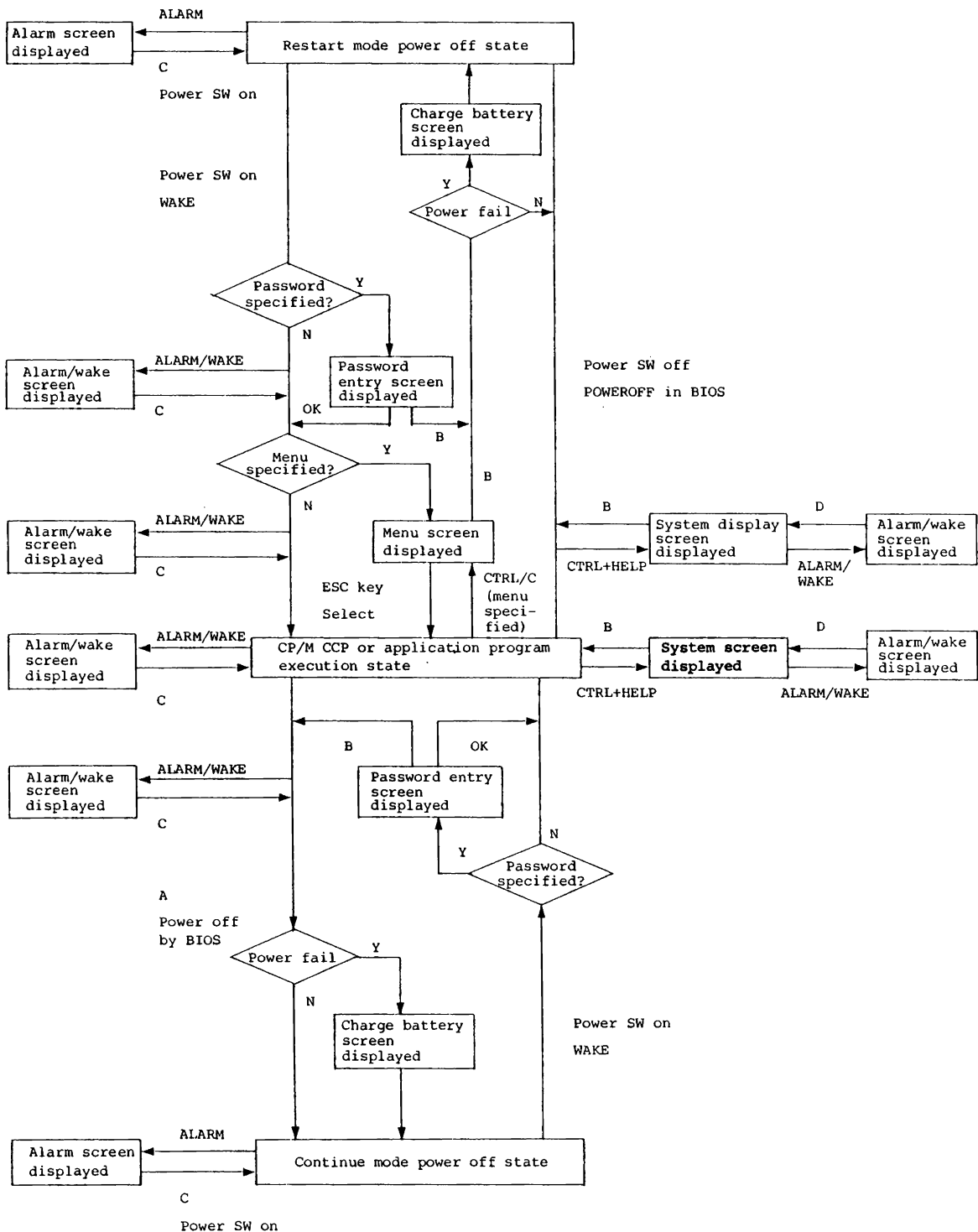
A – CTRL + Power SW off, Auto power off, Power failure
B – A, Power SW off
C – 5Ø seconds, ESC Key, Power SW off, CTRL + Power SW
    off, Power failure
D – C, Power SW off

Chapter 3 MAPLE CP/M Principles of Operation

MAPLE adopts CP/M Version 2.2 as its operating system. Since the basic part of the MAPLE operating system is implemented in ROM, MAPLE CP/M runs in a slightly different way from the CP/M for most disk-based computers. This chapter explains how MAPLE CP/M run on the MAPLE computing system.

3.1 CP/M Memory Organization

3.1.1 Roles of CP/M Modules in ROM and RAM

MAPLE CP/M switches between two 32K-byte banks during execution using a bank switching technique as shown in the figure on the next page. One is a ROM bank containing the major portions of CP/M OS and the other is a RAM bank which makes up the first half of the 64K main RAM memory. The CP/M modules (CCP, BDOS, and BIOS) are apparently loaded in RAM as they are on ordinary disk-based computers. This means that MAPLE application programs can use the CP/M functions in the same way as those which use the standard CP/M. In fact, however, only a 100H bytes of a system area containing the entry points to the BDOS and BIOS are loaded in RAM, making the most part of the RAM

available for application programs. Actual BDOS and
BIOS operations are performed in the OS in ROM that is
activated through bank switching. Control is returned
to the application program again through bank switching
to RAM after processing is terminated.

```
ØØØØH ┌──────────────┐          ┌──────────────┐  ØØØØH
      │              │          │ CP/M system  │
      │              │          │    area      │
      ├──────────────┤          ├──────────────┤  Ø1ØØH
      │    BDOS      │          │              │
      │              │          │              │
      ├──────────────┤          │              │
      │   BIOS1      │          │     TPA      │
      │   BIOS2      │          │              │
      │   BIOS3      │          │              │
      ├──────────────┤          │              │
      │   Screen     │          │              │
      │              │          │              │
      ├──────────────┤          ├──────────────┤
      │              │          │     CCP      │ } 8ØØH
      │              │          ├──────────────┤
      │              │          │    BDOS      │ } 1ØØH
7FFFH └──────────────┘          ├──────────────┤
                                │    BIOS      │ } 1ØØH
                                ├──────────────┤
                                │   RAM Disk   │
                                │              │   Ø to 24K bytes
                                ├──────────────┤   in total
                                │  User BIOS   │
                                ├──────────────┤
                                │   Work area  │
                                │              │
                                │              │
                                └──────────────┘
```
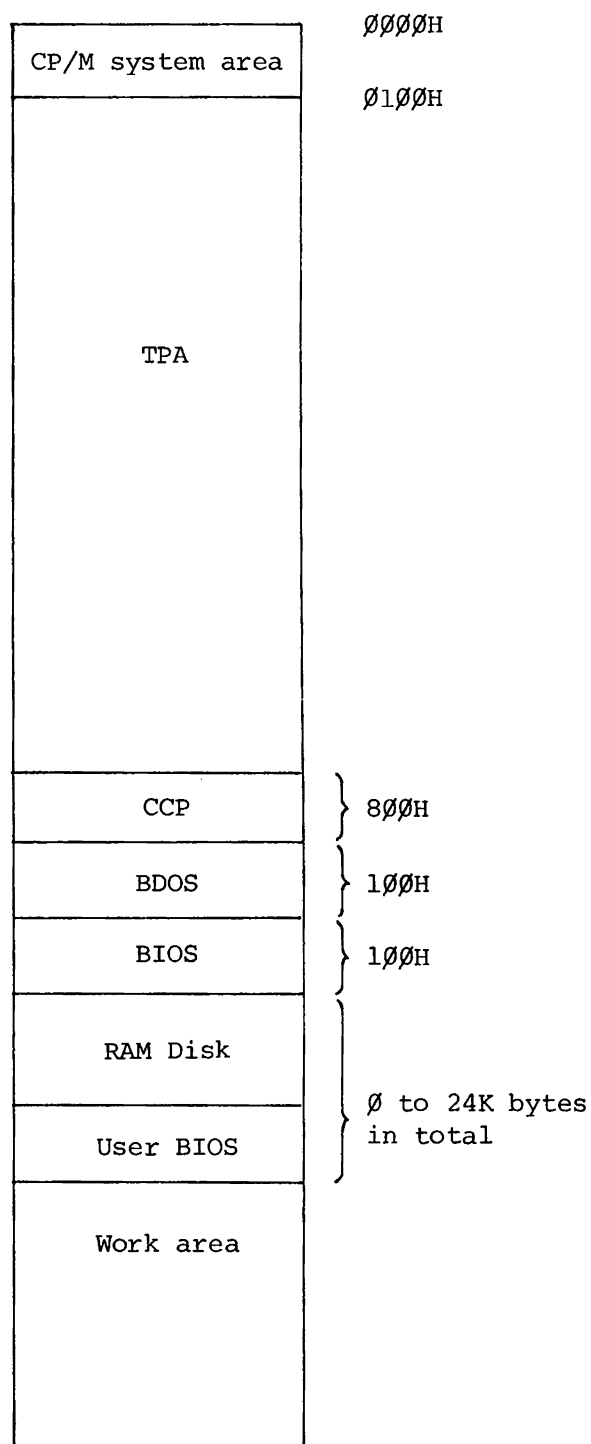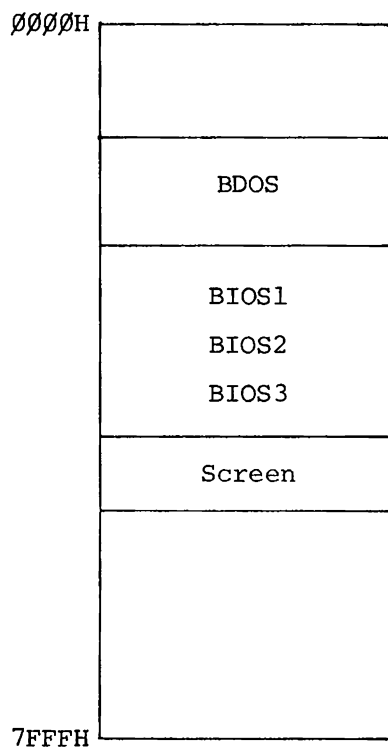
The addresses of the CCP, BDOS, and BIOS in RAM differ
depending on the total size of the RAM disk implemented
and the user BIOS area (0 - 24K bytes). The size of
the CP/M system ranges from 59.5K to 45.5K bytes. The
RAM disk and user BIOS sizes can be changed by the
CONFIG program.

3.1.2 Procedure for Constructing a CP/M System in RAM

On MAPLE, the CP/M system can be loaded from ROM into RAM by three routines: system initialize, reset (CBOOT), and WBOOT. This subsection describes the function of these routines and the timing when they are invoked as well as the interactions between them. The STOP and CTRL/STOP functions for interrupting program execution are also explained here.

The user can take the following measures (must be attempted in this sequence) when his program hangs up:

1. Press the STOP key.

2. Press STOP key while holding down the CTRL key.

3. Press the RESET switch.

4. Hold the SHIFT and GRPH keys simultaneously and press the RESET switch on the side of MAPLE.

5. Press the 75Ø8 RESET switch.

	Operation	Initiated by	When
System initialize	1. Initializes system area 1. 2. Resets the slave CPU 6301. 3. Checks the extended RAM disk unit. 4. Performs system initialization. Sets the year, month, day, hour, minute, and second and the size of the RAM disk and User BIOS. (ASCII Version 1.0 and Japanese-language OS) 5. In ASCII Version B, the system initialization formats the RAM disk and only initializes the system as follows without performing system initialization: Date and time: 1900/00/00 00:00:00 Sunday RAM DISK: 9K bytes User BIOS: 0 bytes	1. Pressing the 7508 RESET switch. 2. Holding the SHIFT and GRPH keys down and pressing the RESET switch.	1. Using the system for the first time after purchase. 2. 7508 hangs up. 3. The extension unit is installed or removed.
Reset	1. Initializes system area 2. 2. Resets the slave CPU 6301. 3. Loads the BIOS to RAM. 4. Checks the RAM disk checksum. 5. Sets the screen to the mode specified by CONFIG.	1. Pressing the RESET switch.	1. Z80 hangs up. 2. 6301 hangs up.
WBOOT	1. Flushes the FD buffer. 2. Sets the cursor to the mode set up by CONFIG.	1. Entering C or the STOP key in CCP mode. 2. Sending control to WBOOT in the program. 3. Turning power on in restart mode.	1. Control jumps to BIOS WBOOT.

	Operation	Initiated by	When
CTRL/STOP	1. Interrupts the current I/O operation. 2. Clears the key buffer and loads it with Ø3H.	1. When holding down the CTRL key and pressing the STOP key.	1. Interrupting application program performing an I/O operation. The application program must terminate on receiving Ø3H.
STOP	1. Clears the key buffer and loads it with Ø3H.	1. Pressing the STOP key.	1. When interrupting the application program must terminate on receiving Ø3H.

Relationships among the system initialize, reset, and WBOOT

```
┌─────────────────────────────┐
│     System initialize       │        75Ø8 is reset.
└─────────────────────────────┘        SHIFT/GRPH + RESET
              │
              ▼
┌─────────────────────────────┐
│           Reset             │        RESET switch
└─────────────────────────────┘
              │
              ▼
┌─────────────────────────────┐
│           BOOT              │       - Execution jumps to WBOOT.
│ 1. Sets the currently       │       - Power is turned on in restart
│    logged in drive to A:.   │         mode.
│ 2. Sets the I/O byte to     │
│    A9H.                     │
│ 3. Displays the CP/M sign-  │
│    on message.              │     ┌─────────────────────────────┐
│ 4. Sets the keyboard and    │     │           WBOOT             │
│    character set types      │     └─────────────────────────────┘
│    according to the DIP     │
│    switch settings.         │
└─────────────────────────────┘

        ┌──────────────────────────────────────────┐
        │   Actions common to BOOT and WBOOT       │
        │ 1. Turns RS-232C power off.               │
        │ 2. Turns ROM capsule power off.           │
        │ 3. Stops the micro cassette.              │
        │ 4. Creates the BIOS and BDOS jump vectors in │
        │    the CP/M system area.                  │
        │ 5. Loads BDOS to RAM.                     │
        └──────────────────────────────────────────┘
                          │
                          ▼
        Loads CCP to RAM and transfers control to CCP
        or the menu display routine.
```

System areas 1, 2, and 3

The RAM work area that MAPLE uses is classified into the following two types:

1. Work areas initialized at a specific timing before use.
2. Work areas used only temporarily.

The work area of the first type is divided into three types called system areas 1, 2, and 3 according to the timing at which initial values are set.

	Initialized when	Work area contents
System area 1	System initialize is invoked.	Initial values of flags indicating PASSWORD and MAPLE basic status.
System area 2	Reset is invoked.	Initial values related to BIOS.
System area 3	WBOOT is invoked.	Initial values related to BDOS.

3.2 BDOS Function Processing Flow

When BDOS is called by a MAPLE application program, control is first transferred to the entry point to the BDOS in RAM. Then the OS switches banks and maps the memory addresses 0000H to 7FFFH into ROM, then calls the real BDOS in ROM. Upon completion of processing, the OS switches the bank to RAM and returns control to the application program with return information loaded in registers.

The BDOS in ROM calls directly the BIOS in ROM.

BDOS call processing flow

(Bank 0)   (Bank 1)

0000H   0005H   0000H

JMP BDOS

①

(0005H)
CALL BDOS

BDOS   ②

⑤   ④

BIOS   ⑦

③

⑥

7FFFH

CCP

┌─────────────────┐
│   │ ⇐──────   BDOS   ─── } 100H
│ ROM bank   │
│ selected.   │   BIOS
│   │
│ BDOS in ROM   │   RAM DISK
│ called.   │   &
│   │   User BIOS
│ RAM bank   │
│ selected.   │   System resident
│   │   area and work
└─────────────────┘   area

                                                                 FFFFH

3-11
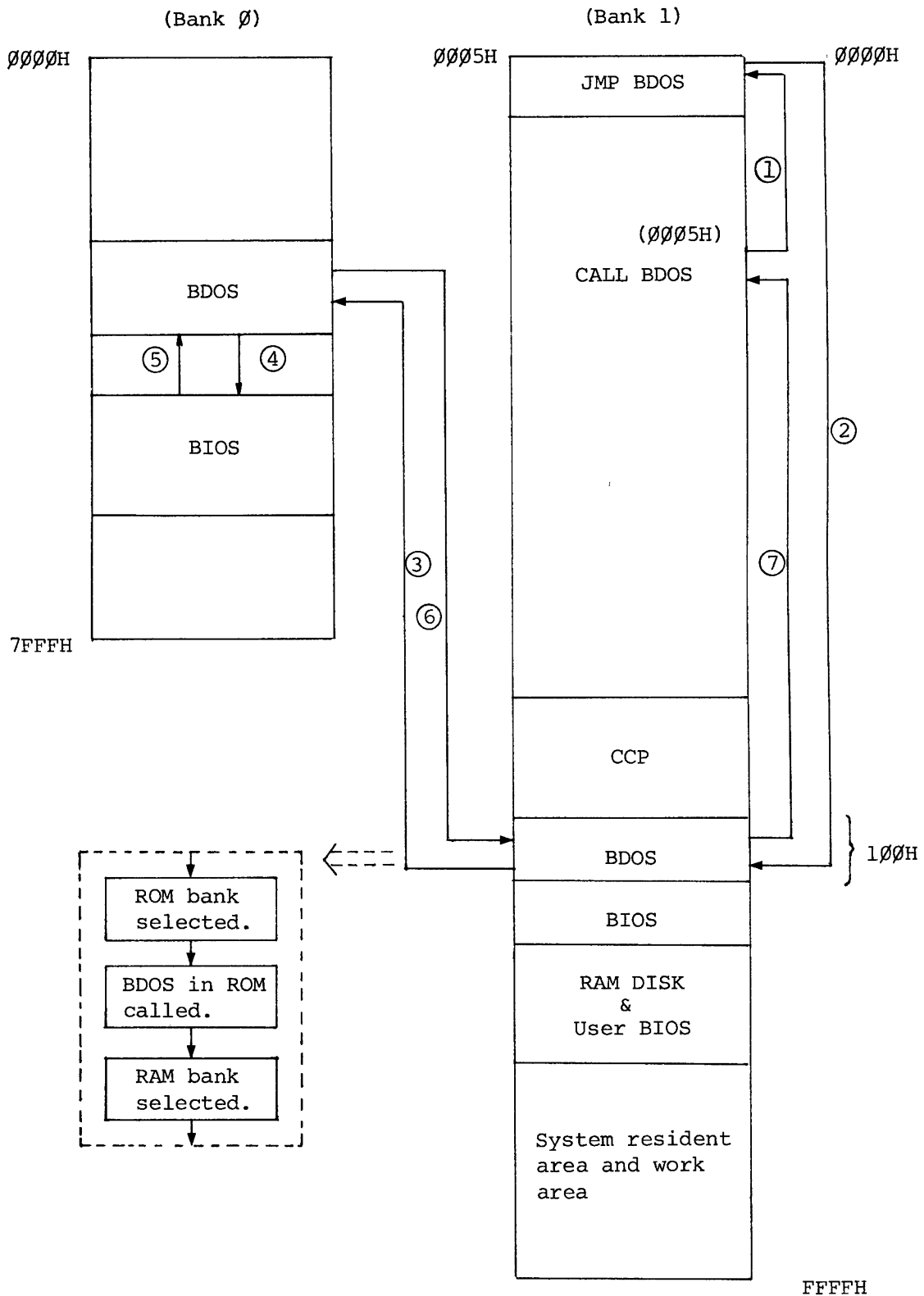
3.3 BDOS Error Recovery Procedure

BDOS can display four types of error conditions. Since these errors are handled totally under BDOS control, it is likely that they destroy the current screen image, initiates a warm boot on receipt of user response from the keyboard after the error display, or even destroy memory data. One of countermeasures to avoid this is to make the application program report and handle error conditions for itself. The MAPLE OS permits the application program to take the following two measures against error conditions to achieve this:

1.   Receiving BDOS error information as a return code.

2.   Rewriting the jump vector for BDOS error processing and performing user-supplied error processing.

The four types of BDOS errors are:

1.   Bad Sector

2.   Bad Select

3.   R/O Disk

4.   R/O File

3.3.1 Receiving BDOS Error Information in Return Code

(1) Changing the BDOS error reporting mode

The application program can receive any BDOS error
information directly in CPU registers by calling
location 0012H (SET ERROR) in OS ROM (bank 0). It can
also have BDOS return any error information by calling
location 0015H (RESET ERROR) in OS ROM.

The application program must use BIOS CALLX (WBOOT +
69H) to directly call a routine in OS ROM. In this
case, the program must reserve a stack area at a
location 8000H or higher in RAM.

(2) Return codes

Error \ Register	A	H	
BAD SECTOR	FFH	Ø1H	
BAD SELECT	FFH	Ø2H	Standard CP/M BDOS errors
R/O DISK	FFH	Ø3H	
R/O FILE	FFH	Ø4H	
MCT ERROR	FFH	Ø5H	BDOS errors unique to MCT

For Bad Sector errors, BDOS stores more detailed error information in memory.

BIOSERROR EQU 0F536H

Data in memory	Error type	
Ø1H	Read error	
Ø2H	Write error	
Ø3H	Write protected.	←─Write protected
Ø4H	Timeout	
Ø5H	Seek error	(MCT only)
Ø6H	CTRL/STOP pressed.	(MCT only)
Ø7H	Power turned off.	

(3) Procedure for identifying errors

Some of the BDOS functions returns 0FFH to the A
register as a usual return code. Therefore, the calling
program must identify errors by examining the H register
as well as the A register. See the figure below.

```
                          │
                          ▼
                ╔══════════════════╗
                ║  SET ERROR       ║
                ║  processing      ║
                ╚══════════════════╝
                          ┊
                          ▼
                ╔══════════════════╗
                ║  BDOS call       ║
                ╚══════════════════╝
                          │
                          ▼            N
                     ◇ Areg = FFH ◇──────────────┐
                          │                        │
                          │ Y                       │
                          ▼                        │
                     ◇ Hreg = 00H ◇──────────────▶│
                          │                        │
                          ▼                        ▼
                   BDOS error            Normal termination
                          ┊                        ┊
                          └────────────────────────┘
                          ┊
                          ▼
                ╔══════════════════╗
                ║  RESET ERROR     ║
                ║  processing      ║
                ╚══════════════════╝
                          │
                          ▼
```
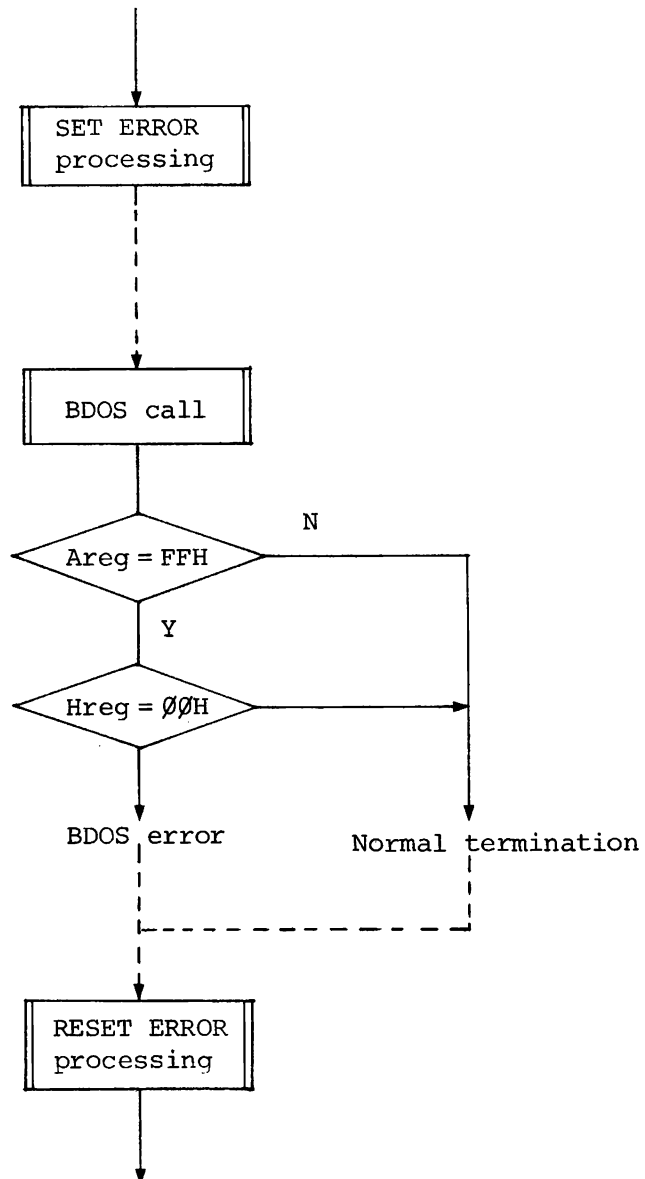
(4) Programming notes

1) Once SET ERROR is executed, BDOS performs no error
    processing and continues only to return error
    status until a RESET ERROR or WBOOT is executed.

2) After execution of SET ERROR, the results are not
    guaranteed unless the application program performs
    its own error checking and recovery processing.

3.3.2 Rewriting the Jump Vector for Processing BDOS Errors

Four jump vectors for processing BDOS errors are located
at the beginning of BDOS in RAM. The application
program can handles error conditions in its own way by
changing the contents of these jump vectors.

ERRVCTR: <------------------

          Address ((Contents of RAM addresses 6 and 7)+3)

    DW   PERERR <---- Address of parameter error processing routine
                      (Bad Sector error)

    DW   SELERR <---- Address of select error processing routine
                      (Bad Select error)

    DW   RODERR <---- Address of read only disk error processing
                      routine (R/O Disk error)

    DW   ROFERR <---- Address of read only file error processing
                      routine (R/O File error)

    The application program can perform its own error

    processing by changing the above addresses.

Programming notes:

(1) On return, the stack area is switched to that for the application program because the stack area for the BDOS was used during BDOS processing.

(2) Bank 1 is selected (all RAM).

(3) The user error processing routine must contain no BDOS calls if it is to return control to BDOS with a RET statement.

3.4 BIOS Function Operation Flow

(1) Outline

The major BIOS operations are carried out by BIOS in ROM as BDOS operations are. To achieve these, when a call to BIOS is made from an application program, the OS receives the call in the BIOS in RAM, switches the active bank to the system bank, and calls the BIOS in the system bank (ROM). After completion of the BIOS processing, the OS returns to the BIOS in RAM with various return information and result data, switching again to the user bank, and returns control to the application program.

The BIOS in RAM always resides in addresses higher than 8000H so that it is not affected by bank switching.

(2) PREBIOS and PSTBIOS

Some BIOS routines uses the slave CPU functions (e.g.,
screen and microcassette handling). Since the main and
slave CPUs communicate commands and data using a
specific protocol, if the main CPU attempts to request
the slave CPU to do one operation while it has already
instructed the slave CPU to do another operation, the
protocol will be destroyed and the communication between
the main and slave CPUs hang up. BIOS controls the
slave CPU properly while BIOS alone is using the slave
CPU. If, however, an interrupt is generated which calls
for a service by the slave CPU (e.g., alarm, power off,
or power failure interrupt), it will try to have the
interrupt source use the slave CPU, ignoring the
execution sequence established between the main and
slave CPU, causing the MAPLE to hang up.

PREBIOS and PSTBIOS are provided to solve this problem.
When a call is made to BIOS, the OS executes PREBIOS to
set on a flag indicating that BIOS processing is in
progress. If an interrupt requesting for a slave CPU
service is generated while this flag is on, the
interrupt handling routine checks this flag and, knowing
that the slave CPU is used by a BIOS routine, makes the

interrupt-driven processing pending after turning on a
flag indicating that an interrupt is held pending.

When the BIOS processing is completed, the OS starts
PSTBIOS, which in turns executes any pending interrupt
routines, clears the flag indicating the execution of a
BIOS routine, and returns control to the application
program.

The flowchart on the next page shows the
relationship between PREBIOS, PSTBIOS, and BIOS
processing.

```
                          ┌─────────────┐
                          │    ENTER    │
                          └──────┬──────┘
                                 │
  PREBIOS                        │
                          ┌──────┴──────┐
                          │ BIOS in-process │          Interrupt generated
                          │ flag set to on  │         ╱
                          └──────┬──────┘        ╱
  ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ┼ ─ ─ ─ ─ ╱
                                 │      ╱                    ◇
                          ┌──────┴──────┐              ◇ BIOS being ◇ ── Y
     BIOS                 │    BIOS     │              ◇ executed?  ◇
                          │ processing  │                    ◇
                          └──────┬──────┘                  N │
  ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ┼ ─ ─          ┌──────────┐   ┌──────────────┐
                                 │              │ Interrupt │   │ Interrupt flag│
  PSTBIOS        N               │              │processing │   │ set to on     │
                 ◇───────────────◇              └────┬─────┘   └──────┬───────┘
                 ◇ Interrupt     ◇                   │                │
                 ◇  flag on?     ◇                   └────────┬───────┘
                       ◇                                      │
                       │ Y
                ┌──────┴──────┐
                │  Interrupt  │
                │ processing  │
                └──────┬──────┘
                       │
                ┌──────┴──────┐
                │ BIOS in-process │
                │ flag set to on  │
                └──────┬──────┘
                       │
                ┌──────┴──────┐
                │    EXIT     │
                └─────────────┘
```

3-22
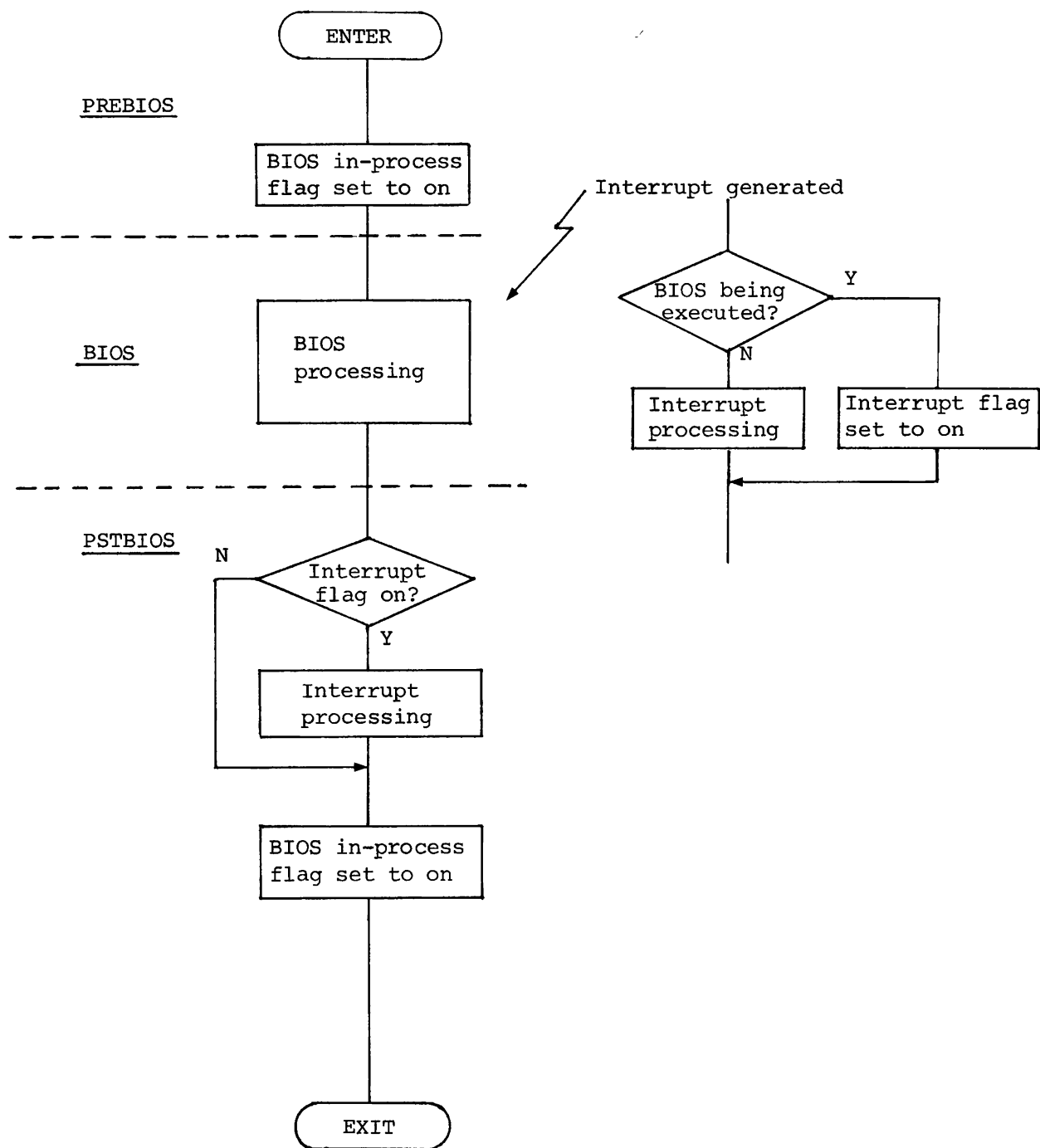
(3) Calling a BIOS routine from an application program

The entry address into the BIOS WBOOT in RAM is located
in addresses 1 and 2 in RAM. To use a BIOS call,
the user program must call BIOS specifying the address
obtained by adding the function offset to this BIOS
entry address. Since every BIOS routine ends with a
RET statement, control returns the statement immediately
following the CALL statement that called the BIOS
routine.

SAMPLE PROGRAM

The sample program below calls a BIOS routine with the
function's offset from the WBOOT (multiple of 3) in the
IX register pair.

BIOS:

```
    PUSH        BC

    LD          BC, (0001H)     ;Entry point to WBOOT.

    ADD         IX, BC

    POP         BC

    JP          (IX)            ;Jump to BIOS.
```

(4) BIOS call operation flow
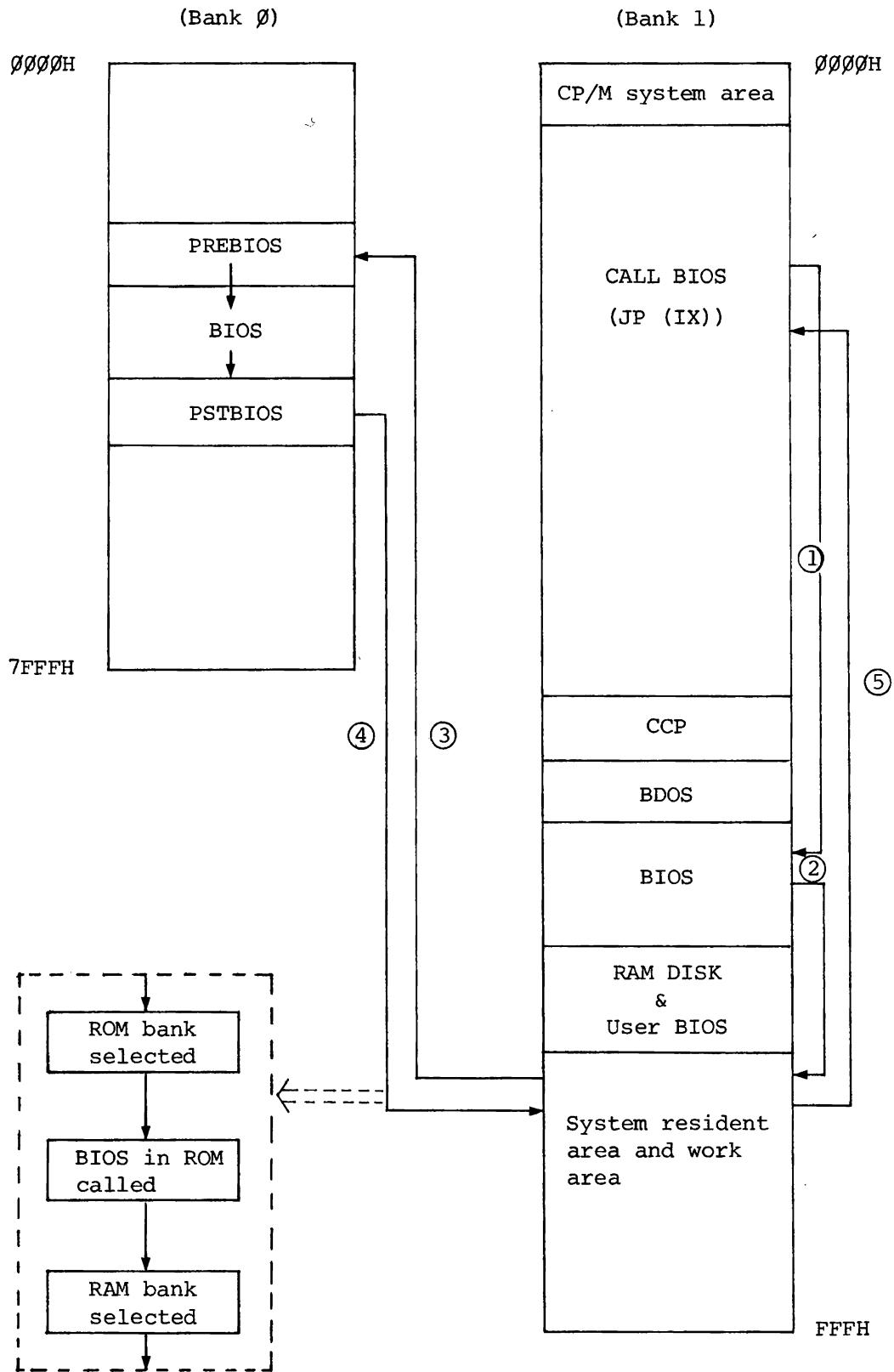
Chapter 4   BIOS Subroutines

MAPLE BIOS is greatly extended for support of a number
of I/O functions. In fact, it contains as many as 44
subroutines. This chapter gives a detailed description
of these subroutines. The BIOS CONOUT routine has many
options, and therefore, a whole chapter is reserved for
it for full description of the function (see Chapter 5).

* Programming Notes on the use of BIOS calls
1) The entry to each function is indicated by the offset
from WBOOT. Find its effective address by adding this
offset to the entry address to the WBOOT located in 01H
and 02H.
2) Save the contents of registers if necessary because
the contents of the registers except those for receiving
the return parameter are not guaranteed.

The entry addresses and functions of BIOS Subroutines

Offset from WBOOT	ENTRY NAME	Function
−Ø3H	BOOT	Performs a cold BOOT.
±ØØH	WBOOT	Performs a warm BOOT.
+Ø3H	CONST	Returns the console input status.
+Ø6H	CONIN	Inputs one character from the console.
+Ø9H	CONOUT	Outputs one character to the console.
+ØCH	LIST	Outputs one character to the LIST device.
+ØFH	PUNCH	Outputs one character to the PUNCH device.
+12H	READER	Inputs one character from the READER device.
+15H	HOME	Positions the disk head to track ØØ.
+18H	SELDSK	Specifies the device.
+1BH	SETTRK	Specifies the track for read or write.
+1EH	SETSEC	Specifies the sector for read or write.
+21H	SETDMA	Specifies the DMA starting address for read or write.
+24H	READ	Reads the specified sector.
+27H	WRITE	Writes data to the specified sector.
+2AH	LISTST	Returns the status of the list device.
+2DH	SECTRN	Translates a logical sector to a physical sector.
+3ØH	PSET	Converts graphics screen data for display.
+33H	SCRNDUMP	Takes a hard copy of the displayed data.
+36H	BEEP	Sounds the speaker.
+39H	RSOPEN	Opens the RS-232C interface.

Offset from WBOOT	ENTRY NAME	Function
+3CH	RSCLOSE	Closes the RS-232C interface.
+3FH	RSINST	Informs whether the RS-232C interface has received data.
+42H	RSOUTST	Checks whether the RS-232C interface is ready for transmission
+45H	RSIN	Receives one character from the RS-232C interface.
+48H	RSOUT	Transfers one character to the RS-232C interface.
+4BH	TIMDAT	Performs clock or alarm functions.
+4EH	(MEMORY)	Does nothing.
+51H	RSIOX	Performs RS-232C functions.
+54H	(LIGHTPEN)	Does nothing.
+57H	MASKI	Sets or resets the interrupt mask.
+5AH	LOADX	Reads the data in the specified bank.
+5DH	STORX	Writes data into the specified bank.
+60H	LDIRX	Transfers data between banks.
+63H	JUMPX	Jumps to the specified bank address
+66H	CALLX	Calls the subroutine at the specified bank address.
+69H	GETPFK	Gets a PF key.
+6CH	PUTPFK	Defines a PF key.
+6FH	ADCVRT	Performs analog data input operations.
+72H	SLAVE	Processes communication with the SLAVE CPU 6301.
+75H	RDVRAM	Reads the contents of VRAM.
+78H	MCMTX	Processes communication with MIOS.

Offset from WBOOT	ENTRY NAME	Function
+7BH	POWEROFF	Turns main power off.
+7EH	USERBIOS	Entry point to the User BIOS.

Entry Name	BOOT	Entry Address	WBOOT - 03H
Function	Performs a CP/M cold boot.		
Entry parameter	None.		
Return parameter	None.		
Explanation			

BOOT is entered by a 7508 or system initialize reset (SIFT/GRPH/RESET), or the depression of the RESET key. This routine is used not by application programs but by the operating system.

BOOT performs the following:

1. Sets the current drive to A:.

2. Sets the I/O byte to 10101001B.

     LST: = LPT: (RS-232C)

     PUN: = UP1: (RS-232C)

     RDR: = UR1: (RS-232C)

     CON: = CRT: (Output: LCD, Input: Keyboard)

3. Displays the CP/M sign-on message.

4. Reads informations of the DIP switches and saves their settings in a work area to identify the keyboard (nationality) and the character set to be used.

5. Loads the CTRL/HELP entry in the keyboard subroutine table with the system display address and the CTRL/PF5 entry with the hardcopy address.

6. Sets the pointer to the PF key table to the system table.

7. Initializes the cursor movement key (arrowed key) codes.

8. Jumps to the routine shared with WBOOT.

Entry Name	WBOOT	Entry Address	WBOOT +0H
Function	Performs a CP/M warm boot.		
Entry parameter	None.		
Return parameter	None.		
Explanation			

WBOOT is entered when power is turned on in restart mode or a JUMP 0 is executed.

WBOOT performs the following:

1. Writes the write data left in the FDD buffer into the floppy disk.

2. Initializes the MCT parameters.

3. Restores the cursor into the state defined by CONFIG.

4. Sets the pointer to the PF key table to the system table.

5. Displays the PF key definitions on line 8 when PF key display mode is specified.

The following processing is common to WBOOT and BOOT:

6. Sets SP to the value for BIOS.

7. Turns the RS-232C interface power off.

   Turns the ROM capsule power off.

   Stops the microcassette.

8. Loads addresses 0 to 2 with the object code of JP WBOOT.

   Loads addresses 5 to 7 with the object code of JP BDOSE.

                                   RAM BDOS starting address + 6

9. Loads BDOS into RAM.

The subsequent actions of WBOOT depends on the system conditions
under which it has executed so far. The actions are shown in the
flowchart on next page.

10

Restart mode powered on?
— N →

Powered on during an alarm generated by a power off?
— Y →

— N →

Powered on by wake?
— N →

— Y →

Auto start string set?
— N →

Wake string set?
— N →

©

Y ↓

Y ↓

MENU display specified?
— N →
ESC entered.

Y ↓

Resident selected.

Move start string to KEY INPUT.

Move wake string to KEY INPUT.

Display MENU.

Ⓐ

Ⓑ

CR entered.

Move command line contents to KEY INPUT.

Go to TPA program.

Load CCP into RAM and transfer control to CCP.

A: When an auto start string is specified and

   - The power switch is turned on.

   - The power switch is turned on while an alarm generated in

     the power off state is being displayed.

B: When power is turned on by wake with a wake string specified.

C: - After BOOT is executed.

   - After WBOOT is executed.

   - When power switch is turned on by wake with no wake string

     specified.

Entry Name	CONST	Entry address	WBOOT + 03H
Function	Returns the status of the console.		
Entry parameter	None.		
Return parameter	A = 00H: Console input buffer is empty. A = 0FFH: Data is present in console input buffer.		
Explanation			

CONST checks the CON: field (bits 0 and 1) of the I/O byte (at address 3) to determine whether the console input device is the keyboard or RS-232C interface and returns the status of the console.

```
CON: Bit 1     Bit 0
      0          0    ⎫  Indicates whether the keyboard buffer
      0          1    ⎬  is empty.
      1          0    ⎫  Indicates whether the RS-232C receive
      1          1    ⎬  buffer is empty.
```

Entry Name	CONIN	Entry address	WBOOT + Ø6H
Function	Returns one character read from the console.		
Entry parameter	None.		
Return parameter	When YPFCMFLG ≒ ØFFH A = ASCII code When YPFCMFLG = ØFFH C = ØØH --> A = ASCII code C = ØFFH --> A reg. contains one of ØEØH through ØE9H which correspond to PF1 through PF9.		
Explanation			

CONIN checks the CON: field of the I/O byte like CONST, and

receives one character from the keyboard or RS-232C interface.

This routine waits until a character is received.

(1) When the keyboard is assigned to the console (I/O byte,

     bits 1 and 0 are 00 or 01)

CONIN operates in different ways depending on the state of

YPFCMFLG (at 0Fl08H) which controls the handling of the PF keys.

1)   When YPFCMFLG ≒ FFH

      When a PF key is pressed, CONIN returns the string defined

for that PF key. Consequently, CONIN cannot determine what PF key is pressed. When a key other than PF keys is pressed, CONIN returns the corresponding ASCII code.

2)   When YPFCMFLG = ØFFH

CONIN returns via the C reg. the information as to whether a PF key is pressed.

- When C = ØØH

Indicates that a key other than PF keys is pressed and the corresponding ASCII code is placed in the A reg.

- When C = ØFFH

Indicates that a PF key is pressed. The A reg. contains either one of EØH through E9H which correspond to PF1 through PF9.

YPFCMFLG is set by directly rewriting the work area or by writing ESC + ØBØH or ESC + ØB1H through the CONOUT routine.
CONIN waits until input data is received. When the auto power off time expires, however, power is automatically turned off during the CONIN routine in the continue mode. When power is turned on again, execution resumes at the CONIN wait state.

(2) When the RS-232C interface is assigned to the console
     (I/O byte, bits 1 and 0 are 10 or 11)

CONIN places the data received from the RS-232C interface into the A reg. When no data is present at the RS-232C interface, CONIN waits until data is received. The operation of this routine is identical to that of RSIN.

Entry Name	CONOUT	Entry Address	WBOOT + 09H
Function	Outputs one character to the console.		
Entry parameter	C = output data		
Return parameter			
Explanation			

See Chapter 6 for details.

Entry Name	LIST	Entry Address	WBOOT + 0CH
Function	Outputs one character to the list device.		
Entry parameter	C = output data		
Return parameter	None		
Explanation			

LIST checks the LST: field (bits 7 and 6) of the I/O byte and sends one character to the corresponding device.

I/O byte

Bit 7	Bit 6		
0	0	(TTY):	Outputs to the serial port.
0	1	(CRT):	Outputs to the LCD (LIST operates in the same way as CONOUT).
* 1	0	(LPT):	Outputs to the RS-232C interface (LIST operates in the same way as RSOUT). LIST waits until DSR and TxRDY are set to 1 indicating that the counterpart receiver is ready for reception.

1   1   (UL1): Does nothing.

*: Default setting.

When the I/O byte is set to serial or RS-232C interface and
LIST is used for the first time after WBOOT, LIST outputs
the command ESC + "R" + x to select the character set
corresponding to the country currently set before sending
the output data.

Entry Name	PUNCH	Entry Address	WBOOT + ØFH
Function	Outputs one character to the punching device.		
Entry parameter	C = output data		
Return parameter	None		
Explanation			

PUNCH checks the PUN: field (bits 5 and 4) of the I/O byte and sends one character to the corresponding device.

I/O byte

Bit 5	Bit 4		
Ø	Ø	(TTY):	Does nothing.
Ø	1	(PTP):	Outputs to the LCD (operates in the same way as CONOUT).
* 1	Ø	(UP1):	Outputs to the RS-232C interface (operates in the same way as RSOUT). PUNCH waits until DSR and TxRDY are set to 1 indicating that the counterpart receiver is ready for reception.

1   1   (UP2): Does nothing.

           *: Default setting.

Entry Name	READER	Entry Address	WBOOT + 12H
Function	Inputs one character from the reader device.		
Entry parameter	None.		
Return parameter	A = input data.		
Explanation			

READER checks the RDR: field (bits 3 and 2) of the I/O byte and reads one character from the corresponding device. When no input data is present, READER waits until data is received.

I/O byte

      Bit 3   Bit 2

        0   0   (TTY): Reads from the keyboard (operates in the same way as CONIN).

        0   1   (PTP): Does nothing.

        1   0   (UP1): Reads from the RS-232C interface (operates in the same way as RSIN).

        1   1   (UP2): Does nothing.

In OS ASCII versions B and later, READER always returns 1AH (EOF) when the PTR or UR2 is selected.

Entry Name	HOME	Entry Address	WBOOT + 15H
Function	Positions the disk head to track 00.		
Entry parameter	None.		
Return parameter	None.		
Explanation			

HOME writes the write data left in the FDD buffer into the floppy

disk and moves the disk head to track 00.

Entry Name	SELDSK	Entry Address	WBOOT + 18H
Function	Specifies the drive.		

Entry parameter	C = logical drive No. 00H = A: --> 08H = I:
	Bit 0 of the E reg. indicates whether the drive
	is to be accessed for the first time after WBOOT.
	  Bit 0 = 0: The first access after WBOOT.
	  Bit 0 = 1: Not the first access after WBOOT.		

Return parameter	HL = 0000H: Parameter error.
	HL ≒ 0000H: Normal termination.
	  HL contains the DPE (disk parameter header)
	  address of the physical drive corresponding
	  to the logical drive.		

Explanation			

Entry parameters 00H through 08H correspond to the logical drives
A: through I:, respectively. Since the correspondence between
the logical drives A: through G: and the actual physical drives is
not fixed, SELDSK specifies the drive after translating the
logical drive into the physical drive. (See "Changing Drives"
for details about logical and physical drives.)

SELDSK sets or resets bit 0 of the E reg. to indicate whether the drive is to be accessed for the first time. When bit 0 = 0, SELDKS takes the following actions according to the selected physical drive:

1. RAM DISK   (Default logical drive is A:.)

    Does nothing.

2. ROM capsule   (Default logical drives are B: and C:.)

1) Turns the ROM capsule power on.

2) Checks whether the 2 bytes of the ROM header contains 0E5H and 37H to determine whether ROM is actually installed and whether the ROM is for ROM capsules. A parameter error is signaled if an error occurs.

3. FDD   (Default logical drives are D:, E:, F:, and G:.)

1) Opens the serial port for communication and turns the drive power on.

2) If the write buffer has been already loaded with write data, SELDSK writes the data onto the FD.

    Example: ON the TF-20 which contains two drives, if drive E: is specified when the preceding write data for drive D: is only placed in the buffer but not actually written on the FD, SELDSK flushes out the buffer before designating drive E:.

3) Otherwise, SELDKS issues the RESET command to the FDD.

    Once the FDD buffer is cleared through operation 2) or 3), the

FDD can be used with the newly specified drive designation.

A parameter error will be reported if an error occurs during

the above processing; e.g., the serial port cannot be opened

or the RESET command is terminated abnormally (no FDD is

installed or no floppy disk is inserted).

4. Microcassette drive (Default logical drive is H:.)

Does nothing.

5. ROM capsule in the extended unit (Default logical drive is

I:.)

1) Checks whether the extended unit is installed.

2) Checks whether ROM is installed in the ROM capsule in the

extended unit and whether the 2 bytes of the header are 0E5H

and 37H which identify the ROM for ROM capsules.

A parameter error will be signaled if an error occurs during

operation 1) or 2).

Entry Name	SETTRK	Entry Address	WBOOT + 1BH
Function	Specifies the track for read or write.		
Entry parameter	BC = track No.		
Return parameter	None.		
Explanation			

The following track numbers can be specified depending on the drive type:

Physical drive	Logical drive	Track No.
RAM DISK	A:	0 - 2: Internal RAM disk
		0 - 7: 60K RAM disk unit
		0 - 7: 64K RAM disk unit
		0 - 14: 120K RAM disk unit
		0 - 15: 128K RAM disk unit
ROM capsule	B:	0 - 7
	C:	
FDD	D:	0 - 39
	E:	

	F:	
	G:	
MCT	H:	0 - 4
ROM capsule in extended unit	I:	0 - 15

Since SETTRK makes no entry parameter check, it reports no error even if a truck number outside the valid range is specified. An error will be reported when an actual read or write operation is performed.

Entry Name	SETSEC	Entry Address	WBOOT + 1EH
Function	Specifies the sector for subsequent read or write.		
Entry parameter	BC = sector No. (0 - 63)		
Return parameter	None.		
Explanation			

Valid sector numbers are 0 through 63. Although SETSEC does not check the entry parameter, an error will be signaled when an actual read or write is performed if a sector number beyond that range is specified.

Entry Name	SETDMA	Entry Address	WBOOT + 21H
Function	Specifies the DMA starting address for read or write.		
Entry parameter	BC = DMA starting address.		
Return parameter	None.		
Explanation			

SETDMA specifies the starting address of the area to be used as the memory buffer during read or write. Data is read from or written onto the drive in 128 byte (1 sector) units.

Entry Name	READ	Entry Address	WBOOT + 24H
Function	Reads the specified sector.		
Entry parameter	None.		
Return parameter	A = 00H: Normal termination. A ≠ 00H: Abnormal termination.		
Explanation			

READ reads the sector specified by SELDSK, SETTRK, and SETSEC and stores the contents in the 128 byte area starting at the address specified by SETDMA.

If the drive is FDD (D:, E:, F:, G:), one of the following codes is returned when an error occurred:

    FAH: Read error.

    FBH: Write error.   Only 0FAH or 0FCH is returned

    FCH: Select error.   by READ.

    FDH: Read only disk.

    FEH: Read only file.

An error will be generated if a READ is executed for MCT (H:).
Use MIOS subroutines for MCT.

Entry Name	WRITE	Entry Address	WBOOT + 27H
Function	Writes the data to the specified sector.		

Entry parameter	C = Specifies how to write.
	
	    ØØH: Write standard format data (write after blocking).
	
	    Ø1H: Write unblocked data (write immediately without blocking).
	
	    Ø2H: Write to a sequential file.		

Return parameter	A = ØØH: Normal termination. A ≠ ØØH: Abnormal termination.		
Explanation			

WRITE writes the data from the 128 byte area starting at the address specified by SETDMA into the sector specified by SETTRK and SETSEC.

If the drive is FDD (D:, E:, F:, G:), one of the following codes is returned when an error occurred:

FAH: Read error.
FBH: Write error.
FCH: Select error.
FDH: Read only disk.
FEH: Read only file.

Only ØFBH, ØFCH, ØFDH, or ØFEH is returned by WRITE.

An error will be generated if a WRITE is specified for a drive
other than RAM disk (A:) and FDD (D:, E:, F:, G:). Use MIOS
subroutines for MCT.

Entry Name	LISTST	Entry Address	WBOOT + 2AH
Function	Returns the status of the list device.		
Entry parameter	None.		
Return parameter	A = FFH: Ready (sending data on the list device is allowed). A = 00H: Busy (sending data on the list device is disallowed).		
Explanation			

LISTST checks the LST: field (bits 7 and 6) of the I/O byte and returns the status of the corresponding device.

I/O byte

Bit 7   Bit 6

Ø   Ø   (TTY): Checks the serial port.

ØFFH: Control In is high.

ØØH: Control In is low.

Ø   1   (CRT): Returns FFH because the device is always set to LCD.

1   Ø   (LPT): Checks the RS-232C interface.

ØFFH: DSR is high.

ØØH: DSR is low.

1   1   (UL1): Always returns ØFFH if no
actual device is defined.

Entry Name	SECTRN	Entry Address	WBOOT + 2DH
Function	Translates a logical sector to a physical sector.		
Entry parameter	BC = Logical sector number.		
Return parameter	HL = Physical sector number.		
Explanation			

Actually, SECTRN performs no actual translation but returns the physical sector number identical to the logical sector number. This function is originally provided to perform skew processing to increase FD performance. Therefore, physical to logical sector translation is not necessary for drives other than FDD.

For FDD, SECTRN need not translate sector numbers because the FDD connected to MAPLE is intelligent to perform logical to physical sector translation.

Entry Name	PSET	Entry Address	WBOOT + 30H
Function	Converts graphics screen data for display.		

Entry parameter	B = Data to be converted.
	
	C = Function.
	
	    01H: AND, 02H: OR, 03H: XOR
	
	    In other cases, PSET loads the C reg. with the data at the address specified by HL.
	
	HL = Graphics screen address of the data to be converted. (0 - 3839)		

Return parameter	A = 00H: Normal termination.
	
	  = FFH: Screen is in character mode.
	
	  = Others: HL contains an address other than graphics screen addresses (0 - 3839).
	
	C = Loaded with the operation result upon normal termination.		

Explanation			

PSET processes the 1 byte data at the address specified by HL and data in the B reg. on the graphics screen according to the data in the C reg., then places the result to the C reg. An error is reported in the following conditions:

- When the screen is not in graphics mode.

- When HL is loaded with an address other than the graphics

  screen addresses (0 - 3839).

PSET only loads the C reg. with the data at the specified address

on the graphics screen when the C reg. contains other than 01H,

02H, and 03H.

Each byte on the graphics screen is assigned an address as shown

below:
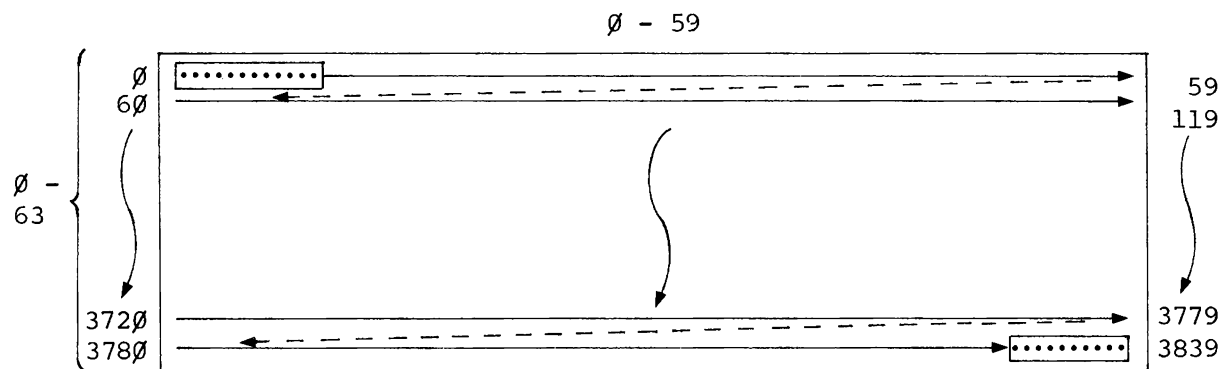
Entry Name	SCRNDUMP	Entry Address	WBOOT + 33H
Function	Takes a hard copy of the displayed data.		
Entry parameter	None.		
Return parameter	LSTERR (F69EH) = 00H: Normal termination. = 0FFH: Terminated with CTRL/STOP key.		
Explanation			

SCRNDUMP checks the I/O byte and dumps (outputs) the current data on the LCD screen onto the device (serial, RS-232C) specified in the LST: field. However, it does nothing if the LST: field is set to CRT (LCD).

The dump operation can be terminated any number of times by pressing the CTRL/STOP key. LSTERR indicates whether the operation was terminated with the CTRL/STOP key.

SCRNDUMP sends the display data to the serial port or RS-232C interface as characters when character mode is selected. It checks the sixth DIP switch and converts special codes to spaces

before output.

Sixth DIP switch

    0: Converts 00H - 1FH, 7FH, and 0FFH to spaces.

    1: Converts 00H - 1FH and 7FH - 0FFH to spaces.

The display data is output to the serial port or RS-232C interface in bit image when graphics mode is selected.

In either mode, seven lines from the top are output on the printer if the PF key definitions are displayed on line 8.

SCRNDUMP

```
┌─────────────┐
│ Check I/O   │
│ byte.       │
└─────────────┘

┌─────────────┐
│ Delete      │
│ cursor.     │
└─────────────┘

┌─────────────┐
│ Output CR   │
│ and LF.     │
└─────────────┘
```

Character screen   Graphics screen

```
                    ◇ Screen type. ◇
```

Character screen:

```
┌──────────────────┐
│ Read VRAM        │
│ contents in      │
│ character codes  │
└──────────────────┘

┌──────────────────┐
│ Convert special  │
│ codes to spaces  │
│ by referencing   │
│ DIP SW.          │
└──────────────────┘

┌──────────────────┐
│ Output on        │
│ printer.         │
└──────────────────┘
```

Repeat for lines 1 through 8 (lines 1 through 7 if the PF key definitions are displayed on line 8), one line at a time.

Graphics screen:

```
┌──────────────────┐
│ Output ESC       │         ⎤ Set line
│ + "A" + Ø8H      │         ⎥ spacing to
│ on printer.      │         ⎦ 8/72 inch.
└──────────────────┘

┌──────────────────┐
│ Output ESC       │         ⎤ Print a line
│ + "K" + ØEØH     │         ⎥ in 48Ø-dot bit
│ + Ø1H on         │         ⎦ image
│ printer.         │
└──────────────────┘

┌──────────────────┐
│ Read VRAM        │
│ contents in      │
│ bit image.       │
└──────────────────┘

┌──────────────────┐
│ Output on        │
│ printer.         │
└──────────────────┘

┌──────────────────┐
│ Output ESC       │         Set line
│ + "2" on         │         specing to
│ printer.         │         1/6 inch.
└──────────────────┘
```

```
┌──────────────────┐
│ Load ØØH         │
│ into LSTERR.     │
└──────────────────┘

┌──────────────────┐
│ Display          │
│ cursor.          │
└──────────────────┘
```

EXIT

Throughout the processing, SCRNDUMP checks whether the CTRL/STOP key is pressed. If pressed, it immediately terminates the operation with LSTERR loaded with FFH.

Entry Name	BEEP	Entry Address	WBOOT + 36H
Function	Sounds the speaker.		
Entry parameter	C = Specifies the duration of a beep in 100 ms units. BEEP does nothing if C = 0. DE = Specifies the period in 3.2 us units. $$\text{Frequency} = \frac{1}{3.2 \times (DE)} \times 106 \text{ Hz}$$		
Return parameter	None.		
Explanation			

BEEP generates a beep sound in the period specified by DE with the duration of time specified by C.

The processing can be terminated any number of times by pressing the CTRL/STOP key or turning the power switch off.

BEEP can be used as a 100 ms software timer because it waits for the length of time specified by the C reg. without generating sound if DE = 0000H.

Entry Name	RSOPEN	Entry Address	WBOOT + 39H
Function	Opens the RS-232C interface.		
Entry parameter	None.		
Return parameter	A = 00H: Normal termination. = 02H: Already open. = 04H: An invalid specification was found in the conditions set by CONFIG for RS-232C. This error causes no problem as long as CONFIG specifies the conditions for RS-232C but may cause a problem if the work area has been updated directly by the application program.		
Explanation			

RSOPEN initializes the RS-232C interface based on the conditions set by CONFIG, turns RS-232C power on, enables RS-232C receive interrupts (8251 interrupts) for RS-232C communication.

RSOPEN must be executed before executing the following routines:

RSIN

RSINST

RSOUTST

RSOUT

Entry Name	RSCLOSE	Entry Address	WBOOT + 3CH
Function	Closes the RS-232C interface.		
Entry parameter	None.		
Return parameter	None.		
Explanation			

RSCLOSE turns RS-232C power off and disables RS-232C receive interrupts.

Entry Name	RSINST	Entry Address	WBOOT + 3FH
Function	Informs whether the RS-232C interface has received data.		
Entry parameter	None.		
Return parameter	See below.		
Explanation			

The status at termination is as follows:

1) Z flag = 1: Normal termination.

    A = FFH: Received data present.

    A = ØØH: No received data present.

    BC = Number of received data bytes in the buffer.

2) Z flag = ØØH: Abnormal termination.

    A = Ø3H: RS-232C is not open.

Entry Name	RSOUTST	Entry Address	WBOOT + 42H
Function	Checks whether the RS-232C interface is ready for transmission.		
Entry parameter	None.		

Return parameter	A = 00H: Transmission disabled. (Z flag = 1)
	  = FFH: Transmission enabled. (Z flag = 1)
	  = 03H: RS-232C is not open. (Z flag = 0)		

Explanation			

The RS-232C interface is enabled for transmission when the following two conditions are met:

1) 8251 TxRDY = 1.

    (For Overseas Version 1.0, TxEMPTY must also be set to 1.)

2) No XOFF is received when XON/XOFF control is specified.

Entry Name	RSIN	Entry Address	WBOOT + 45H
Function	Receives one character from RS-232C.		
Entry parameter	None.		

Return parameter	Z flag = 1: Normal termination.
	  A = Received data.
	Z flag = Ø: Abnormal termination.
	  A = Ø3H: RS-232C is not open.
	  A = Ø4H: CTRL/STOP key is pressed.		

Explanation

When no data is present at the RS-232C interface, RSIN waits until data is received. Processing can be terminated by pressing CTRL/STOP key.

If XON/XOFF control is specified, RSIN sends an XON when the number of the received bytes in the buffer has reduced down to 1/4 of the buffer capacity after it sent an XOFF.
When SI/SO is specified, RSIN performs SI/SO processing on the received data.

As explained above, XON/XOFF and SI/SO codes are processed by the operating system and not returned to the application program as data bytes.

Entry Name	RSOUT	Entry Address	WBOOT + 48H
Function	Transfers one character to RS-232C.		
Entry parameter	None.		
Return parameter	Z flag = 1: Normal termination. Z flag = 0: Abnormal termination. A = 03H: RS-232C is not open. A = 04H: CTRL/STOP key was pressed.		
Explanation			

RSOUT checks whether the RS-232C interface is enabled for output (conditions are the same as with RSOUTST) and, if it is disabled, waits until the interface is ready for transmission. Processing can be terminated by pressing the CTRL/STOP key.

RSOUT sends an SI or SO code before sending the pertinent data byte if SI/SO control is specified.

Entry Name	TIMDAT	Entry Address	WBOOT + 4BH
Function	Performs clock and alarm functions.		
Entry parameter	Described below.		
Return parameter	Described below.		
Explanation			

TIMDAT provides the following six functions:

1. Reads the time. (C = 00H)

2. Sets the time. (C = 0FFH)

3. Enables the alarm/wake function. (C = 80H)

4. Disables the alarm/wake function. (C = 81H)

5. Sets the alarm/wake time. (C = 82H)

6. Reads the alarm/wake time. (C = 84H)

The calling program must call TIMDAT after loading the C reg. with the code of the function to be performed and the D reg. with the starting address of the packet (time descriptor) for transferring time-related data. TIMDAT will do nothing if the C reg. is loaded with a code other than the above codes.

TIMDAT assumes the following clock specifications:

- Maximum time count is 23:59:59 12/31/1999.

- Leap year processing is performed automatically.

- The time is represented in the 24-hour system.

- The day of the week is not set automatically but updated when the day changes.

Time descriptor structure

The time descriptor consists of 11 bytes as shown below. Not all bytes are necessarily used by a function.

```
(DE)----------→
```

①	Loaded with the lowest two digits of the year in BCD code.	1 byte
②	Loaded with the month in BCD code.	1 byte
③	Loaded with the day in BCD code.	1 byte
④	Loaded with the hour in BCD code.	1 byte
⑤	Loaded with the minute in BCD code.	1 byte
⑥	Loaded with the second in BCD code.	1 byte
⑦	Loaded with the day of the week.	1 byte
⑧	Loaded with the alarm/wake type.	1 byte
⑨	Loaded with the address.	2 bytes
⑩	Loaded with the status.	1 byte

(1) - (6): Year, month, day, hour, minute, second

The time data 1984, 09, 14, 15, 53, 28 is loaded as follows:

84H, 09H, 14H, 15H, 53H, 28H

(1)　(2)　(3)　(4)　(5)　(6)

(7): Day of the week

00H, 01H, 02H, 03H, 04H, 05H, 06H

SUN. MON. TUE. WED. THU. FRI. SAT.

(8): Type

Specifies the alarm/wake type.

00H --- No specification.

01H --- Sets the alarm.

(Displays an alarm message at the specified time.)

02H --- Specifies wake1.

(Performs the function identified by the string at the address specified in (9) at the specified time.)

03H --- Specifies wake2.

(Executes the subroutine at the address specified in (9) at the specified time.)

(9): Address

The meaning of the address differs depending on the type
specified in (8).

Type   Meaning

01H ----- Starting address of the alarm message.

02H ----- Starting address of the string identifying the
            function to be executed during wake1.

03H ----- Starting address of the subroutine (processing) to
            be executed during wake2.

The alarm message and wake1 string must be defined in the
following format:

```
                    41 bytes in total
|<----------------------------------------------------->|
 _____
|   |                                                    |
|   |              Message or string                     |
|___|_____|
  ^ |
  | |<------------------------------------------------->|
  |                    40 bytes
  |
  |_ Message length (1 byte):
```
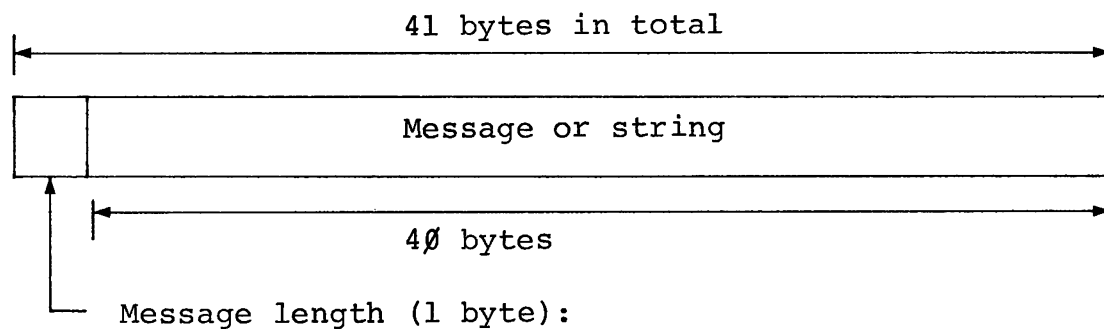
Specify the actual message text or string length in
binary from 00H to 28H. 00H indicates no message or
null string.

(10): Status

Identifies the alarm/wake interrupt type.

Interrupt type   Status value

Alarm/wake time is specified. ------- 00H

(via BIOS TIMDAT).

Alarm/wake interrupt is generated. -- 01H

Alarm/wake time is read ------------- Set to 00H after

(via BIOS TIMDAT).   the current status

                                         is returned.

TIMDAT returns 01H only when it has read an alarm/wake time
for the first time after an alarm/wake interrupt occurs.
TIMDAT continues to return 00H whenever called until the
next interrupt occurs.

Entry Name	TIMDAT (1)	Entry Address	WBOOT + 4BH
Function	Reads the time.		
Entry parameter	C = 00H DE = Time descriptor starting address.		
Return parameter	DE = Time Descriptor starting address.		
Explanation			

TIMDAT (1) loads the time descriptor fields (1) to (7) with the year, month, day, hour, minute, second, and day of the week to set the clock.

Entry Name	TIMDAT (2)	Entry Address	WBOOT + 4BH
Function	Sets the time.		
Entry parameter	C = ØFFH DE = Time descriptor starting address.		
Return parameter	DE = Time descriptor starting address.		
Explanation			

TIMDAT (2) loads the time descriptor fields (1) to (7) with the year, month, day, hour, minute, second, and day of the week that are read from the clock. The BCD digits which are loaded with ØFH codes retain the previous time settings.

Since TIMDAT (2) makes no check, the validity of the subsequent information supplied by the clock is not guaranteed if logically invalid data is specified in this function.

Entry Name	TIMDAT (3)	Entry Address	WBOOT + 4BH
Function	Enables an alarm/wake function.		
Entry parameter	C = 80H		
Return parameter	None.		
Explanation			

No alarm/wake interrupt will be generated even when an alarm/wake time is specified until the alarm/wake function is enabled by TIMDAT (3).

Entry Name	TIMDAT (4)	Entry Address	WBOOT + 4BH
Function	Disables an alarm/wake function.		
Entry parameter	C = 81H		
Return parameter	None.		
Explanation			

No alarm/wake interrupt occurs once TIMDAT (4) is executed.

To use the alarm/wake function again, it is necessary to redefine alarm/wake time using the following steps:

1) Specify the alarm/wake time.

2) Enable the alarm/wake time.

Entry Name	TIMDAT (5)	Entry Address	WBOOT + 4BH
Function	Specifies the alarm/wake time.		
Entry parameter	C = 82H DE = Time descriptor starting address.		
Return parameter	DE = Time descriptor starting address.		
Explanation			

Call TIMDAT (5) after filling the month to address fields (entries (2) - (9)) in the time descriptor .

The year cannot be specified for the alarm/wake function. The value in the unit place in the second field (the lowest 4 bits of (6)) is also ignored because TIMDAT (5) monitors only the value in the ten's place.

Any BCD digits which are set to 0FH (four bits are all set to 1) in the entries from the month to the day of the week are regarded as matching any time value. For example, alarm/wake will be invoked at the specified time every day if the month and day are set to 0FFH.

Since TIMDAT (5) makes no entry data check, normal clock

operation cannot be guaranteed if invalid data is specified.

No alarm/wake interrupt will be generated even when an alarm/wake

time is specified until the alarm/wake function is enabled by

TIMDAT (5).

Entry Name	TIMDAT (6)	Entry Address	WBOOT + 4BH
Function	Reads the alarm/wake time.		
Entry parameter	C = 84H DE = Time descriptor starting address.		
Return parameter	DE = Time descriptor starting address.		
Explanation			

The current alarm/wake settings are loaded into the year to status fields of the time descriptor ((1) - (1Ø)) after TIMDAT (6) is executed. The year field and the first digit of the second field are always set to ØFFH and ØFH, respectively. This is because they are never set by TIMDAT (5).

The validity of the data loaded into the time descriptor is not guaranteed if TIMDAT (6) is executed with no alarm/wake information specified.

Entry Name	MEMORY	Entry Address	WBOOT + 4EH
Function	Does nothing.		
Entry parameter	None.		
Return parameter	None.		
Explanation			

Entry Name	RSIOX	Entry Address	WBOOT + 51H
Function	Performs various RS-232C functions.		
Entry parameter	Described below.		
Return parameter	Described below.		
Explanation			

RSIOX provides the following ten functions which are identified by the contents of the B reg.:

1. Opens RS-232C. (B = 10H)

2. Closes RS-232C. (B = 20H)

3. Informs whether RS-232C has received data. (B = 30H)

4. Checks whether RS-232C is enabled for transmission. (B = 40H)

5. Receives one character from RS-232C. (B = 50H)

6. Sends one character from RS-232C. (B = 60H)

7. Checks the control line status. (B = 70H)

8. Sets the control line. (B = 80)

9. Checks the error status. (B = 90H)

10. Checks whether RS-232C is open. (B = 0F0H)

Entry Name	RSIOX (OPEN)	Entry Address	WBOOT + 51H
Function	Opens the RS-232C interface.		
Entry parameter	B = 10H HL = Parameter block starting address.		
Return parameter	A = 00: Normal termination. (Z flag = 1) = 02H: Already open. (Z flag = 0) = 03H: Invalid parameter. (Z flag = 0) HL = Parameter block starting address. (The parameter block is loaded with status data.)		
Explanation			

RSIOX (OPEN) initializes the RS-232C interface based on the conditions set in the specified parameter block, turns RS-232C power on, enables the RS-232C controller (8251) for receive interrupts) to ready the interface for communication.

RSIOX (OPEN) has the same functions as RSOPEN (WBOOT + 39H) except that it allows the user to initialize the RS-232C interface.

The calling program must always call RSIOX before performing I/O operations to or from the RS-232C interface.

Parameter block structure

```
(HL) -- -- --->  ┌─────────────────────────────────────┐
              1  │ Receive Buffer Starting Address     │   2 bytes
                 ├─────────────────────────────────────┤
              2  │ Receive Buffer Length               │   2 bytes
                 ├─────────────────────────────────────┤
              3  │ Baud Rate                           │   1 byte
                 ├─────────────────────────────────────┤
              4  │ Bits/Char                           │   1 byte
                 ├─────────────────────────────────────┤
              5  │ Parity                              │   1 byte
                 ├─────────────────────────────────────┤
              6  │ Stop Bits                           │   1 byte
                 ├─────────────────────────────────────┤
              7  │ Special Parameter                   │   1 byte
                 └─────────────────────────────────────┘
```

(1)   Receive Buffer Starting Address

Specifies the starting address of the receive buffer. The

buffer may be located anywhere in the CP/M TPA.

(2)   Receive Buffer Length

Specifies the length of the receive buffer.

(3)   Bit Rate

Specifies the bit rate. The table below lists the codes

that correspond to the available bit rates.

Code	Bit Rate (BPS)
ØFH	19200
ØEH	9600
ØDH	4800
ØCH	2400
ØAH	1200
Ø8H	600
Ø6H	300
Ø5H	200
Ø4H	150
Ø2H	110
81H	75/1200 (Tx/Rx)
8ØH	1200/75 (Tx/Rx)

Ø5H, Ø4H, Ø2H ◄── Not supported in the overseas versions.

Tx and Rx represent the transmit and receive bit

rates, respectively. Tx and Rx may be different.

4-63

(4) Bit/Char

Specifies the character length in bits.

02H --- 7 bits/character

03H --- 8 bits/character

(5) Parity

Specifies parity check type.

00H --- No parity

01H --- Odd

03H --- Even

(6) Stop Bits

Specifies the number of stop bits.

01H --- 1 bit

03H --- 2 bits

(7) Special Parameter

Specifies the RS-232C operating modes and status on a bit basis.

Bit	Description
0	Controls the DTR line. 0: OFF (-8V) 1: ON (+8V)
1	Controls the RTS line. 0: OFF (-8V) 1: ON (+8V)
2	Specifies whether SI/SO is to be controlled. 0: Controlled. (Valid only for 7 bits/char. data width) 1: Not controlled.
3	Not used.
4	Specifies whether XON/XOFF control is to be used. 0: Controlled. 1: Not controlled.
5 - 7	Not used.

This byte must be set to 0FFH when not used.

Parameter block contents on return

On return, the HL reg. retains the starting address of the parameter block that was specified on entry. The contents of the parameter block are changed as follows:

(HL) --->		
1	Status	1 byte
2	GET Point	2 bytes
3	PUT Point	2 bytes
4	Receive Buffer Starting Address	2 bytes
5	Receive Buffer Length	2 bytes

(1) Status

Indicates the RS-232C status.

Bit	Description
Ø	Indicates whether RS-232C is open. Ø: Open. 1: Not open.
1	Indicates whether the receive buffer is full. Ø: Not full. 1: Full.
2	Indicates whether a receive buffer overflow occurred. Ø: No overflow occurred. 1: Overflow occurred. Some data must have been discarded.
3	Indicates the CD line status (inverted). Ø: CD line is high. (+3 ~ +15V) 1: CD line is low. (-3 ~ -15V)
4	Indicates whether a parity error occurred. Ø: No parity error occurred. 1: Parity error occurred.
5	Indicates whether an overrun error occurred in 8251 during data reception. Ø: No overrun error occurred. 1: Overrun error occurred.

	Overrun errors are likely to occur when data transfer is too fast.

6	Indicates whether a framing error occurred during data reception.
	
	    0: No framing error occurred.
	    1: Framing error occurred.
	
	Framing errors occur when the parameters of the RS-232C (bit rate, bits/char, parity, stop bits) do not match those of the counterpart terminal.

7	Indicates the DSR line status.
	
	    0: DSR line is high. (+3 ~ +15V)
	    1: DSR line is low. (-3 ~ -15V)

Bits 0, 1, 3, and 7 always indicate the current status. Bits 2, 4, 5, and 6, on the other hand, retains the error status until the RSIOX error check function is executed once an error occurred.

(2) GET Point

The address of the next data to be taken from the receive buffer.

(3) PUT point

The receive buffer address into which the next data received by 8251 is to be placed.

(4) Receive Buffer Starting Address

The address specified on entry.

(5) Receive Buffer Length

The length specified on entry.

Entry Name	RSIOX (CLOSE)	Entry Address	WBOOT + 51H
Function	Closes the RS-232C interface.		
Entry parameter	B = 20H		
Return parameter	None.		
Explanation			

RSIOX (CLOSE) turns RS-232C power off and disables RS-232C receive interrupts. The functions of RSIOX (CLOSE) is identical to those of RSCLOSE (WBOOT + 3CH).

Entry Name	RSIOX (INSTS)	Entry Address	WBOOT + 51H
Function	Indicates whether there is any data in the receive buffer.		
Entry parameter	B = 30H HL = Starting address of the field for storing 9-byte return information.		
Return parameter	Described below.		
Explanation			

The status information that RSIOX (INSTS) returns on termination is as follows:

(1) Z flag = 1: Normal termination.

      A = 0FFH: Data has been received.

        = 00H: No data in the receive buffer.

    BC = Number of bytes of received data in the buffer.

    HL = Address specified on entry. The nine bytes starting at this address contains the return information described earlier (see RSIOX (OPEN)).

(2) Z flag = 0: Abnormal termination.

    A = 03H: RS-232C is not open.

    HL retains the previous value.

Entry Name	RSIOX (OUTST)	Entry Address	WBOOT + 51H
Function	Checks whether RS-232C is enabled for transmission.		
Entry parameter	B = 40H HL = Starting address of the field for storing 9-byte return information.		
Return parameter	Described below.		
Explanation			

The status information RSIOX (OUTST) returns on termination is as follows:

(1) Z flag = 1: Normal termination.

    A = 00H: Transmission disabled.

     = 0FFH: Transmission enabled.

    HL = The address specified on entry. The nine bytes starting at this address contains the return information described earlier (see RSIOX (OPEN)).

The RS-232C interface is enabled for transmission when the following two conditions are satisfied:

1) 8251 TxRDY = 1

(For Overseas Version 1.0, TxEMPTY must also be set to 1.)

2) No XOFF is received when XON/XOFF control is specified.

(2) Z flag = 0: Abnormal termination.

A = 03H: RS-232C is not open.

HL retains the previous value.

Entry Name	RSIOX (GET)	Entry Address	WBOOT + 51H
Function	Receives one character from the RS-232C interface.		
Entry parameter	B = 50H HL = Starting address of the field for storing 9-byte return information.		
Return parameter	Described below.		
Explanation			

RSIOX (GET) returns the following status on termination:

(1) Z flag = 1: Normal termination.

    A = Received data.

    HL = The address specified on entry. The nine bytes

        starting at this address contains the return

        information described earlier (see RSIOX (OPEN)).

(2) Z flag = 0: Abnormal termination.

    A = 03H: RS-232C is not open.

      = 04H: CTRL/STOP key is pressed.

    HL retains the previous value.

The actual function of RSIOX (GET) is identical to that of RSIN

    (WBOOT + 45H).

Entry Name	RSIOX (PUT)	Entry Address	WBOOT + 51H
Function	Transfers one character to the RS-232C interface.		
Entry parameter	B = 60H C = Send data HL = Starting address of the field for storing 9-byte return information.		
Return parameter	Described below.		
Explanation			

RSIOX (PUT) returns the following status on termination:

(1) Z flag = 1: Normal termination.

HL = The address specified on entry. The nine bytes starting at this address contains the return information described earlier (see RSIOX (OPEN)).

(2) Z flag = 0: Abnormal termination.

A = 03H: RS-232C is not open.

= 04H: CTRL/STOP key is pressed.

HL retains the previous value.

The actual functions of RSIOX (PUT) is identical to those of RSOUT (WBOOT + 48H).

Entry Name	RSIOX (CTLIN)	Entry Address	WBOOT + 51H
Function	Reads the control line status.		
Entry parameter	B = 70H		
Return parameter	Described below.		
Explanation			

RXIOX (CTLIN) returns the DSR and CD status when the RS-232C is open.

(1) Z flag = 1: Normal termination

      A reg. Bit 7 = DSR status.

                0: +3V to +8V

                1: Lower than +3V

          Bit 3 = CD status.

                0: Lower than +3V

                1: +3V to +8V

        All bits other than bits 7 and 3 are set to 0.

(2) Z flag = 0: Abnormal termination

      A = 03H: RS-232C is not open.

Entry Name	RSIOX (SETCTL)	Entry Address	WBOOT + 51H
Function	Sets control lines.		
Entry parameter	B = 80H C = Data set (see below).		
Return parameter	Z flag = 1: Normal termination. Z flag = 0: Abnormal termination. A = 03H: RS-232C is not open.		
Explanation			

RSIOX (SETCTL) sets the DTR and/or RTS line states according to the contents of the C reg.

    C reg. Bit 0: Sets the DTR state.

           = 0: DTR set to - 8V (Low)

           = 1: DTR set to + 8V (High)

      Bit 1: Sets the RTS state.

           = 0: RTS set to - 8V (Low)

           = 1: RTS set to + 8V (High)

      Bits 2 - 7: Not used.

Entry Name	RSIOX (ERSTS)	Entry Address	WBOOT + 51H
Function	Checks the RS-232C error status.		
Entry parameter	B = 90H		
Return parameter	Described below.		
Explanation			

RSIOX (ERSTS) returns the error status of the RS-232C interface when it is open. All errors are cleared on termination of RSIOX (ERSTS).

(1) Z flag = 1: Normal termination.

   A = Error status
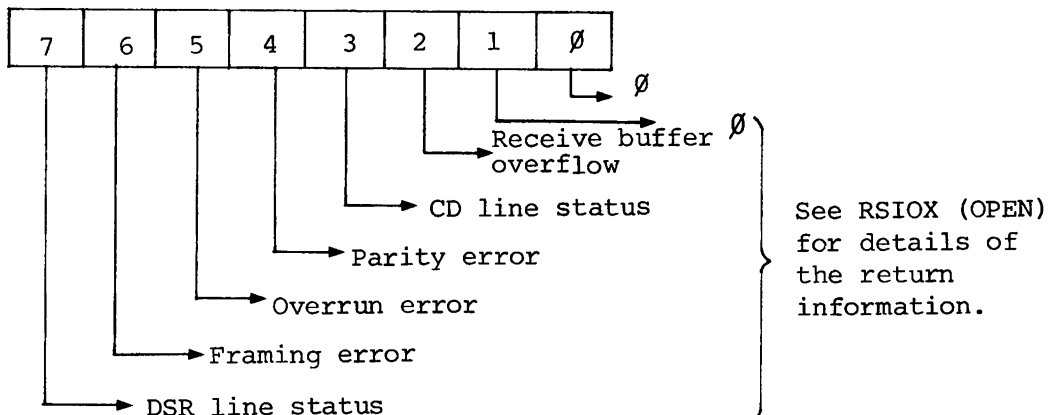
(2) Z flag = 0: Abnormal termination.

   A = 03H: RS-232C is not open.

Entry Name	RSIOX (SENS)	Entry Address	WBOOT + 51H
Function	Checks whether the RS-232C interface is open.		
Entry parameter	B = ØFØH		

Return parameter	Z flag = 1: RS-232C is not open.
	
	   A = ØØH
	
	Z flag = Ø: RS-232C is open.
	
	   A = Ø2H		

Explanation			

Entry Name	LIGHTPEN	Entry Address	WBOOT + 54H
Function	Does nothing.		
Entry parameter	None.		
Return parameter	None.		
Explanation			

Entry Name	MASKI	Entry Address	WBOOT + 57H
Function	Sets or resets interrupt mask.		
Entry parameter	Described below.		
Return parameter	Described below.		
Explanation			

MASKI enables or disables the six interrupts supported by MAPLE.

(1) Entry parameters

B = Function

= 0: Inhibits interrupts from the devices whose corresponding bit in the C reg. is 1.

= 1: Enables interrupts from the devices whose corresponding bit in the C reg. is 1.

$\geq$ 2: Checks the current enabled or disabled status.

C = Specifies which type of interrupts are to be processed according to the contents in the B reg.

Bit 0: 7508 interrupts

Bit 1: RS-232C (8251) receive interrupts

Bit 2: RS-232C Carrier Detect interrupts

Bit 3: FRC overflow interrupts

Bit 4: Bar code reader interrupts

Bit 5: External interrupts

Bit 6: Not used.

Bit 7: Not used.

The Interrupts for which the corresponding bits are set to 1 are processed according to the specification in the B reg. The interrupts for which the corresponding bit is set to 0 retain their previous state. Bits 6 and 7 must be set to 0.

(2) Return parameter

A = Loaded with return information indicating whether the individual interrupts are enabled after this function is executed. The correspondence between the bits and interrupt types is the same as that shown above. Interrupts are enabled if the corresponding bit is set to 1 and disabled if it is set to 0.

See Chapter 10 for details on individual interrupts.

Entry Name	LOADX	Entry Address	WBOOT + 5AH
Function	Reads one byte of data from the specified bank.		

Entry parameter	C = Bank from which data is to be read.
	
	   00H = User bank
	
	   0FFH = System bank
	
	HL = Address of the data to be read.		

Return parameter	A = Data Other registers retain the previous values.		
Explanation			

LOADX is used in application programs to read the contents of OS ROM. The user bank is selected when a value other than 00H and 0FFH is specified in the C reg.

Entry Name	STORX	Entry Address	WBOOT + 5DH
Function	Writes one byte of data to the specified bank.		

Entry parameter	A = Data to be written.
	
	C = Bank to which data is to be written.
	
	    00H = User bank
	
	    0FFH = System bank
	
	HL = Address at which data is to be written.		

Return parameter	All registers retain the previous values.		
Explanation			

STORX is not used in application programs. Nothing will happen
if it is used to write data into the system bank ROM.

The user bank is selected when a value other than 00H and 0FFH is
specified in the C reg.

Entry Name	LDIRX	Entry Address	WBOOT + 60H
Function	Transfers the data on the specified bank onto another bank.		
Entry parameter	A = 00H: Transfers data from the system to user bank. = 0FFH: Transfers data from the user to system bank. HL = Starting address of the data to be transferred. DE = Starting address of the destination to which data is to be transferred. BC = Number of bytes of data to be transferred.		
Return parameter	A = 00H BC = 0000H DE = DE + BC HL = HL + BC } Register contents on termination.		
Explanation			

LDIRX is used in application programs to transfer the contents of OS ROM to RAM. Specifying a value other than 00H and 0FFH in the A reg. causes the same effect as specifying 00H.

Entry Name	JUMPX	Entry Address	WBOOT + 63H
Function	Jumps to the specified bank address.		
Entry parameter	(DISBNK) = 00H: Jumps to the specified address on the user bank. = 0FFH: Jumps to the specified address on the system bank. IX = Destination of jump.		
Return parameter	None.		
Explanation			

JUMPX causes program execution to jump to an address in OS ROM.
JUMPX is rarely used in application programs.

This BIOS call is also terminated when a RET statement is
encountered in the routine at the jump address. Since control
branches with the stack in the BIOS, an error may occur if the
stack level goes too deep during the execution of the called
routine.

The DISBNK address is:

0F539H --- for Overseas Version OS

0F2B6H --- for Japanese Version OS

Specifying a value other than 00H and 0FFH in DISBNK has the
same effect as specifying 00H.

Entry Name	CALLX	Entry Address	WBOOT + 66H
Function	Calls the specified bank address.		
Entry parameter	(DISBNK) = 00H: Calls the specified address on the user bank. = 0FFH: Calls the specified address on the system bank. IX = Called routine address		
Return parameter	None.		
Explanation			

CALLX is used by application programs to directly call a routine in OS ROM.

Since the routine is called with the stack for BIOS still, unexpected results may occur if the called subroutine uses too large an amount of stack area.

The DISBNK address is:

        0F539H --- For Overseas Version OS

        0F2B6H --- For Japanese Version OS

Specifying a code other than 00H and 0FFH in DISBNK has the same effect as specifying 00H.

Entry Name	GETPFK	Entry Address	WBOOT + 69H
Function	Reads in PF key data.		
Entry parameter	C = PF key number - 1 PF1 = 00H --- PF10 = 09H HL = Starting address of the character string to be read.		
Return parameter	HL = Retains the previous value.		
Explanation			

GETPFK gets a character string defined for a PF key in 16-byte format as shown below. GETPFK does nothing when a value other than 00H to 09H is specified in the C reg.

(HL)

Number of characters	Character 1	Character 2	Character 3	Character 4	---------	Character 14	Character 15

15 bytes
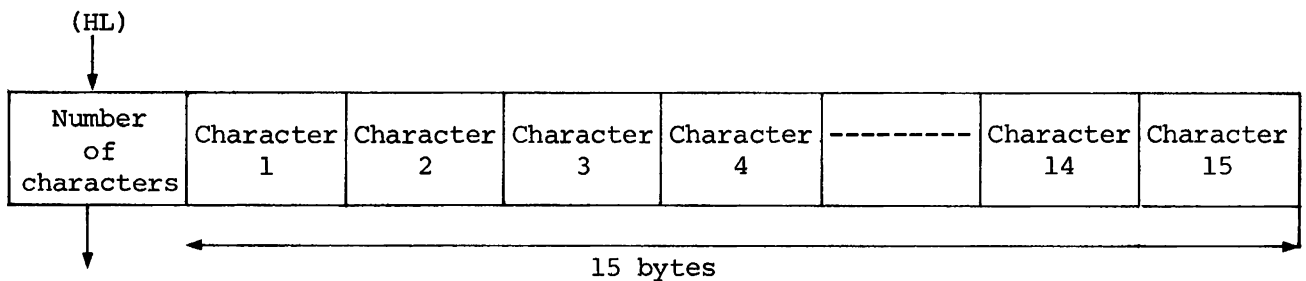
ØØH - ØFH:
    Indicates the number of characters in the string. ØØH indicates that no string is defined for this PF key.

    Example

Ø3H	"P"	"I"	"P"				----------		
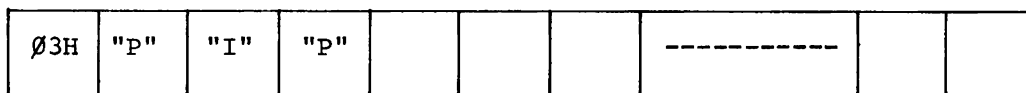
The contents of the subsequent bytes are not guaranteed.

Entry Name	PUTPFK	Entry Address	WBOOT + 6CH
Function	Defines a PF key.		

Entry parameter	C = PF key number - 1
	
	     PF1 = 00H --- PF10 = 09H
	
	HL = Starting address of the character string to
	     be assigned.		

Return parameter	HL = Retains the previous value.		
Explanation			

PUTPFK assigns a character string to a PF key in the 16-byte

format. The maximum string length is 15 characters. PUTPFK does

nothing when a value other than 00H to 09H is specified in the C

reg.
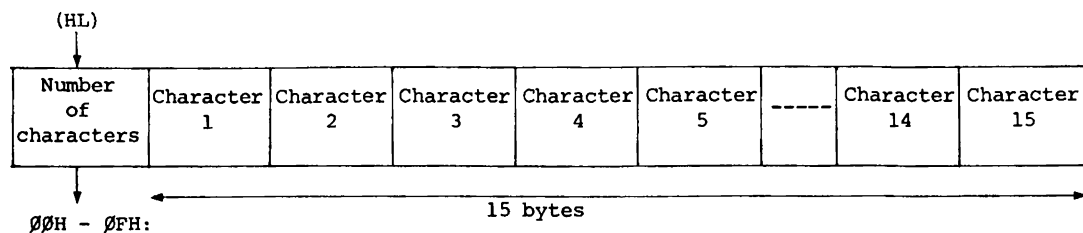
Specifies the number of defined characters in binary. 00H indicates
that no string is defined for the specified PF key.

If old PF key definitions are displayed on the screen, they are also updated as they are redefined by PUTPFK.

Entry Name	ADCVRT	Entry Address	WBOOT + 6FH
Function	Performs an analog data input operation.		
Entry parameter	C = Analog data to be selected.		
Return parameter	A = AD conversion results		
Explanation			

ADCVRT converts analog data selected by the parameter in the C reg. to digital data and returns the results to the A reg.

    C = 00H: A/D channel 1 --- Data from the analog jack.

    C = 01H: A/D channel 2 --- Data from the bar code reader
                                 connector.

    C = 02H: DIP SW settings.

    C = 03H: Battery voltage.

    C = 04H: Main switch (for Power ON/OFF) and analog input
                 connector trigger terminal settings.

ADCVRT does nothing when the C reg is loaded with a value other than 00H to 04H.

The pages that follow describe what data is returned to the A reg. according to the value specified in the C reg.

(1) When the C reg. = 00H or 01H

A voltage 0 to +2V applied to the A/D jack is converted
to a digital quantity and placed into the highest 6
bits of the A reg. (resolution of 6 bits).

```
        Bit                                    Bit
         7    6    5    4    3    2    1    Ø
       ┌────┬────┬────┬────┬────┬────┬────┬────┐
Areg.  │    │    │    │    │    │    │    │    │
       └────┴────┴────┴────┴────┴────┴────┴────┘
```

Ø ~ +2V   Not defined.

⇓
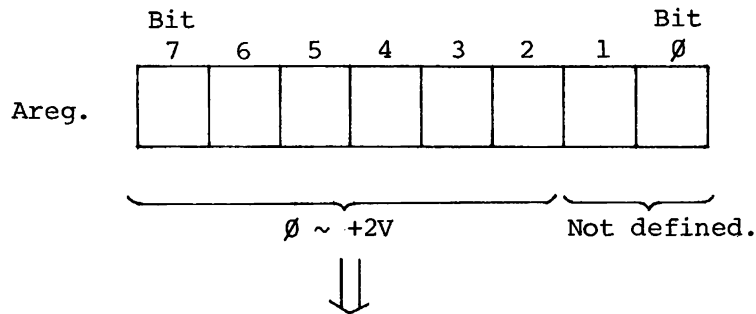
Each bit corresponds to $2V \div 2^6 \fallingdotseq 32mV$. These bits
are all set to 1 when a voltage higher than +2V is
input. They are set to Ø when a negative voltage
is input.

(2) When the C reg. = 02H

The settings for the DIP switches on the main unit back
panel are placed into the A reg. in the following format:

(3) When the C reg. = 03H

The data about the battery voltage is placed in the A reg. See Chapter 11 for the correspondence between the battery voltages and the A reg. values.

(4) When the C reg. = 04H

The main switch settings and the analog input connector trigger terminal state are placed into the A reg.

```
          Bit                           Bit
           7   6   5   4   3   2   1   0
Areg.    ┌───┬───┬───┬───┬───┬───┬───┬───┐
         │ Ø │ Ø │ Ø │ Ø │ Ø │ Ø │   │   │
         └───┴───┴───┴───┴───┴───┴───┴───┘
                                    │   │
               Trigger      ◄───────┘   │
               terminal state           │
                                        │
               Ø = OFF  (+5V)           │
               1 = ON   (ØV)            │
                                        │
                                        │
               Main switch     ◄────────┘
               settings.
               Ø: Main switch is set to OFF.
               1: Main switch is set to ON.
```
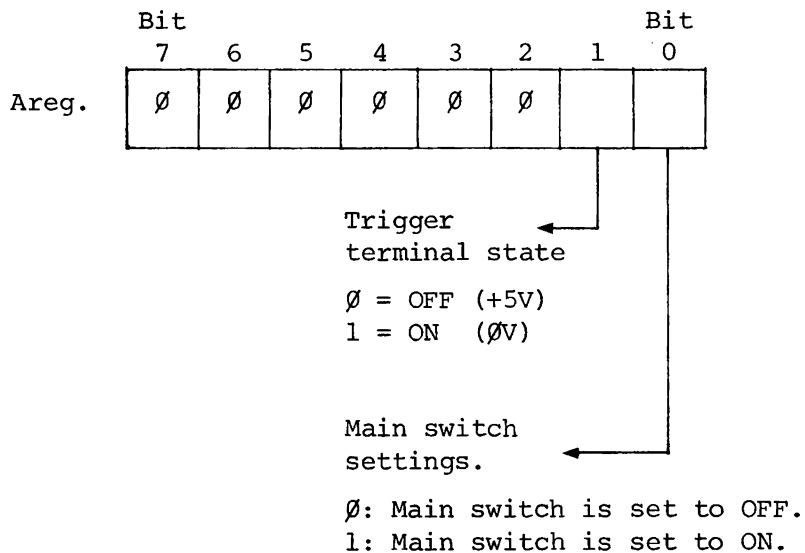
MAPLE may be started even when the main switch is in the off position (by the wake function).

Entry Name	SLAVE	Entry Address	WBOOT + 72H
Function	Controls the communication with the SLAVE CPU.		
Entry parameter	DE = Communication packet starting address.		
Return parameter	A = 00H: Normal termination. ≠ 00H: Abnormal termination. DE = Retains the previous value.		
Explanation			

SLAVE is used by the application program to control the SLAVE CPU directly. See Chapter 13 for details on the functions that SLAVE can perform and the command and data used by SLAVE.

The SLVFLG field in the work area must be set as follows before this BIOS function is called:

The SLVFLG address is:

       0F358H --- For Overseas Version OS

       0F080H --- For Japanese Version OS

Bit 7: Always set to ON.

Bit 6: Set to ON when accessing SLAVE memory (executing command

          00H, 01H, or 02H). Otherwise, this bit is set OFF.

Bit 5: Set to ON when writing data into the SLAVE CPU privileged

memory (addresses 80H - 0ADH). Otherwise, this bit is set

to OFF.

SLAVE immediately terminates abnormally if the SLVFLG field is

found to be set improperly. The calling program must clears the

SLVFLG field to 00H after returning from this BIOS subroutine.

The communication packet has the following format:

```
(DE) ──────────►┌──────────────────────────┐
         ①      │ Send packet              │   2 bytes
                │ starting address         │
                ├──────────────────────────┤
         ②      │ Send packet length       │   2 bytes
                │                          │
                ├──────────────────────────┤
         ③      │ Receive packet           │   2 bytes
                │ starting address         │
                ├──────────────────────────┤
         ④      │ Receive packet length    │   2 bytes
                │                          │
                └──────────────────────────┘
```
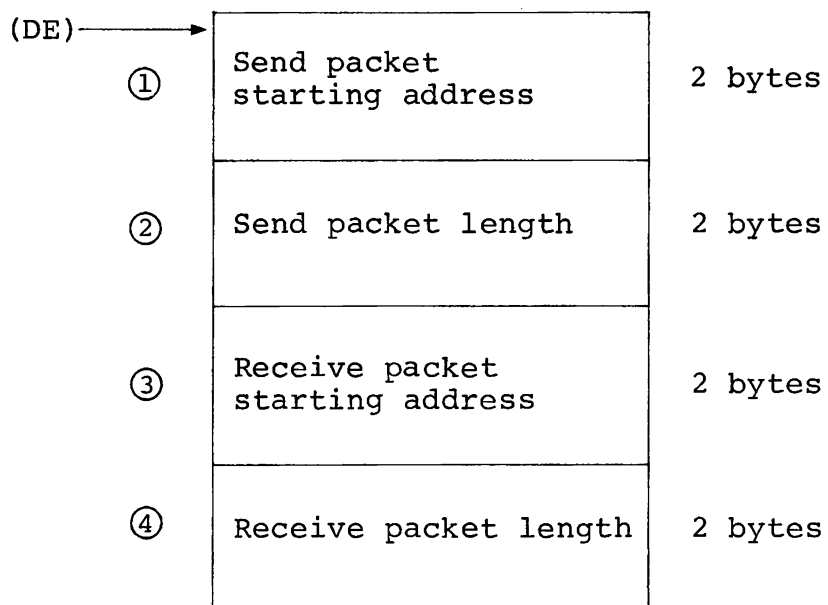
(1) Send packet starting address

(2) Send packet length

A send packet refers to a buffer area which contains a command or a command plus parameters to be passed to the slave CPU.

Send packet

Command	Parameters

Send packet starting address →
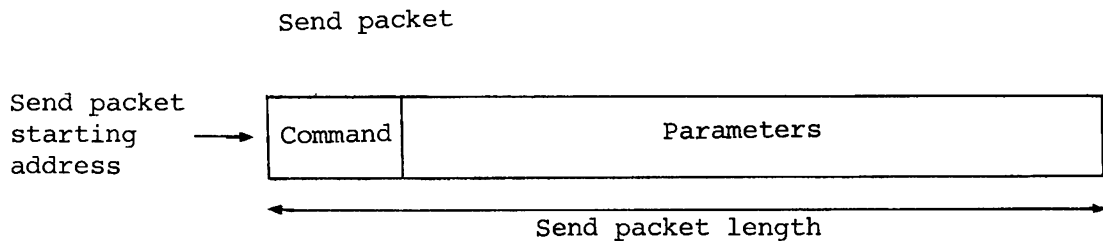
Send packet length

A send packet always begins with a 1-byte command, so the length of a send packet is normally longer 1 byte. When the length is 0, SLAVE does nothing for send requests and performs only receive processing.

(3) Receive packet starting address

(4) Receive packet length

A receive packet is an area for storing the return code and parameters, if any, which the slave CPU returns after processing the command and the parameters passed from the SLAVE CPU in the above format.

Receive packet

Return code	Parameters

Receive packet starting address →

Receive packet length
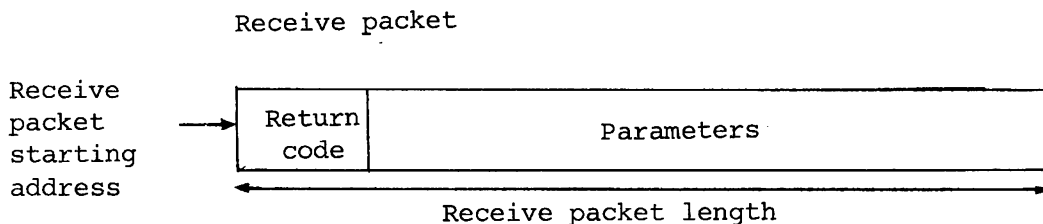
The return code and the contents of the A reg. are the same when SLAVE terminates normally. Since a return code is always returned on normal termination, the receive packet length should longer than 1 byte. When the receive packet length is 0, no data is received from the slave CPU.

The main program can do its own tasks while the slave CPU is processing a command from the main program. The calling program can receive the return code and parameters that the slave CPU returns in response to the previous command by first sending a command or parameter with a receive packet length of 0 specified, then, after performing its main task, issuing a command with a send packet length of 0. During this operation, however, the main program cannot perform any operation which involves slave CPU processing (e.g., screen or MCT processing). (Attempting to do so would result in a SLAVE hang-up.)

Entry Name	RDVRAM	Entry Address	WBOOT + 75H
Function	Reads the contents of VRAM.		
Entry parameter	B = Starting column number in which read is to begin. (1 -80) C = Starting row number in which read is to begin. (1 - Bottom of screen) DE = Number of characters to be read. HL = Address of the area for storing the read data.		
Return parameter	A = 00H: Normal termination. = 01H: Display extends beyond the screen during a read. = FFH: Screen is in graphics mode. Or the starting position specified by B and C is outside the virtual screen. HL = Retains the previous value.		
Explanation			

RDVRAM reads the data on the character mode screen. The screen

has the following structure:

```
                    Set in the B reg.
          1 ───────────────────────────► 80
        ┌ ┌─┬──────────────────────────────┐ ┐
        │ │ │                              │ │} 8 rows ──── Used by the OS
        │ │ │        System screen         │ │
        │ │ ├──────────────────────────────┤ │
        │ │ │        Real screen           │ │} 8 rows ──── Same display as LCD.
 Set in │ │ ├──────────────────────────────┤ │
 the C  │ │ │                              │ │
 reg.  ◄┤ │ │                              │ │
        │ │ │        Virtual screen        │ │
        │ │ │                              │ │} Variable ── See Chapter 16 for
        │ │ │                              │ │              details.
        │ ▼ │                              │ │
        └   │                              │ │
 8 + 8 +    │                              │ │
(Number of  └──────────────────────────────┘ ┘
 rows in the
 virtual screen)
```
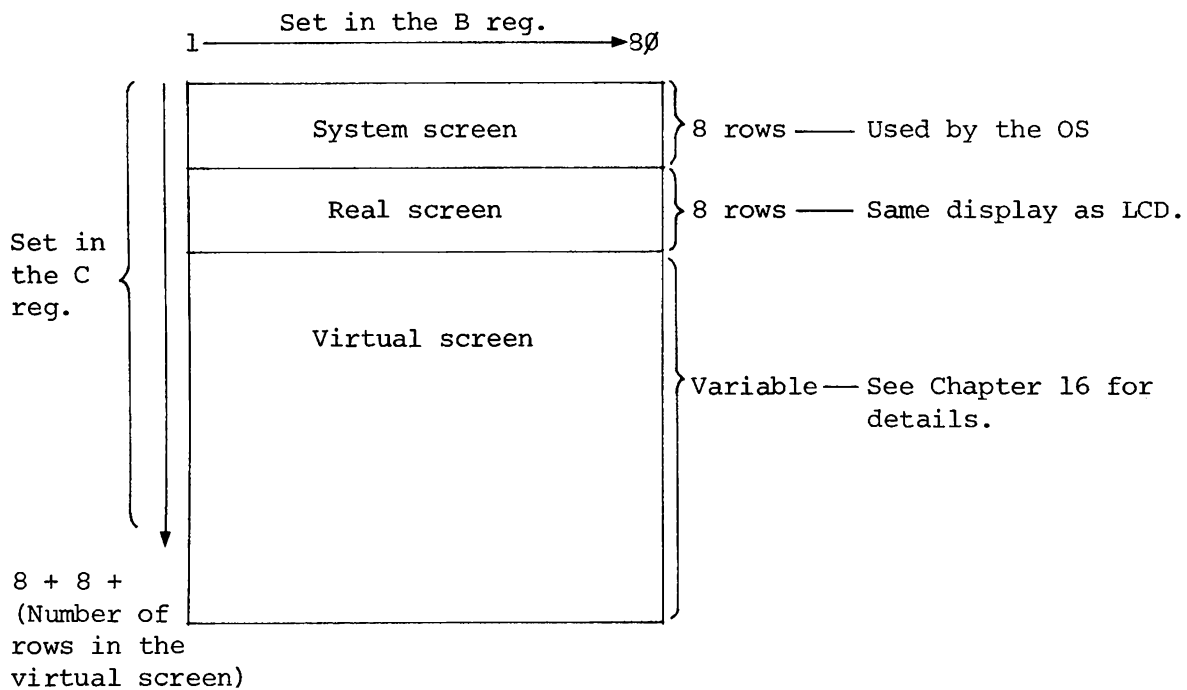
RDVRAM reads the number of characters specified by DE starting at the position designated by B and C and stores them sequentially into the area starting at the address designated by DE. Characters are read from left to right in a row. After the 80th character is read, the leftmost character in the next row is read.

When the number specified in DE is too large and display extends beyond the screen, 00H codes are returned as extra characters until the number of the returned characters equals the value specified in DE. In this case, the A reg. is loaded with a return code of 01H.

Entry Name	MCMTX	Entry Address	WBOOT + 78H
Function	Processes MIOS communication.		
Entry parameter	B = MIOS function code (00H - 15H)		
Return parameter	Described below.		
Explanation			

MCMTX is used to communicate with MIOS (entering commands or receiving data) to control MCT directly.

See Chapter 14 for details of MIOS functions.

Entry Name	POWEROFF	Entry Address	WBOOT + 7BH
Function	Turns main power off.		
Entry parameter	C = 00H: Main power turned off in continue mode. = 01H: Main power turned off in restart mode.		
Return parameter	None.		
Explanation			

POWEROFF is used in application programs to turn MAPLE main power off.

If power has been set off in continue mode, execution continues with the command following this BIOS call when power is turned on. The I/O settings established before the power-off is restored at the same time. This BIOS call must be followed by an EI instruction when power is turned off in continue mode.

If power has been set off in restart mode, execution will start at WBOOT when power is turned on.

See Chapter 9 for details of power-on/off.

Entry Name	USERBIOS	Entry Address	WBOOT + 7EH
Function	Provides the entry to USERBIOS.		
Entry parameter	None.		
Return parameter	None.		
Explanation			

USERBIOS provides an entry point through which the application program makes BIOS calls after loading its own BIOS routine in the RAM USERBIOS area. Presently, USERBIOS serves no purpose.

The following procedure must be observed when using a user-provided BIOS routine through the entry point at USERBIOS:

1) Load the BIOS routine into the RAM USERBIOS area.

2) Replace the contents of addresses (WBOOT + 7EH) + 1 and (WBOOT + 7EH) + 2 with the entry address bytes of the user routine in the USERBIOS area.

3) Call this BIOS in the application program.

See "USERBIOS Usage" for details.

Chapter 5   Keyboard

5.1 General

The MAPLE is furnished with a typewriter keyboard which

contains special keys such as cursor movement keys

(arrow keys) and programmable function keys.   I/O

operations concerning the keyboard is controlled by the

7508 sub-CPU.   When a key entry is made, the 7508

informs the Z80 CPU of the presence of the key entry by

generating an interrupt.   The OS, on receipt of the

interrupt, fetches information from the 7508 identifying

the key location and takes the corresponding action.   In

addition to this key entry function, a number of MAPLE

keyboard functions are supported at the OS level.   Those

keyboard functions are fully discussed in this chapter

(see Chapter 11 for the 7508 CPU).

5.2 Keys and Keyboard Types

- Number of keys: 72 (73 keys for Japanese-language

keyboard)

- Number of switch keys: 6

        * What is a switch key?

        When an ordinary key is pressed, the 7508 CPU

        provides only the information that indicates the

depression of the key. When a switch key is
pressed, however, it provides two types of
information, that is the information indicating the
depression of a key and the information indicating
the release of the key. This kind of keys include
SHIFT and CTRL are used to switch the keyboard
mode. These keys are all controlled by the OS and
application programs need not concern about
this.

- Keyboard types

The MAPLE supports twelve types of keyboards to
accommodate various languages. Keyboard and OS key entry
routine assignments are defined by DIP-SW 1 through 4 in the
MAPLE's ROM compartment. DIP-SW settings are shown on the next
page (see the end of this mannual for key assignments for
different countries).

Keyboard type	DIP-SW				Object OS
	4	3	2	1	
USASCII	1	1	1	1	ASCII OS
France	1	1	1	0	
Germany	1	1	0	1	
England	1	1	0	0	
Denmark	1	0	1	1	
Sweden	1	0	1	0	
Italy	1	0	0	1	
Spain	1	0	0	0	
Norway	0	1	1	0	
Kana	0	0	0	0	Japanese-language JIS OS
Japanese-language JIS keyboard	0	0	0	1	
Japanese-language touch type keyboard	0	0	0	1	Japanese-language touch type OS

0 --- OFF
1 --- ON

Auto repeat keys and switches

Auto repeat keys   (Keys other than shaded keys)

Switch keys

Keyboard other than Japanese-language keyboard
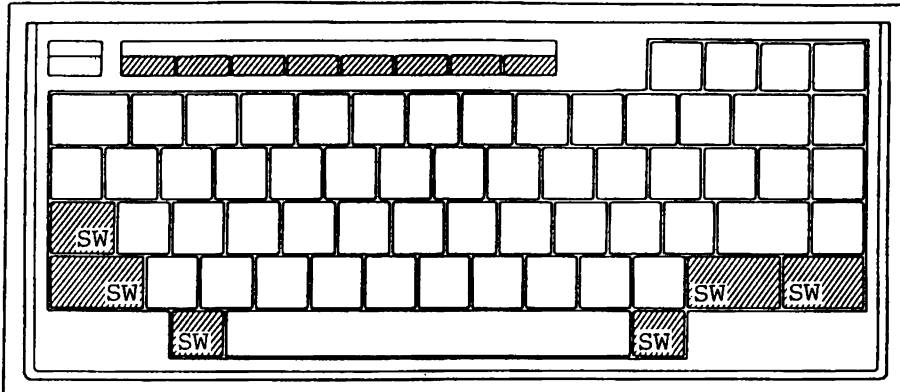
Auto repeat keys and switches
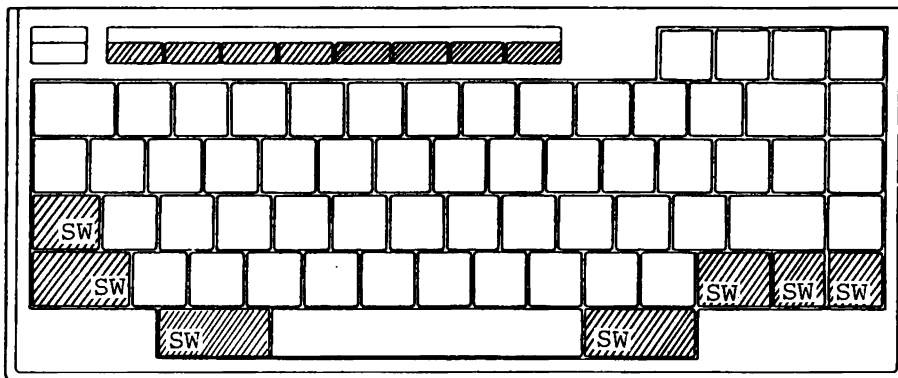
5.3 OS Key Routine Functions

- Keyboard buffers: 32 (The 7508 sub-CPU has 7 unique
                        buffers own.)

- N-key rollover feature: Provided.

- Auto repeat feature: Provided. (See the previous page
                        for auto repeat keys.)
                        Repeat start time -- 656 ms
                        Repeat period ---- 70 ms

- Auto repeat setting:
                        Auto repeat ON/OFF state, repeat
                        start time, and repeat period can
                        be changed using the BIOS CONOUT
                        routine.

- The CAPS, NUM, and GRAPH keyboard modes are indicated by LEDs
  on the keyboard.

5.4 Operation Flow

The steps below show the sequence of operations from key depression to transfer of the key data to the application program.

(1) A key is pressed.

(2) The 7508 scans the keyboard every 30 ms and, if a key entry is sensed, loads the corresponding hardware code into its own buffer.

(3) The 7508 reports the Z80 of a key entry via the interrupt line.

(4)(5)(6) The Z80 takes data from the 7508 buffer via the 7508 port and stores the data into the keyboard buffer. Any data overflowing the keyboard buffer is discarded.

(7) The key routine takes hardware codes out of the keyboard buffer, one at a time, and returns the corresponding key codes to the application program after making the following checks:

- Code for changing the keyboard mode ?

    (SHIFT key, CTRL key, etc.)

- PF key ?

- Subroutine call required?

    (CTRL/ESC - CTRL/PFK)

The above steps are illustrated in flowchart form on the next page.

* 7508 hardware codes

The 7508 hardware codes only identify the corresponding key on the keyboard and have no relation with the keyboard shift mode. Consequently, the key routine determines what code is actually entered according to the previously established state of the SHIFT, GRAPH, or CTRL key. (See Chapter 11 for details on hardware codes.)

Hardware code

```
        ┌─────────────┐
        │   PF key?   │──── Y ────┐
        └─────────────┘           │
               │ N                │
        ┌─────────────┐           │
        │   Key for   │           │
        │ determining │──── Y ────┤
        │ shift mode? │           │
        └─────────────┘           │
               │ N                │
```

Determine key code
based on entered
hardware code con-
sidering shift mode.

Perform key code
conversion according
to DIP switch
settings specifying
keyboard version.

```
        ┌─────────────┐
        │  Key code   │
        │ requiring a sub-│── Y ──┐
        │ routine call?│         │
        └─────────────┘         │
               │ N              │
```

Get characters
out of PF key
string, one at
a time.

Subroutine
call.

Determine key-
board shift
mode.

Return to BIOS.   Return to the beginning
                   of key routine.

Press switch

①

75Ø8

②

Buffer (7 bytes)

③

⑤

Z8Ø

Interrupt
processing

④

⑥ Keyboard buffer
(32 bytes)

Key routine ⑦

BIOS
(CONST, CONIN)

BDOS

Application program

5.5 Keyboard States

5.5.1 Keyboard Mode Transition

The MAPLE ASCII keyboard operates in three modes:
Normal, CAPS and NUM. The Japanese-language OS supports
the Kana mode in addition to these modes.

Normal: Unshifted letters are input in lowercase. For keys
which have two characters on their keytop, lower
letters are input.

CAPS: The same as the Normal mode except that unshifted
letters are accepted in upper case.

NUM: Numbers are input from the numeric keys which
are aligned horizontally on the top of the
keyboard or from the keys having a number
indicated at the upper right of the keytop.
Some symbols are also input. Other keys are
ignored.

Kana: Kana characters are input.

Mode transition diagram (Kana keyboard)
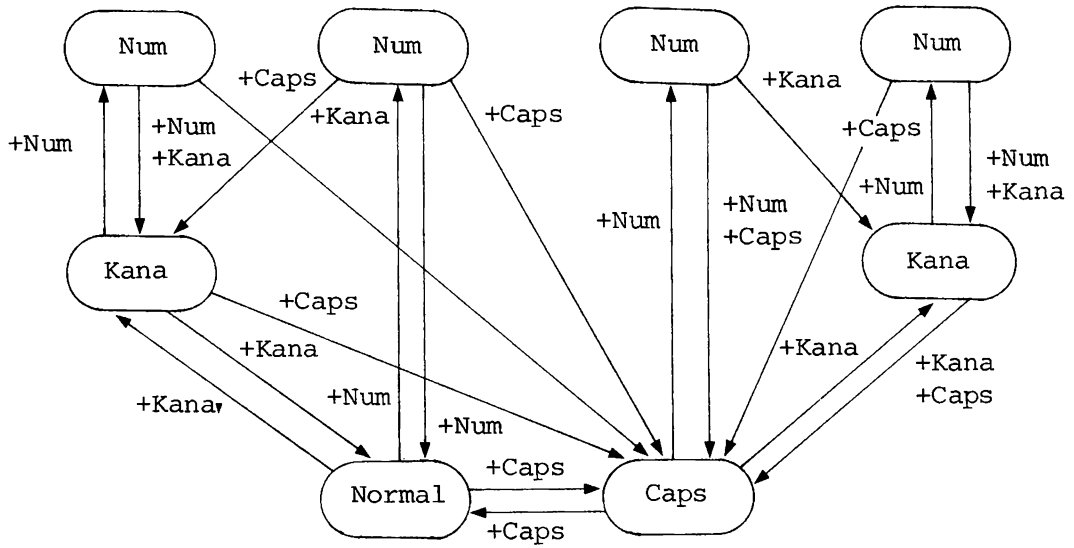
Power on, Reset
Wake

Mode transition diagram (Keyboards except Kana and

Japanese-language keyboards)

Power on
Reset
Wake
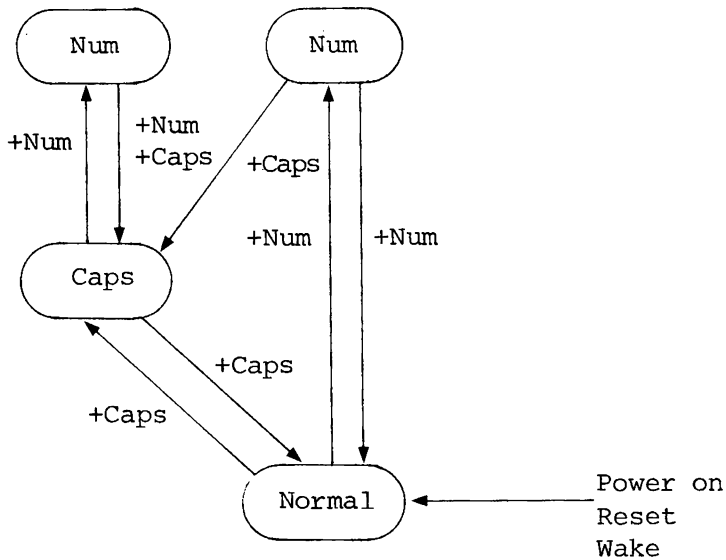
5.5.2 Keyboard State Transition

In any of the keyboard modes given in the previous subsection, a depression of a key returns different codes depending on whether the key is pressed singly or together with the SHIFT, GRPH, or CTRL key. The state transition diagram for the MAPLE keyboard is shown on the next page. The codes here refer to those codes which the application program receives from the keyboard through the BIOS CONIN function or a BDOS function.

Keyboard states (for Non-Japanese-language keyboards)
The keyboard state is determined by the combination of
the keyboard mode and the state of the SHIFT, GRPH, and
CTRL keys. The CTRL key has a higher priority than the
GRPH key; i.e., if the CTRL and GRPH keys are pressed
simultaneously, only the CTRL key is validated.

```
Mode                                          State

Normal ─────────────────────────────────────► Normal
                                      +Shift
                                           └─► Normal shift

Caps ───────────────────────────────────────► Caps
                                   +S
                                        └─────► Caps shift
                      +Graph shift
                    +Graph
              +Controll shift
            +Controll
                                          └──► Graph
          +Controll shift
                                            └─► Graph shift

                                            └─► Control

                                              └► Control shift
        +G    +G +S   +C  +C +S
Num ─────────────────────────────────────────► Num
                              +S
                                   └──────────► Num shift

Kana ────────────────────────────────────────► Kana
(Only for Kana              +S
 keyboard)                     └─────────────► Kana shift
```
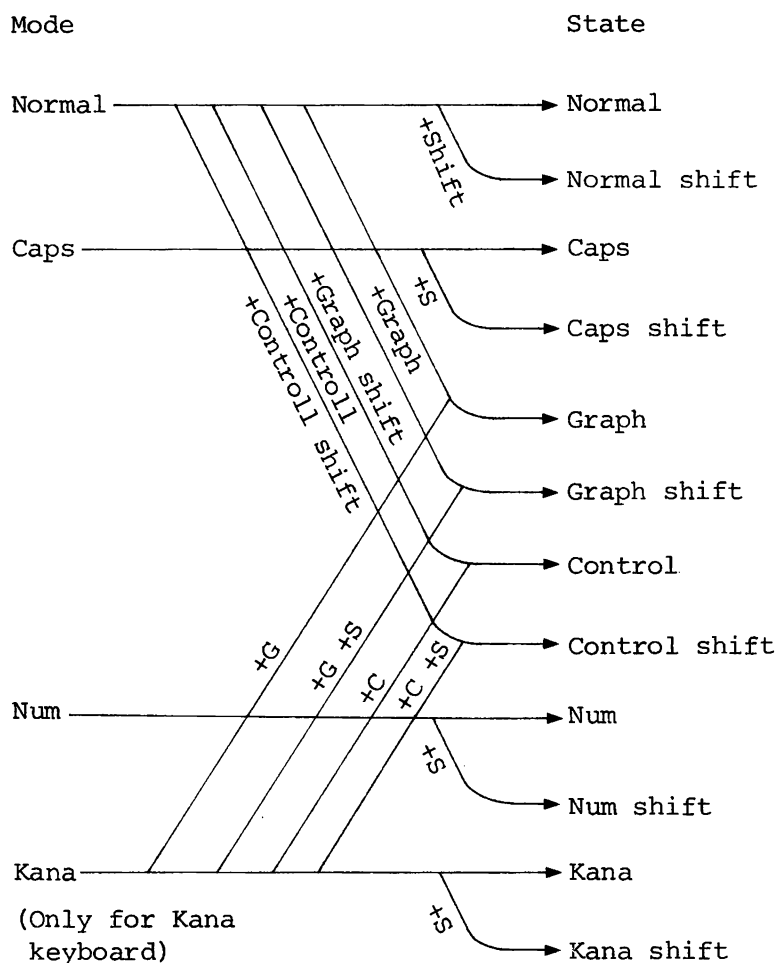
The codes received from keyboards may differ depending
on the state in which the keyboard is.

CTRL —Y→ CTRL

N

GRAPH —Y→ GRAPH

N

NUM —Y→ NUM

N

CAPS —Y→

N

SHIFT ←Y— SHIFT   SHIFT —Y→ CAPS/SHIFT

N   N

NORMAL   CAPS
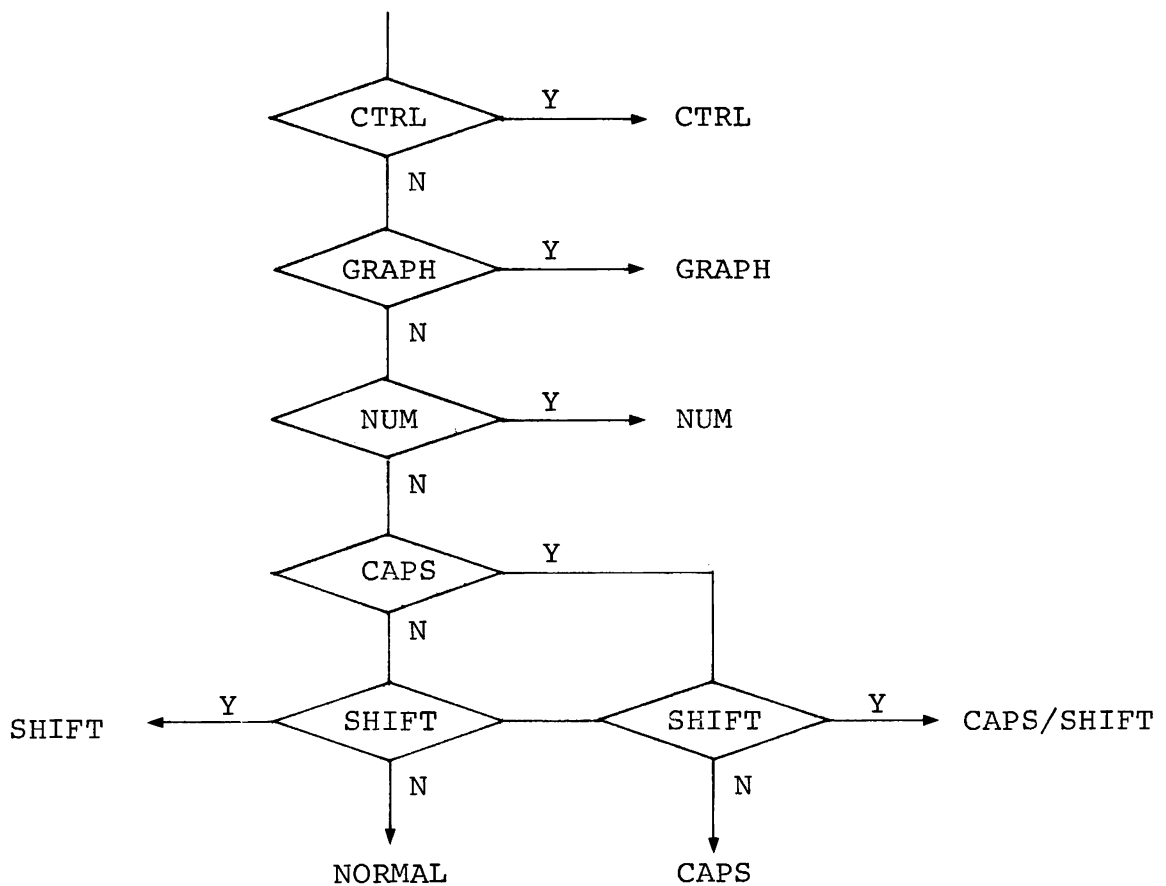
The precedence of the mode keys are as follows:

1. CTRL

2. GRAPH

3. NUM

4. CAPS

5. SHIFT

The shift mode of a higher precedence is honored when
two or more shift mode keys are pressed at the same
time.

5.6 Special Keys

There are some keys which perform special functions besides returning a code when pressed. They are called special keys. The functions of the special keys are described below.

(1) STOP: Clears the key buffer and places only ^C (03H) code into the buffer. Since the STOP key is normally used to interrupt program execution, when pressed, it clears all existing key codes except the ^C code off the buffer so that the MAPLE can respond immediately to this key. You can also enter ^C by typing C while holding down the CTRL key. In this case, the key buffer is not cleared at all.

(2) CTRL/STOP: This key sequence not only performs the above functions but also interrupt the current I/O operation such as an RS-232C receive operation. For example, press these keys to interrupt a program which is stalling, waiting for data from the RS-232C interface. The execution of the RS-232C receive routine is then interrupted and control is returned to the application program, which can then terminates itself by monitoring the ^C code.

Since both STOP and CTRL/STOP load the key buffer with
03H, it is impossible to tell which key was pressed
from the contents of the key buffer alone. The key can
be identified, however, by checking the following flags
in the system work area:

CSTOPFLG --- Overseas version = F10BH

　　　　　　　Japanese-language version = EE25H

　　= 00H: CTRL/STOP not pressed.

　　≠ 00H: CTRL/STOP pressed.

BRKFLG --- Overseas version = F10AH

　　　　　　Japanese-language version = EE24H

　　= 00H: STOP or CTRL/STOP not pressed.

　　≠ 00H: STOP or CTRL/STOP pressed.

Both CSTOPFLG and BRKFLG are set to 00H when the key
buffer is cleared by CONIN.
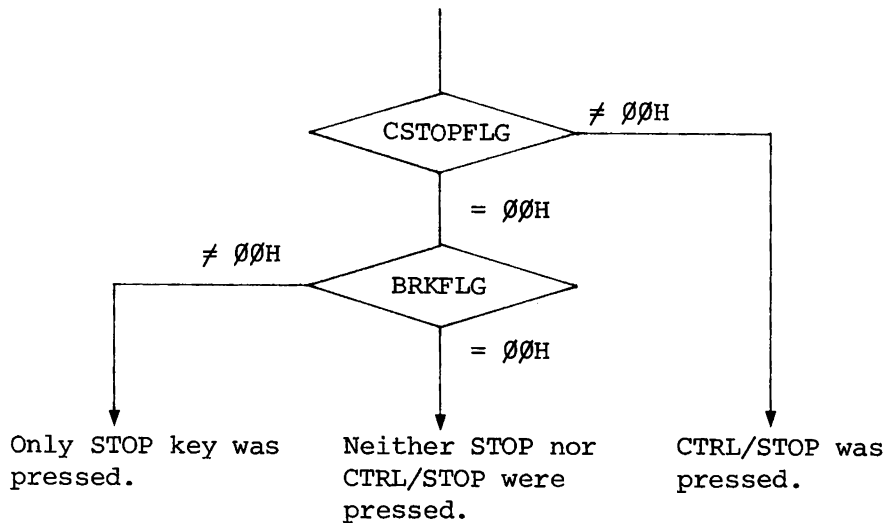
(3) SHIFT/INS: Toggle between the tracking mode and non-tracking mode.

(4) CTRL/INS: Display the portion of the screen on which the cursor is currently positioned. This key sequence is used in non-tracking mode to scroll the screen up to the cursor position.

(5) Cursor movement keys (arrow keys)

There are four cursor movement keys: ↑, ↓, ←, and →.
Since each of them may be pressed independently or in
combination with the SHIFT or CTRL key, it may be
assumed that there are logically twelve movement keys.
The user can assign a code from 00H to 0FFH to each of
these keys. Especially, the OS takes special actions
when codes 80H and 0F8H to 0FFH are entered. These
codes can be set by the application program sending ESC
+ F3H, ESC + F4H, and ESC + F5H via the CONOUT BIOS call.

Code   OS action
00H ------------- The key routine returns the code.
                        (Same as with ordinary keys.)
7FH

80H ------------- No action.

81H ------------- The key routine returns the code.
                        (Same as with ordinary keys.)
0F7H

0F8H ------------ Scrolls the screen one line up.

0F9H ------------ Scrolls the screen one line down.

0FAH ------------ Scrolls the screen one page (8 lines)
                        up.

0FBH ------------- Scrolls the screen one page (8 lines)
                   down.

0FCH ------------- Scrolls the screen to the top of the
                   virtual screen.

0FDH ------------- Scrolls the screen to the bottom of
                   the virtual screen.

0FEH ------------- Displays virtual screen 1.

0FFH ------------- Displays virtual screen 2.

The table below lists the initial values for the cursor
movement keys.

Cursor movement keys	Initial value
⟶	1CH
⟵	1DH
↑	1EH
↓	1FH
SHIFT/ ⟶	8ØH
SHIFT/ ⟵	8ØH
SHIFT/ ↑	F8H
SHIFT/ ↓	F9H
CTRL/ ⟶	FFH
CTRL/ ⟵	FEH
CTRL/ ↑	FAH
CTRL/ ↓	FBH

(6) Programmable function keys

The keys PF1 to PF5 at the top of the keyboard are programmable function keys. They are used with or without the SHIFT key and are numbered PF6 (SHIFT/PF1) to PF10 (SHIFT/PF5) when used with the SHIFT key. Any string of up to 15 characters can be assigned to each of these keys in the following ways:

1) Using the BIOS PUTPFK (WBOOT + 6CH) function.

2) Defining a programmable function key table (160 bytes) having the same structure as that owned by the OS and storing its starting address in the first two bytes at YPFKSTR (0F103H) or at 0EDE9H for Japanese-language OS.

The user using the ASCII OS Version 1.0 must define the programmable function key table in a user area between addresses 8000H to 0FFFFH. If he is using the CP/M CCP area, however, he can define it in any user area at locations 8000H or higher. Those who use other operating systems may define the table anywhere in the user area.

Programmable function key table

Ø3H	d	i	r														For PF1
Ø4H	t	y	p	e													For PF2
Ø4H	s	t	a	t													For PF3
Ø3H	p	i	p														For PF4
Ø6H	c	o	n	f	i	g											For PF5
Ø4H	d	u	m	p													For PF6
Ø6H	s	u	b	m	i	t											For PF7
Ø3H	d	d	t														For PF8
Ø5H	t	e	r	m	CR												For PF9
Ø7H	f	i	l	i	n	K	CR										For PF10

For string definition (15 bytes)

Indicates the string length (ØØH - ØFH). Loaded
with ØØH when no string is defined.

Definitions made using the first method will be reserved until the next CBOOT (depression of the RESET switch) is executed. Therefore, if programmable function keys definitions are modified in an application program, the program must restore the table with the original definitions before terminating processing.

```
                    Start
                      |
                      |
                      v                  Work area
          +----------------------+    +--------------------+
          | Save current         |    |                    |
          | definitions with     |- - - ->                 |
          | BIOS GETPFK.         |    |                    |
          +----------------------+    |                    |
                      |               |                    |
                      v               +--------------------+
          +----------------------+               |
          | Define new           |               |
          | functions with       |               |
          | BIOS PUTPFK.         |               |
          +----------------------+               |
                      |                           |
                 Processing                       |
                      |                           |
                      v                           |
          +----------------------+               |
          | Restore original     |               |
          | definitions with     |<- - - - - - - -
          | BIOS PUTPFK.         |
          +----------------------+
                      |
                      v
                    End
```
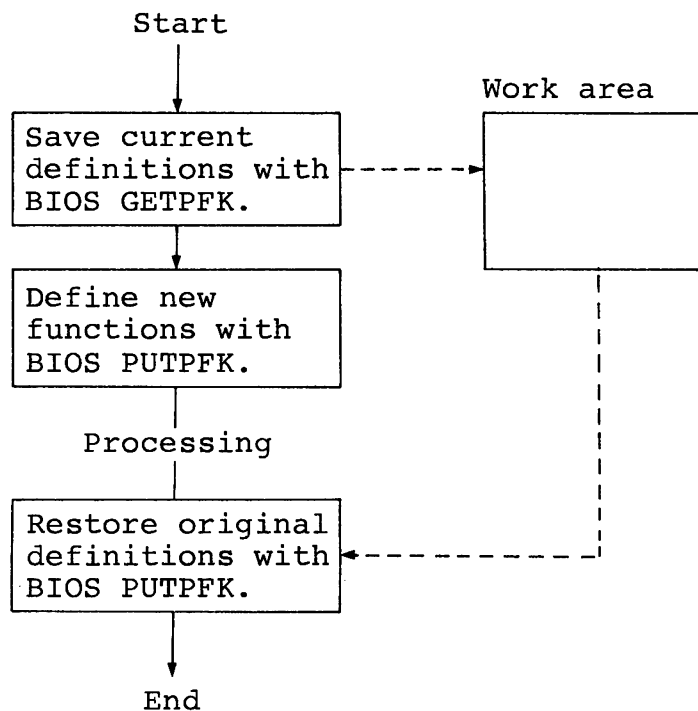
The second method causes the OS to restore the contents of YPFKSTR into its original programmable function key table when a WBOOT is executed. Application programs need not perform any special processing before termination.

The user is recommended to employ the second method because some application programs may unconditionally call WBOOT when a BDOS error occurs.

(7) Keys calling a subroutine

Predefined processing can be performed in the form of a subroutine call by pressing the ESC, PAUSE, HELP, or PF1 to PF5 key while holding down the CTRL key. Since the entries for such subroutines are managed in a table form, application programs can use their own key routines via key entries by changing the entry values in that table.

The starting address of the table is 0F1BAH for the overseas version and 0EED3H for the Japanese-language version.

Table structure

DW   XXXX : CTRL/ESC subroutine entry

DW   XXXX : CTRL/PAUSE subroutine entry

DW   XXXX : CTRL/HELP subroutine entry

DW   XXXX : CTRL/PF1 subroutine entry

DW   XXXX : CTRL/PF2 subroutine entry

DW   XXXX : CTRL/PF3 subroutine entry

DW   XXXX : CTRL/PF4 subroutine entry

DW   XXXX : CTRL/PF5 subroutine entry

The OS use the following two key subroutines:

      CTRL/HELP: System display processing

CTRL/PF5: Hard copy processing

Chapter 6 CONOUT

6.1 Outline

The MAPLE is provided with an LCD of 8 lines by 80
columns as its primary display device. The MAPLE OS
supports several screen modes to allow the user to make
effective use of this LCD. This chapter describes the
screen modes in full detail.

6.2 Screen Configuration

The block diagram for the screen hardware and its peripheral devices is shown on the next page. The screen operation flow is as follows:

1) The program running on the Z80 CPU sends data to be displayed to the SLAVE CPU, directly or using the CONOUT function of BIOS (see Chapter 13, "SLAVE CPU"). (1 in the diagram on the next page.)

2) The SLAVE CPU loads the data into VRAM via the LCD controller (SED1320). (2 and 3 in the diagram.)

3) The LCD controller reads the data in VRAM and displays it on the LCD. (4 and 5 in the diagram.)

VRAM

LCD

⑥ ⑦ ③ ④ ⑤

LCD controller
(SED132Ø)

(Character
generator)

⑦ ②

Z8Ø

① →

SLAVE CPU
(63Ø1)

⑧
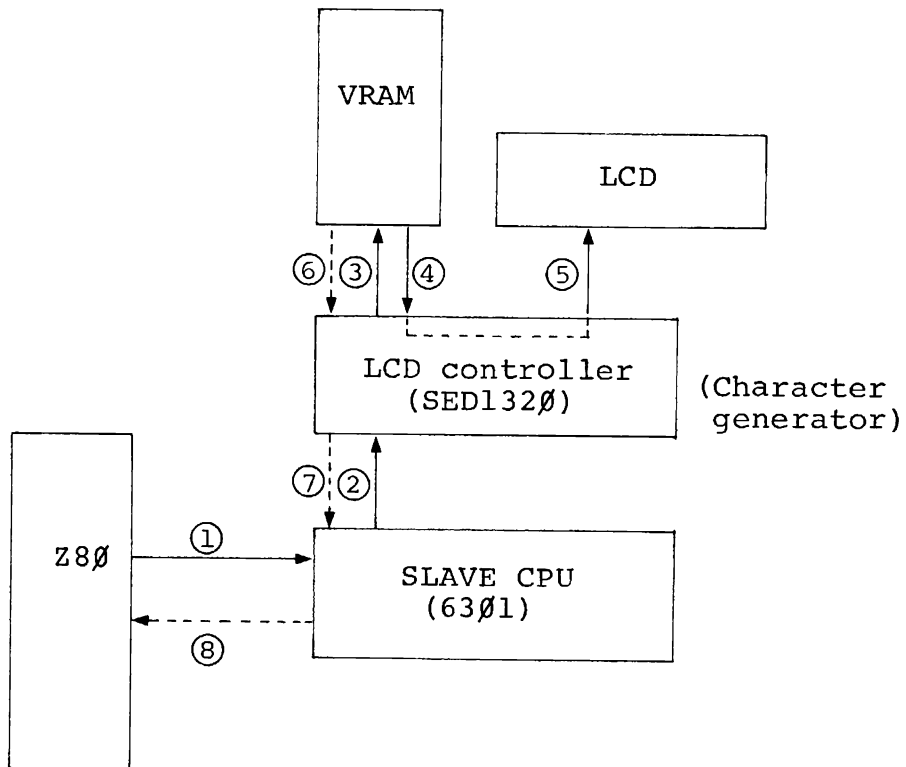
The contents of VRAM can be read in the flow from 6 to
8.

Character fonts are stored in the LCD controller and
their corresponding codes in VRAM. The LCD controller
reads the codes for specified characters from VRAM (4),
converts them into fonts, and sends them to the LCD (5).

Fonts for external characters are defined at the
beginning of VRAM so, if the code read from VRAM is an
external character, the LCD controller reads the
corresponding font from VRAM and transfers it to the
LCD.

See Chapter 13, "SLAVE CPU" for VRAM memory maps.

6.3 Screen Modes

1) System screen and user screen

MAPLE CP/M supports two types of screens; the system screen and the user screen. These two screens are independent of each other. That is, manipulating one screen does not affect the other screen at all.

The user screen has four modes. (These modes are detailed later.)

A hierarchical diagram of the screen modes is shown below.

```
                    +------------------+
                    |   Screen mode    |
                    +------------------+
                            |
          +-----------------+-----------------+
          |                                   |
+------------------+              +------------------+
| System screen    |              |  User screen     |
+------------------+              +------------------+
                                          |
                        +--------+--------+--------+--------+
                        |        |        |        |
                   +--------+ +--------+ +--------+ +--------+
                   | MODE 0 | | MODE 1 | | MODE 2 | | MODE 3 |
                   +--------+ +--------+ +--------+ +--------+
```
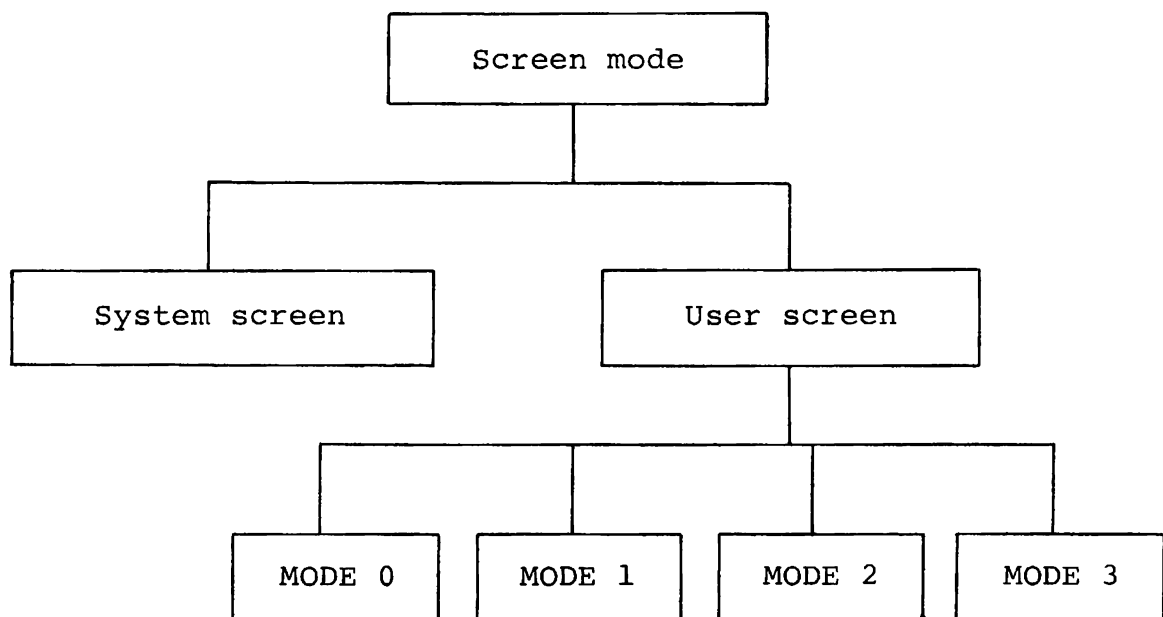
2) Relationship to VRAM

The screens are located in VRAM as shown below:

Character screen
(MODE Ø, 1, 2)

Graphics screen
(MODE 3)

	Character screen		Graphics screen	
8ØØØH	External font definition area	32 characters (32 × 8 = 256)	External font definition area	8ØØØH
81ØØH	System screen	8 lines	System screen	81ØØH
838ØH	Real screen	8 lines	Graphics screen	838ØH
86ØØH	Virtual screen	User screen		
95ØØH	System area		System area	928ØH
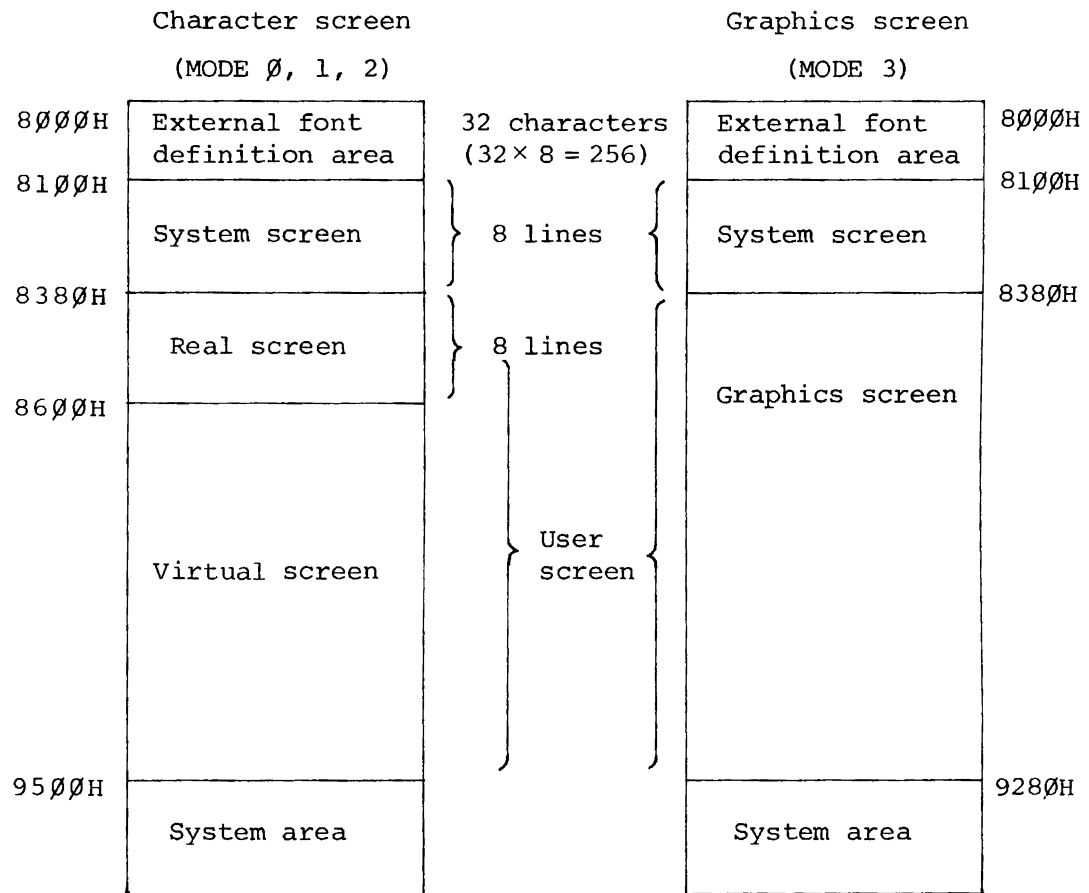
3) System screen

The system screen consists of 8 lines of 80 columns and
is used by the OS to display:

- System Display

- Alarm/wake message

- Password entry prompt message

- "CHARGE BATTERY"

Usually, no application programs normally can send data
onto the system screen.

4) User screen concept

The user screen is divided into three screens: real, virtual, and window screens.

(1) Real screen

The MAPLE LCD display can display 8 lines by 80 columns of data which makes up the real screen. The real screen consists of a window screen and a 1-line function key display area. The function key display area is optional. When no function key definitions are displayed, the real screen size equals the window screen size. From now on, the number of lines of the window screen is represented by h where h is 7 or 8.

(2) Window screen

The window screen is included in the real screen and made up of h lines of 80 columns. The window screen works like a window through which h lines of the virtual screen can be viewed. This window screen can scroll up and down over the virtual screen.

Real screen	Window screen
	Function key display area (optional)
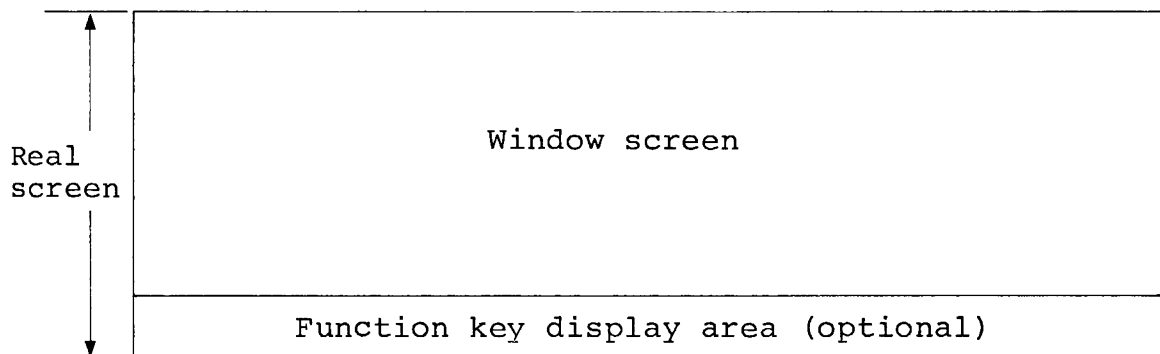
(3) Virtual screen

Although the MAPLE has a large (8 lines of 80 columns)
display screen for this type of display, it has
implemented the concept of virtual screen to meet the
needs of the applications programs which require
larger screens. The MAPLE provides two virtual
screens for the application programs. These screens
may be used for different purposes and displayed
alternately, e.g., displaying data on one screen while
writing display data onto the other screen. The two
screens may be displayed concurrently in some modes.
The sizes of the virtual screens are determined by the
screen mode and user specification.

The entire contents of a virtual screen can be viewed
by scrolling the window screen up and down over the
virtual screen. The window screen scrolls only
vertically.

Window screen

Virtual screen
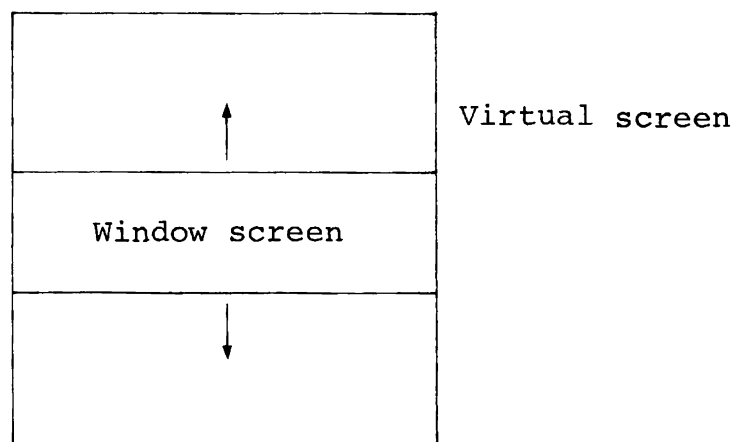
5) User screen modes

There are four user screen modes numbered from MODE 0 to MODE 3. MODE 0 to MODE 2 are character screen modes in which only characters can be displayed. MODE 3 is a graphics screen mode in which both graphics and characters can be displayed. Switching of screen modes can be easily performed using the BIOS call CONFIG or CONOUT.

a) MODE 0 (80-column mode)

Two 80-column wide virtual screens are available in this mode. Their sizes may be defined as desired, as long as one screen consists of at least eight lines and their total number of lines do not exceed 48. The window screen is always located on one of these virtual screens.

Screen RAM structure in MODE 0

80 columns

System screen — 8 lines

LCD ⟸ Real screen — 8 lines

or

Virtual screen 1

Window screen
(8 lines)

nl lines

Virtual screen 2 — n2 lines

Window screen
(8 lines)

The window
screen is
always on one
of the virtual
screen. The
contents of the
window screen
are transferred
to the real
screen for
display on the
LCD.

$h \leqq nl \leqq 40$
$h \leqq n2 \leqq 40$

$nl + n2 \leqq 48$
$(h = 7 \text{ or } 8)$
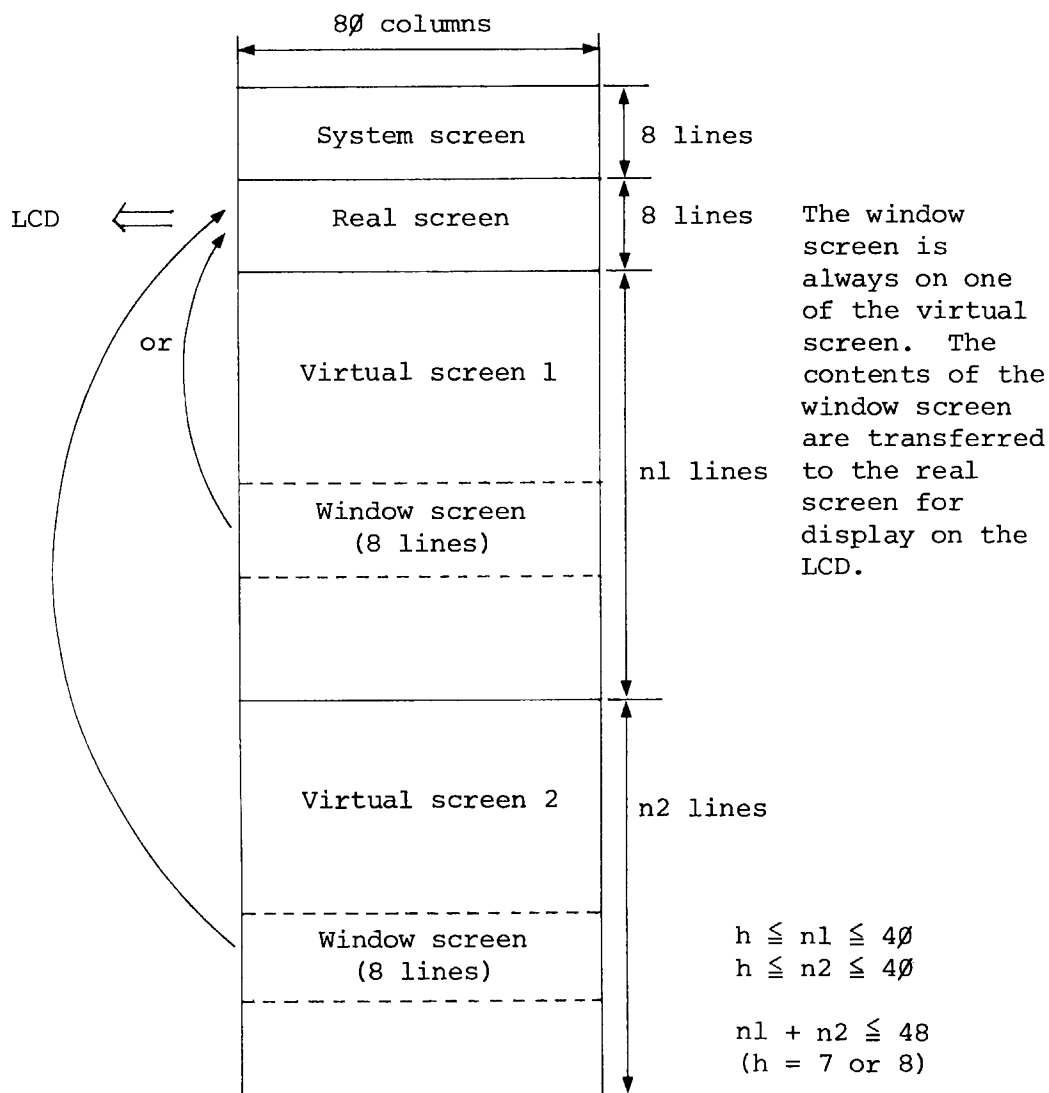
b) MODE 1 (39-column mode)

In this mode, either of the two virtual screens is 39

columns wide. The window screen (WS) has 2h lines. The

first h lines of data is displayed on the left half of

the real screen and the second h lines of data on the

right half of the real screen.

The WS rests on one of the two virtual screens and can

scroll up and down as required.

The two virtual screens can accommodate the same number
of lines in the range 2h ≦ n ≦ 48.

Screen RAM structure in MODE 1

80 columns

System screen — 8 lines

LCD
↑
Real screen

WS1   WS2 — 8 lines

39 columns   2 columns   39 columns

Virtual screen 1

Virtual screen 2

WS 1 (h lines)

2h lines

WS 2 (h lines)

n lines

2h ≦ n ≦ 48
(h = 7 or 8)

Used only to separate two virtual
screens. Two blank columns are
placed on the real screen.
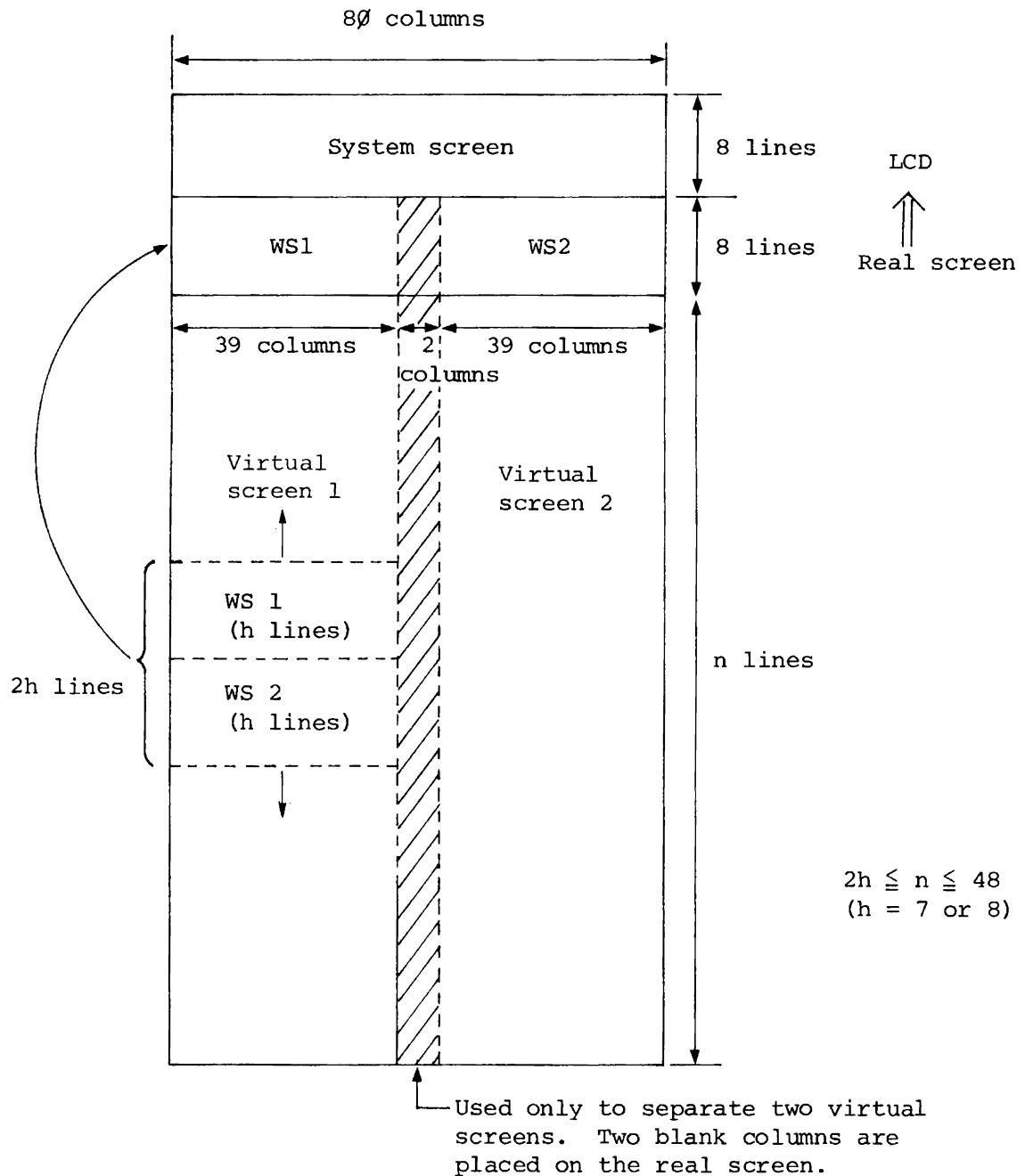
c) MODE 2 (Split screen)

In this mode, two virtual screens are available. Either of them may consist of any number of columns provided that the total number of columns is 79. Since each virtual screen is provided with its own window screen, the contents of the two virtual screens can be displayed on the real screen at the same time. These window screens can scroll independently over the associated virtual window.

Any character or symbol may be defined as the delimiter to separate the two virtual screens on the real screen.

Screen RAM structure in MODE 2

80 columns

System screen — 8 lines

WS1   WS2 — 8 lines

LCD
↑
Real screen

m1 columns   m2 columns

Virtual screen 1

1 column

Virtual screen 2

WS 1 (h lines)

WS 2 (h lines)

n lines

$$n_1 + n_2 = 79$$
$$2h \leqq n \leqq 48$$
$$(h = 7 \text{ or } 8)$$

Delimiter. Any character or symbol may be defined.
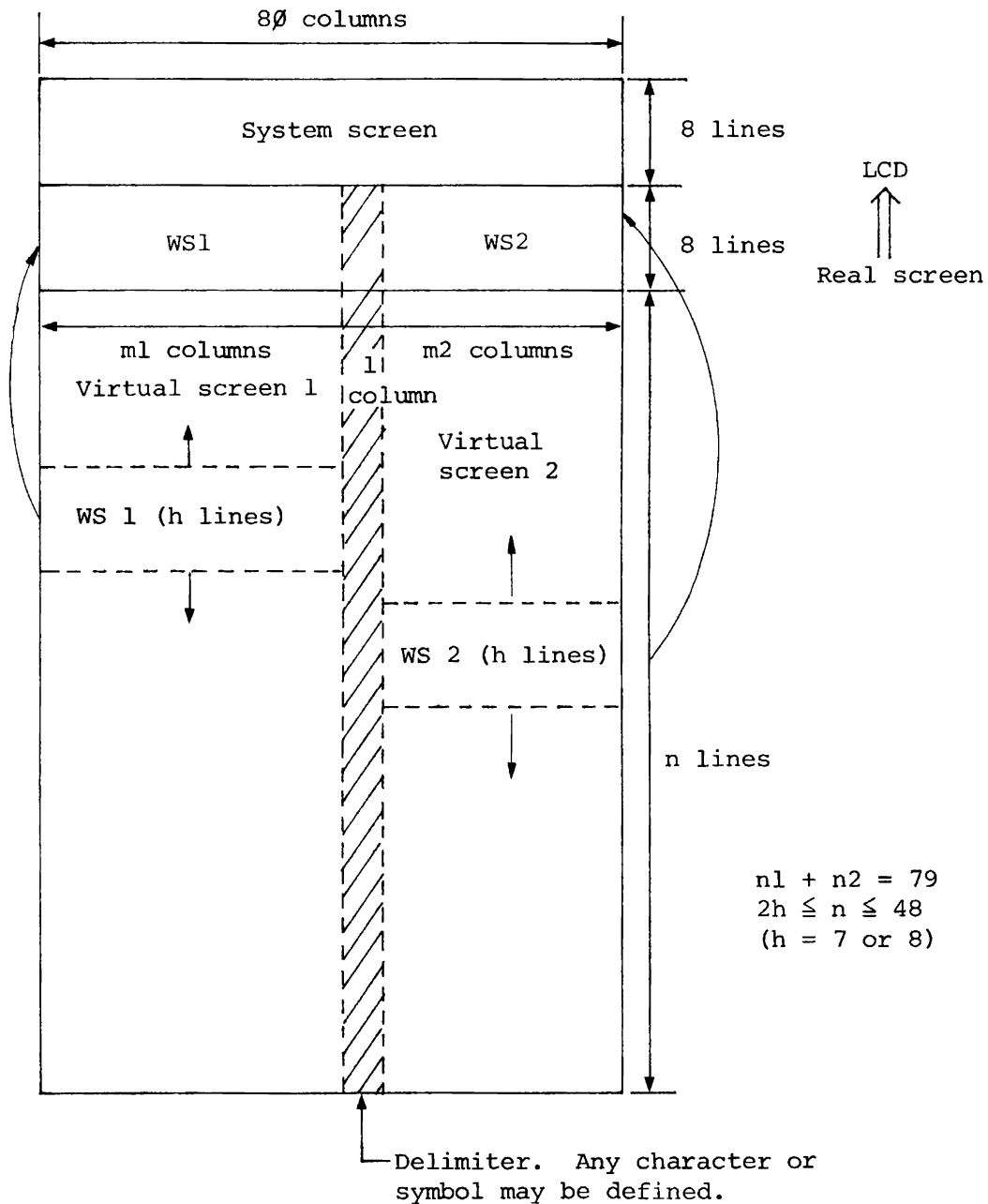
d) MODE 3 (Graphics mode)

This mode permits the application program to display graphics patterns on a dot basis. In this mode, the VRAM is occupied by the 48Ø dots by 64 dots graphics screen. Accordingly, there can be only one user screen at a time. No virtual screen is supported and only a real screen is available in this mode.

Screen RAM structure in mode 3

System screen	8 lines
Graphics screen 48Ø dots x 64 dots	Corresponds to the LCD screen of 8 lines of 8Ø columns. ⇓⇓ LCD
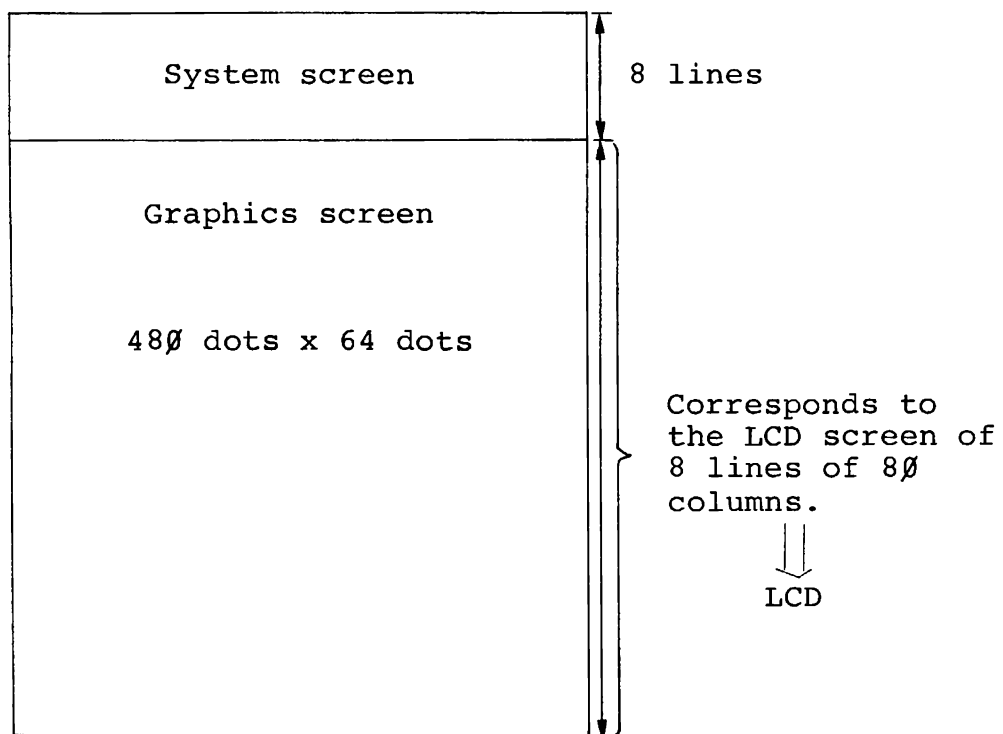
6.4 Special Screen Features

(1) Display screen and write screen

As explained in the previous section, two virtual screens are supported in MODE 0 through MODE 2. However, data can be written only onto the screen in which the cursor is stationed (selected virtual screen). In MODE 0 and MODE 1, only one of the two virtual screens holding the cursor is displayed at a time.

Only one screen is supported in MODE 3 and used for both display and write.

(2) Scroll mode

In character screen modes (MODE 0, 1, 2), the window screen scrolls up and down over the virtual screen in the tracking mode and non-tracking modes.

1) Tracking mode

In this mode, the window screen scrolls following the cursor. That is, the cursor is always in the window (real) screen.

2) Non-tracking mode

The window screen does not follow the cursor movement.

The WS remains in the current position even if the

cursor moves beyond the WS. The cursor stays in the

virtual screen and newly entered data is placed at

the cursor position.

When the mode is switched to the tracking mode, the

WS continues to scroll until the cursor appears in

the WS.

(3) Cursor functions and types

1) Character screen

Any combination of the following cursor functions and

types are available on the character screen:

- Cursor display: ON/OFF

- Cursor type: Underline/block

- Cursor blink: ON/OFF

2) Graphics screen

The cursor type is preset to underline and nonblink

on the graphics screen. The user can control only

the cursor display ON/OFF state.

(4) Block flash (blink)

The displayed data can blink as the selected file

name can on the MENU screen. Since this function is

supported by the slave CPU, the block flash command
must be issued directly to the slave CPU using the
BIOS SLAVE call. For further information, see
Chapter 13, "SLAVE CPU".

(5) User defined characters

User characters can be defined under the following
conditions:

Number of user defined characters: 32 maximum

User defined character codes: 0E0H - 0FFH

0E0H and 0E1H are used by the OS. Although the user
can overwrite the OS defined characters under these
two codes with his own characters, the user defined
characters will be overwritten again by the OS
defined characters if the RESET switch is pressed.
User defined characters under the other codes remain
unchanged when the RESET switch is pressed.

0E0H = " △ "   0E1H = " ⏎ "

Characters can be defined by sending the ESC + 0E0H
sequence through the CONOUT routine.

User defined characters are displayed on the screen
but not printed on the printer.

(6) Character sets

The MAPLE supports the character sets for the
countries listed below. The user can select any of
them by sending the ESC + "C" sequence via CONOUT.
See the character set tables at the back of this
manual.

- USASCII

- FRANCE

- GERMANY

- ENGLAND

- DENMARK

- SWEDEN

- ITALY

- SPAIN

- NORWAY

6.5 How to Use CONOUT

Display of data on the screen is primarily accomplished
by the CONOUT BIOS call. Call the CONOUT using the
following calling sequence:

    Entry address = WBOOT + 09H

    Entry parameter = Load into C reg.

The CONOUT supports various control codes and ESC sequences.
The control codes and ESC sequences are fully described
in the next section. When entering more than one data
byte such as when sending an ESC sequence, call the CONOUT
the required number of times with each data byte loaded
in the C reg.

6.6 CONOUT Functions

The pages that follow list the CONOUT functions that the
MAPLE OS supports to handle control codes or ESC
sequences.

CONOUT SPECIFICATIONS (1)

CODE	FUNCTION	SYSTEM MODE	MODE 0/1/2	MODE 3
05H	ERASE END OF LINE	Deletes to the end of the line from the cursor position on the screen.	Deletes to the end of the line from the cursor position on the virtual screen in write operation.	Same as in system mode.
07H	BELL	Sounds the speaker at 440 Hz for 1 minute. In ASCII Ver. B, this function sounds the speaker for 200 ms.	Same as in system mode.	Same as in system mode.
08H	BACK SPACE	Moves the cursor one position to the left. The cursor does not move if it is in the home position.	Moves the cursor one position to the left in the currently selected virtual screen. The cursor does not move when it is in the home position.	Same as in system mode.
09H	TAB	Does nothing.	Searches for the next tab position following the cursor position to the right in the currently selected virtual screen and moves the cursor to the first tab position encountered. When no tab position is found on the current line, the function moves the cursor to the beginning of the next line. Tab position = (1+8n) positions $n = 0, 1, 2 \ldots$	Searches for the next tab position starting at the current cursor position on the screen to the right and moves the cursor to the first tab position encountered. When no tab position is found on the line, the function moves the cursor to the beginning of the next line. Tab position = (1+8n) positions $n = 0, 1, 2 \ldots$
0AH	LINE FEED	Does nothing.	Moves the cursor down one line in the currently selected virtual screen. The function scrolls up one line when the cursor is on the bottom line of the virtual screen.	Moves the cursor down one line on the screen. The function scrolls up one line when the cursor is on the bottom line of the screen.

CONOUT SPECIFICATIONS (1)

CODE	FUNCTION	SYSTEM MODE	MODE 0/1/2	MODE 3
OBH	HOME	Moves the cursor to the upper left corner on the screen.	Moves the cursor to the upper left corner in the currently selected virtual screen. How the cursor behaves depends on the tracking/non-tracking specification.	Same as in system mode.
OCH	CLEAR SCREEN & HOME	Clears the entire screen and moves the cursor to the beginning of the screen.	Clears the entire virtual screen in write operation and moves the cursor to the beginning of the virtual screen on which the cursor rests. How the cursor behaves depends on the tracking/non-tracking specification.	Same as in system mode.
ODH	CARRIAGE RETURN	Moves the cursor to the first column of the current line. When a character has been displayed in the last column on the line, this function moves the cursor to the first column of the previous line (the line on which the last character was displayed).	Moves the cursor to the first column of the current line. When a character has been displayed in the last column on a line on the virtual screen, this function moves the cursor to the first column of the previous line (the line on which the last character was displayed).	Same as in system mode.

CONOUT SPECIFICATIONS (2)

CODE	FUNCTION	SYSTEM MODE	MODE 0/1/2	MODE 3
10H	SCREEN UP	Does nothing.	Moves up the window screen a screenful of lines (h lines) over the currently selected virtual screen. Display starts at the home position when it moves up beyond the home position. The cursor is held in the original position on the virtual screen.	Does nothing.
11H	SCREEN DOWN	Does nothing.	Moves down the window screen a screenful of lines (h lines) over the currently selected virtual screen. The last line of the virtual screen is set to the bottom of the window screen when display moves down beyond the end of the virtual screen. The cursor is held in the original position on the virtual screen.	Does nothing.
1AH	ERASE END OF SCREEN	Clears to end of the screen from the current cursor position.	Clears to the end of the virtual screen from the current cursor position.	Same as in system mode.
1BH	ESC	Receives the next code as the second parameter of an ESC sequence.	Same as in system mode.	Same as in system mode.

CONOUT SPECIFICATIONS (3)

CODE	FUNCTION	SYSTEM MODE	MODE 0/1/2	MODE 3
1CH	CURSOR RIGHT	Moves the cursor one position to the right on the screen. If the cursor is in the last column on a line, moves it to the beginning of the next line. This function does nothing when the cursor is in the last column of the last line on the screen.	Moves the cursor one position to the right in the currently selected virtual screen. When the cursor is in the last column on a line, moves it to the beginning of the next line. When the cursor is in the last column on the last line of the window screen, display automatically scrolls up one line in the tracking mode and goes beyond the screen in the non-tracking mode. The function does nothing when the cursor is in the last column of the last line on the virtual screen.	Same as in system mode.
1DH	CURSOR LEFT	Moves the cursor one position to the left on the screen. When the cursor is in the first column on a line, moves it to the last column on the previous line. This function does nothing when the cursor is in the home position on the screen.	Moves the cursor one position to the left in the currently selected virtual screen. When the cursor is in the first column on a line, moves it to the last column on the previous line. When the cursor is in the first column on the first line of the window screen, the display automatically scrolls down one line in the tracking mode or it goes beyond the screen in the non-tracking mode. The function does nothing when the cursor is in the home position on the virtual screen.	Same as in system mode.

CONOUT SPECIFICATIONS (3)

CODE	FUNCTION	SYSTEM MODE	MODE 0/1/2	MODE 3
1EH	CURSOR UP	Moves the cursor up one line on the screen. This function does nothing when the cursor is on the first line of the screen.	Moves the cursor up one line in the currently selected virtual screen. When the cursor is on the first line of the window screen, display automatically scrolls down one line in the tracking mode and goes beyond the screen in the non-tracking mode. The function does nothing when the cursor is on the first line of the virtual screen.	Same as in system mode.
1FH	CURSOR DOWN	Moves the cursor down one line on the screen. This function does nothing when the cursor is on the last line of the screen.	Moves the cursor down one line in the currently selected virtual screen. When the cursor is on the last line of the window screen, display automatically scrolls down one line in the tracking mode and goes beyond the screen in the non-tracking mode. The function does nothing when the cursor is on the last line of the virtual screen.	Same as in system mode.

CONOUT SPECIFICATIONS (4)

CODE	FUNCTION	SYSTEM MODE	MODE 0/1/2	MODE 3
ESC"%"	ACCESS CGROM DIRECTLY	Causes MAPLE CGROM to read the character associated with the specified code and displays it in the cursor position on the screen. (Command sequence) 1st byte: ESC 2nd byte: "%" 3rd byte: n $0 \leqq n \leqq 255$	Causes MAPLE CGROM to read the character associated with the specified code and displays it in the cursor position in the currently selected virtual screen. (Command sequence) 1st byte: ESC 2nd byte: "%" 3rd byte: n $0 \leqq n \leqq 255$	Same as in system mode.
ESC"(" ESC")"	PASS THROUGH	Does nothing.	Does nothing.	Does nothing.
ESC"*"	CLEAR SCREEN	Clears the screen and places the cursor in the home position.	Clears the currently selected virtual screen and places the cursor in the home position.	Same as in system mode.
ESC"0"	REVERSE ON	Does nothing.	Does nothing.	Does nothing.
ESC"1"	REVERSE OFF	Does nothing.	Does nothing.	Does nothing.
ESC"2"	CURSOR OFF	Suppresses the cursor display. The cursor can move, though invisible.	Same as in system mode.	Same as in system mode.
ESC"3"	CURSOR ON	Displays the cursor.	Same as in system mode.	Same as in system mode.
ESC"4"	UNDERLINE ON	Does nothing.	Does nothing.	Does nothing.
ESC"5"	UNDERLINE OFF	Does nothing.	Does nothing.	Does nothing.
ESC"6"	HIGHLIGHT ON	Does nothing.	Does nothing.	Does nothing.
ESC"7"	HIGHLIGHT OFF	Does nothing.	Does nothing.	Does nothing.

CONOUT SPECIFICATIONS (5)

CODE	FUNCTION	SYSTEM MODE	MODE 0/1/2	MODE 3
ESC"8"	BLINK ON	Does nothing.	Does nothing.	Does nothing.
ESC"9"	BLINK OFF	Does nothing.	Does nothing.	Does nothing.
ESC"<"	PUSH CURSOR POSITION	Does nothing.	Does nothing.	Does nothing.
ESC"="	SET CURSOR POSITION	Specifies the cursor position on the screen. (Command sequence) 1st byte: ESC 2nd byte: "=" 3rd byte: Row position (m + 31) 4th byte: Column position (n + 31) m and n indicates the position on the virtual screen, where its home position is (1,1).	Specifies the cursor position in the currently selected virtual screen. If the cursor goes off the screen in the tracking mode, this function redisplays the screen so that the cursor appears in the center of the screen. (Command sequence) 1st byte: ESC 2nd byte: "=" 3rd byte: Row position (m + 31) 4th byte: Column position (n + 31) m and n indicates the position on the virtual screen, where its home position is (1,1).	Same as in system mode.
ESC">"	POP CURSOR POSITION	Does nothing.	Does nothing.	Does nothing.

CONOUT SPECIFICATIONS (5)

CODE	FUNCTION	SYSTEM MODE	MODE Ø/1/2	MODE 3
ESC"C"	SET CHARACTER-SET TABLE	Does nothing under the Japanese-language OS. Under the OS for other countries, this function sets up the specified character set. (Command sequence) 1st byte: ESC 2nd byte: "C" 3rd byte: Country Identification Character U: USASCII F: FRANCE G: GERMANY E: ENGLAND D: DENMARK W: SWEDEN I: ITALY S: SPAIN N: NORWAY	Same as in system mode.	Same as in system mode.
ESC"L"	CHANGE CRT COLOR	Does nothing. In new ASCII version (M25Ø3ØCB) kana mode, this function displays the parameters as they are (garbage data).	Same as in system mode.	Same as in system mode.

CONOUT SPECIFICATIONS (6)

CODE	FUNCTION	SYSTEM MODE	MODE 0/1/2	MODE 3
ESC"p"	SCREEN DUMP	Produces a hard copy of the screen on the printer.	Produces a hard copy of the current window screen on the printer.	Same as in system mode.
ESC"T"	ERASE END OF LINE	Deletes to the end of the line from the cursor position on the screen.	Deletes to the end of the line from the cursor position in the currently selected virtual screen.	Same as in system mode.
ESC"y"	ERASE END OF SCREEN	Clears to the end of the screen from the current cursor position.	Clears to the end of the virtual screen from the current cursor position.	Same as in system mode.
ESC 7BH	SECRET	Changes the subsequent output characters to spaces.	Same as in system mode.	Same as in system mode.
ESC 7DH	NON SECRET	Cancels the above mode.	Same as in system mode.	Same as in system mode.
ESC 80H	1BYTE CODE TO 2BYTE CODE (KATAKANA)	Does nothing. In new ASCII version (M25Ø3ØCB) kana mode, this function displays the parameters as they are (garbage data).	Same as in system mode.	Same as in system mode.
ESC 81H	1BYTE CODE TO 2BYTE CODE (HIRAGANA)	Does nothing. In new ASCII version (M25Ø3ØCB) kana mode, this function displays the parameters as they are (garbage data).	Same as in system mode.	Same as in system mode.
ESC 82H	2BYTE CODE TO 2BYTE CODE	Does nothing. In new ASCII version (M25Ø3ØCB) kana mode, this function displays the parameters as they are (garbage data).	Same as in system mode.	Same as in system mode.

CONOUT SPECIFICATIONS (7)

CODE	FUNCTION	SYSTEM MODE	MODE 0/1/2	MODE 3
ESC 90H	PARTIAL SCROLL UP	Does nothing.	(Command sequence) 1st byte: ESC 2nd byte: 9ØH 3rd byte: n-1 ($\emptyset \leq (n-1) \leq R-1$) 4th byte: m ($1 \leq m \leq R$) $(n-1) + (m-1) < R$, where R is the number of the lines on the virtual screen in mode $\emptyset$, 1, or 2 and the number of the lines on the window screen in mode 3. One character line is made up of 8 dot lines. The m-line screen segment from the nth line scrolls up one line as the above command sequence is processed. The $(n+m-1)$th line is left blank.	
ESC 91H	PARTIAL SCROLL DOWN	Does nothing.	(Command sequence) 1st byte: ESC 2nd byte: 91H 3rd byte: n-1 ($\emptyset \leq (n-1) \leq R-1$) 4th byte: m ($1 \leq m \leq R$) $(n-1) + (m-1) < R$, where R is the number of the lines on the virtual screen in mode $\emptyset$, 1, or 2 and the number of the lines on the window screen in mode 3. One character line is made up of 8 dot lines. The m-line screen segment from the nth line scrolls down one line as the above command sequence is processed. The nth line is left blank.	
ESC 92H	SCROLL RIGHT n character	Does nothing. In new ASCII version (M25Ø3ØCB) kana mode, this function displays the parameters as they are (garbage data).	Same as in system mode.	Same as in system mode.
ESC 93H	SCROLL LEFT n character	Does nothing. In new ASCII version (M25Ø3ØCB) kana mode, this function displays the parameters as they are (garbage data).	Same as in system mode.	Same as in system mode.

CONOUT SPECIFICATIONS (8)

CODE	FUNCTION	SYSTEM MODE	MODE 0/1/2	MODE 3
ESC 94H	SET SCROLL STEP	Does nothing.	Specifies the number of lines to scroll when the scroll up or down n line (ESC 96H or 97H) function is executed. (Command sequence) 1st byte: ESC 2nd byte: 94H 3rd byte: Number of lines to be scrolled (n) $1 \leq n \leq h$ (h = number of window lines)	Does nothing.
ESC 95H	SET SCROLL MODE	Does nothing.	Enables and disables the auto-matic scroll. The modes in which the automatic scroll is enabled and disabled are call-ed "tracking mode" and "non-tracking mode," respectively. The default is tracking mode. (Command sequence) 1st byte: ESC 2nd byte: 95H 3rd byte: mode $\emptyset$ (tracking mode) or 1 (non-tracking mode)	Does nothing
ESC 96H	SCROLL UP n LINES	Does nothing.	Moves up the window screen n lines over the virtual screen on which the cursor rests. Display starts in the home position when it moves up beyond the home position. The cursor is held in the original position on the virtual screen. The value of n is specified by the ESC 94H sequence. The default value is 1.	Does nothing.

CONOUT SPECIFICATIONS (8)

CODE	FUNCTION	SYSTEM MODE	MODE 0/1/2	MODE 3
ESC 97H	SCROLL DOWN n LINES	Does nothing.	Moves down the window screen a screenful of lines (n lines) over the virtual screen on which the cursor rests. The last line of the virtual screen is set to the bottom of the window screen when display moves down beyond the end of the virtual screen. The cursor is held in the original position on the virtual screen. The value of n is specified by the ESC 94H sequence. The default value is 1.	Does nothing.

CONOUT SPECIFICATIONS (9)

CODE	FUNCTION	SYSTEM MODE	MODE 0/1/2	MODE 3
ESC 0A0H	INS LED ON	Turns on the insert mode LED.	Same as in system mode.	Same as in system mode.
ESC 0A1H	INS LED OFF	Turns off the insert mode LED.	Same as in system mode	Same as in system mode.
ESC 0A2H	CAPS LOCK LED ON	Turns on the CAPS LOCK LED.	Same as in system mode.	Same as in system mode.
ESC 0A3H	CAPS LOCK LED OFF	Turns off the CAPS LOCK LED.	Same as in system mode.	Same as in system mode.
ESC 0A4H	NUM LED ON	Turns on the NUM input LED.	Same as in system mode.	Same as in system mode.
ESC 0A5H	NUM LED OFF	Turns off the NUM input LED.	Same as in system mode.	Same as in system mode.
ESC 0B0H	FUNCTION KEY CHECK MODE ON	Enables the programmable function key (PF keys) entry. (YPFCMFLG is set to 0FFH.) (Return code) When the C register is loaded with 0, the ASCII code corresponding to the pressed key other than PF keys is returned to the A register. When the C register is loaded with 0FFH, one of the following codes associated with PF1 to PF10, is returned to the A register. OS for the countries other than Japan: 0E0H - 0E9H OS for Japan: 0C0H - 0C9H	Same as in system mode.	Same as in system mode.
ESC 0B1H	FUNCTION KEY CHECK MODE OFF	Disables the programmable function key entry. (YPFCMFLG is set to 0.) When a programmable function key is pressed, the associated string is returned.	Same as in system mode.	Same as in system mode.

CONOUT SPECIFICATIONS (1∅)

CODE	FUNCTION	SYSTEM MODE	MODE 0/1/2	MODE 3
ESC 0C6H	DOT LINE WRITE	Does nothing.	Does nothing.	Draws a line of user-specified dot pattern on the LCD screen. (Command sequence) 1st byte: ESC 2nd byte: ∅C6H 3rd byte: Starting position (horizontal) $<H>$ ml 4th byte: Starting position (horizontal) $<L>$ 5th byte: Starting position (vertical) $<H>$ m2 6th byte: Starting position (vertical) $<L>$ 7th byte: Ending position (horizontal) $<H>$ nl 8th byte: Ending position (horizontal) $<L>$ 9th byte: Ending position (vertical) $<H>$ n2 10th byte: Ending position (vertical $<L>$ 11th byte: Mask pattern 1 12th byte: Mask pattern 2 13th byte: Operation where: $\|ml-nl\| < 16383$ $\|m2-n2\| < 16383$ The pattern must be specified in bit image in 16 bits from mask pattern 1, bit 7 through mask pattern 2, bit ∅. Line segments are masked sequentially by this mask pattern in the mode specified by operation. The dot coordinates pf a slanted line.

CONOUT SPECIFICATIONS (1∅)

CODE	FUNCTION	SYSTEM MODE	MODE 0/1/2	MODE 3
				are automatically calculated by the function. (Operation) Operation specifies the mode in which the dot coordinates corresponding to the 1 bits of the (16-bit) mask pattern are to be masked. The operation codes are: 1 = Off 2 = On 3 = Complement See separate sheets for dot coordinates.

CODE	FUNCTION	SYSTEM MODE	MODE 0/1/2	MODE 3
ESC 0C7H	PSET/PRESET	Does nothing.	Does nothing.	Sets and resets the specified point on the LCD screen. (Command sequence) 1st byte: ESC 2nd byte: 0C7H 3rd byte: Function code 4th byte: Plot position (vertical) nl 5th byte: Plot position (horizontal) <H> 6th byte: Plot position (horizontal) <L>n2 $0 \le nl \le 63$, $0 \le n2 \le 479$ Function code: 1 = PSET (ON) 0 = PRESET (OFF)
ESC 0D0H	DISPLAY MODE SET	Does nothing.	Changes the screen mode and clears the screen. Mode 0 (Command sequence) 1st byte: ESC 2nd byte: 0D0H 3rd byte: Mode [0] 4th byte: Number of lines on VS1, nl 5th byte: Number of lines on VS2, n2 $nl + n2 \le 48$, nl, $n2 \le 8$ The number of columns on VS1 and VS2 is 80. Mode 1 (Command sequence) 1st byte: ESC 2nd byte: 0D0H 3rd byte: Mode [1] 4th byte: Number of lines on VS1, n	Mode 2 (Command sequence) 1st byte: ESC 2nd byte: 0D0H 3rd byte: Mode [2] 4th byte: Number of lines on VS1, n 5th byte: Number of columns on VS1, m 6th byte: Screen delimiter $8 \le n \le 48$, $1 \le m \le 78$ VS1 and VS2 have the same number of lines. VS2 = 79 - (number of columns on VS1) Mode 3 (Command sequence) 1st byte: ESC 2nd byte: 0D0H

CONOUT SPECIFICATIONS (11)

CODE	FUNCTION	SYSTEM MODE	MODE 0/1/2	MODE 3
			$16 \leq n \leq 48$ The number of columns on VS1 and VS2 are 39.	3rd byte: Mode [3]
ESC 0D1H	SELECT DISPLAY SCREEN	Does nothing.	Specifies which virtual screen is to be displayed. (Command sequence) 1st byte: ESC 2nd byte: 0D1H 3rd byte: 0/1 0 = VS1 1 = VS2 The default is VS1.	Does nothing.

CONOUT SPECIFICATIONS (12)

CODE	FUNCTION	SYSTEM MODE	MODE 0/1/2	MODE 3
ESC 0D2H	DIRECT DISPLAY OF PHYSICAL SCREEN	Displays a specified character in the specified position on the real screen. (Command sequence) 1st byte: ESC 2nd byte: ØD2H 3rd byte: Row position (1-8) 4th byte: Column position (1-8Ø) The position must be specified with column and row numbers. This function displays a character directly in any location on the 8Ø×8 screen. This function uses not internal but CG codes.	Same as in system mode.	Same as in system mode.
ESC 0D3H	SELECT FUNCTION KEY DISPLAY	Specifies whether or not function key definitions are to be displayed on the screen. (Command sequence) 1st byte: ESC 2nd byte: ØD3H 3rd byte: Ø/1 Ø = Displayed (The window screen has 7 lines.) 1 = Not displayed (The window screen has 8 lines.)	Same as in system mode.	Same as in system mode.
ESC 0D4H	LOCATE TOP OF SCREEN	Dos nothing.	Sets the window screen to the beginning of the virtual screen on which the cursor rests. The cursor is held in the original position.	Does nothing.

CONOUT SPECIFICATIONS (12)

CODE	FUNCTION	SYSTEM MODE	MODE 0/1/2	MODE 3
ESC 0D5H	LOCATE END OF SCREEN	Does nothing.	Sets the window screen to the end of the virtual screen on which the cursor rests. The cursor is held in the original position.	Does nothing.

CODE	FUNCTION	SYSTEM MODE	MODE 0/1/2	MODE 3
ESC 0D6H	SELECT CURSOR KIND	Used to select the type of the cursor. (Command sequence) 1st byte: ESC 2nd byte: 0D6H 3rd byte: Type of the cursor [0/1/2/3] 0 = Block and blink 1 = Block and nonblink 2 = Underline and blink 3 = Underline and nonblink The default is 0.	Same as in system mode.	Does nothing. (Set to the nonblink underline cursor.)
ESC 0D7H	FIND CURSOR	Does nothing.	Moves the window screen to the cursor position over the virtual screen so that the cursor line will appear near the center of the screen.	Does nothing.
ESC 0E0H	SET DOWN LOAD CHARACTER	Does nothing.	Defines external characters with the codes 0E0H - 0FFH. (Command sequence) 1st byte: ESC 2nd byte: 0E0H 3rd byte: Character code (0E0H - 0FFH) 4th byte: Character pattern (1) 5th byte: Character pattern (2) 6th byte: Character pattern (3) 7th byte: Character pattern (4) 8th byte: Character pattern (5) 9th byte: Character pattern (6) 10th byte: Character pattern (7) 11th byte: Character pattern (8) (ROW SCAN) The lowest 6 bits are used to display a character pattern.	Defines external characters with the codes 0E0H - 0FFH. (grid pattern diagram) . . * . . . 1 (Pattern 1) . . . * . . 2 (Pattern 2) . . . . * 3 (Pattern 3) . . . . * . 4 (Pattern 4) . . * * * * 5 (Pattern 5) . . * . * . 6 (Pattern 6) . . . * . * 7 (Pattern 7) . . . * . * 8 (Pattern 8) 543210 (bit position) The lowest 6 bits are used to display a character pattern.

CONOUT SPECIFICATIONS (14)

CODE	FUNCTION	SYSTEM MODE	MODE Ø/1/2	MODE 3
ESC 0F0H	KEYBOARD REPEAT ON/OFF	Does nothing.	Controls the keyboard repeat function (accepting inputs repeatedly while the key is held down). The default is REPEAT ON. (Command sequence) 1st byte: ESC 2nd byte: ØFØH 3rd byte: <Ø/1> where Ø = REPEAT OFF 1 = REPEAT ON	
ESC 0F1H	SET KEYBOARD REPEAT START TIME	Does nothing.	Specifies the keyboard repeat start time (the interval between the time the first character is entered and the time the second character is taken in when a key is held down). (Command sequence) 1st byte: ESC 2nd byte: ØF1H 3rd byte: n where $1 \leq n \leq 127$ time = n/64 SEC The default value is approx. 656 ms.	
ESC 0F2H	SET KEYBOARD REPEAT INTERVAL TIME	Does nothing.	Specifies the keyboard repeat interval time. (Command sequence) 1st byte: ESC 2nd byte: ØF2H 3rd byte: n where $1 \leq n \leq 127$ time = n/256 SEC The default is about 7Ø ms.	

CONOUT SPECIFICATIONS (14)

CODE	FUNCTION	SYSTEM MODE	MODE 0/1/2	MODE 3
ESC 0F3H	SET ARROW KEY CODE	Does nothing.	Defines the arrow key codes. (Command sequence) 1st byte: ESC 2nd byte: 0F3H Default 3rd byte: Code of → --- 1CH 4th byte: Code of ← --- 1DH 5th byte: Code of ↑ --- 1EH 6th byte: Code of ↓ --- 1FH See "Arrow Key Function Chart" for details.	
ESC 0F4H	SET SCROLL KEY CODE	Does nothing.	Defines codes for SHIFT + arrow keys. (Command sequence) 1st byte: ESC 2nd byte: 0F4H Default 3rd byte: Code of SHIFT + → --- 80H 4th byte: Code of SHIFT + ← --- 80H 5th byte: Code of SHIFT + ↑ --- 0F8H 6th byte: Code of SHIFT + ↓ --- 0F9H See "Arrow Key Function Chart" for details.	

CONOUT SPECIFICATIONS (15)

CODE	FUNCTION	SYSTEM MODE	MODE 0/1/2	MODE 3
ESC 0F5H	SET CTRL KEY CODE	Does nothing.	Defines codes for CTRL + arrow keys. (Command sequence) 1st byte: ESC 2nd byte: 0F4H Default 3rd byte: Code of CTRL + → --- 0FFH 4th byte: Code of CTRL + ↓ --- 0FEH 5th byte: Code of CTRL + ↑ --- 0FAH 6th byte: Code of CTRL + → --- 0FBH See "Arrow Kay Function Chart" for details.	
ESC 0F6H	CLEAR KEY BUFFER	Clears the keyboard buffer (Clears entire data previously entered.)	Same as in system mode.	Same as in system mode.
ESC 0F7H	SET KEY SHIFT	Defines the key shift code. (Command sequence) 1st byte: ESC 2nd byte: 0F7H 3rd byte: Shift Code Shift code Shift mode bit4 ON NUM bit3 ON HIRAGANA bit2 ON KATAKANA bit1 ON CAPS bit0 ON NORMAL The function of the CTRL and SHIFT keys differs depending on the current keyboard state. When two or more bits are ON, the mode corresponding to the highest bit is taken.	Same as in system mode.	Same as in system mode.

6-42

CONOUT SPECIFICATIONS (1)

CODE	FUNCTION	MODE 4	MODE 5	REMARK
05H	ERASE END OF LINE	Same as in system mode.	Same as in system mode.	
07H	BELL	Same as in system mode.	Same as in system mode	
08H	BACK SPACE	Same as in system mode.	Same as in system mode.	
09H	TAB	Moves the cursor to the next tab position on the screen. This function moves the cursor to the beginning of the next line when no tab position is found on the current line. Tab position: Full width mode $8n + 1$ Double width mode $16n + 1$ ($n = 1, 2, \ldots$)	Same as in mode 4.	
0AH	LINE FEED	Moves the cursor down one line on the screen. This function causes the screen to scroll up one line when the cursor is on the bottom line of the screen.	Same as in mode 4.	
0BH	HOME	Same as in system mode.	Same as in system mode.	
0CH	CLEAR SCREEN & HOME	Same as in system mode.	Same as in system mode.	
0DH	CARRIAGE RETURN	Same as in system mode.	Same as in system mode.	

6-43

CONOUT SPECIFICATIONS (2)

CODE	FUNCTION	MODE 4	MODE 5	REMARKS
10H	SCREEN UP	Does nothing.	Does nothing.	
11H	SCREEN DOWN	Does nothing.	Does nothing.	
1AH	ERASE END OF SCREEN	Same as in system mode.	Same as in system mode.	
1BH	ESC	Same as in system mode.	Same as in system mode.	
1CH	CURSOR RIGHT	Moves the cursor one character position (1 column in full width mode or 2 columns in double width mode) to the right on the screen. When the cursor is in the last column on a line, this function moves the cursor to the first column of the next line. This function does nothing when the cursor is in the last column of the last line on the screen.	Same as in mode 4.	
1DH	CURSOR LEFT	Moves the cursor one character position (1 column in full width mode or 2 columns in double width mode) to the left on the screen. When the cursor is in the first column on a line, this function moves the cursor to the last column of the previous line. This function does nothing when the cursor is in the first column of the first line on the screen.	Same as in mode 4.	
1EH	CURSOR UP	Same as in system mode.	Same as in system mode.	
1FH	CURSOR DOWN	Same as in system mode.	Same as in system mode.	

CONOUT SPECIFICATIONS (3)

CODE	FUNCTION	MODE 4	MODE 5	REMARKS
ESC"%"	ACCESS CGROM DIRECTLY	Does nothing.	Does nothing.	(Command sequence) 1st byte: ESC 2nd byte: "%" 3rd byte: n ($\emptyset \leq n \leq 255$)
ESC"(" ESC")"	PASS THROUGH	Does nothing.	Does nothing.	
ESC"*"	CLEAR SCREEN	Same as in system mode.	Same as $\emptyset$CH.	
ESC"0"	REVERSE ON	Displays the subsequent output characters in reverse video.	Does nothing.	Reverse video character size = 1 character size + underline area size The character size depends on the current character width.
ESC"1"	REVERSE OFF	Cancels the reverse video display function.	Does nothing.	
ESC"2"	CURSOR OFF	Same as in system mode.	Same as in system mode.	
ESC"3"	CURSOR ON	Same as in system mode.	Same as in system mode.	
ESC"4"	UNDERLINE ON	Displays characters with underlines.	Same as in mode 4.	
ESC"5"	UNDERLINE OFF	Cancels the underline display function.	Same as in mode 4.	
ESC"6"	HIGHLIGHT ON	Does nothing.	Does nothing.	
ESC"7"	HIGHLIGHT OFF	Does nothing.	Does nothing.	

CONOUT SPECIFICATIONS (4)

CODE		MODE 4	MODE 5	REMARKS
ESC"8"	BLINK ON	Does nothing.	Does nothing.	
ESC"9"	BLINK OFF	Does nothing.	Does nothing.	
ESC"<"	PUSH CURSOR POSITION	Does nothing.	Does nothing.	
ESC"="	SET CURSOR POSITION	Same as in system mode.	Same as in system mode.	
ESC">"	POP CURSOR POSITION	Does nothing.	Does nothing.	
ESC"C"	SET CHARACTER- SET TABLE	Does nothing.	Does nothing.	
ESC"L"	CHANGE CRT COLOR	Does nothing.	Does nothing.	Nothing is executed in old ASCII version (M25Ø3ØCA). In new ASCII version (M25Ø3ØCB), the data in the third parameter is displayed as is.
ESC"p"	SCREEN DUMP	Same as in system mode. (Only 2 or 3 lines (or entire screen) may be specified. See ESC + ØF8H and ESC + ØF9H descriptions for format.)	Same as in system mode.	
ESC"T"	ERASE END OF LINE	Same as in system mode.	Same as in system mode.	Same as ESC Ø5H.
ESC"Y"	ERASE END OF SCREEN	Same as in system mode.	Same as in system mode.	Same as ESC 1AH.
ESC 7BH	SECRET	Same as in system mode.	Same as in system mode.	
ESC 7CH	CHANGE V-RAM	Does nothing.	Does nothing.	
ESC 7DH	NON SECRET	Same as in system mode.	Same as in system mode.	

CONOUT SPECIFICATIONS (5)

CODE	FUNCTION	MODE 4	MODE 5	REMARKS
ESC 80H	1BYTE CODE TO 2BYTE CODE (KATAKANA)	Converts a JIS C6220 code (katakana) into a shift JIS code and places it in the DE registers. (Command sequence) 1st byte: ESC 2nd byte: 80H 3rd byte: JIS C6220 code (Return) DE registers: Shift JIS code See separate sheets for the character conversion chart.		Nothing is executed in old ASCII version (M2503ØCA). In new ASCII version (M2503ØCB), the data in the third parameter is displayed as is.
ESC 81H	1BYTE CODE TO 2BYTE CODE (HIRAGANA)	Converts a JIS C6220 code (hiragana) into a shift JIS code and places it in the DE registers. (Command sequence) 1st byte: ESC 2nd byte: 81H 3rd byte: JIS C6220 code (Return) DE registers: Shift JIS code See separate sheets for the character conversion chart.		Nothing is executed in old ASCII version (M2503ØCA). In new ASCII version (M2503ØCB), the data in the third parameter is displayed as is.
ESC 82H	2BYTE CODE TO 1BYTE CODE	Converts a shift JIS code into a JIS C6220 code and places it in the DE registers. (Command sequence) 1st byte: ESC 2nd byte: 82H 3rd byte: Shift JIS (High) 4th byte: Shift JIS (Low) (Return) DE registers: JIS C6220 code See separate sheets for the character conversion chart.		Nothing is executed in old ASCII version (M2503ØCA). In new ASCII version (M2503ØCB), the data in the third parameter is displayed as is.
ESC 83H	JIS C6226 CODE TO SHIFT JIS CODE	Converts a JIS C6226 code into a shift JIS code and places it in the DE registers. (Command sequence) 1st byte: ESC 2nd byte: 83H 3rd byte: JIS C6226 (High) 4th byte: JIS C6226 (Low)		

CONOUT SPECIFICATIONS (5)

CODE	FUNCTION	MODE 4	MODE 5	REMARKS
		(Return) DE registers: Shift JIS code See separate sheets for the character conversion table.		
ESC 84H	SHIFT JIS CODE TO JIS C6226 CODE	Converts a shift JIS code into the JIS C6226 code and places it in the DE registers. (Command sequence) 1st byte: ESC 2nd byte: 84H 3rd byte: Shift JIS (High) 4th byte: Shift JIS (Low) (Return) DE registers: JIS C6226 code See separate sheets for the character conversion chart.		

CONOUT SPECIFICATIONS (6)

CODE	FUNCTION	MODE 4	MODE 5	REMARKS
ESC 90H	PARTIAL SCROLL UP	Does nothing.	Does nothing.	
ESC 91H	PARTIAL SCROLL DOWN	Does nothing.	Does nothing.	
ESC 92H	SCROLL RIGHT n CHARACTER	Scrolls the screen to the right by n columns. (Command sequence) 1st byte: ESC 2nd byte: 92H 3rd byte: Number of columns to be scrolled, n $1 \leq n \leq$ (maximum number of columns on the screen) Does nothing when an illegal number of columns are specified.	Does nothing.	Nothing is executed in old ASCII version (M25Ø3ØCA). In new ASCII version (M25Ø3ØCB), the data in the third parameter is displayed as is.
ESC 93H	SCROLL LEFT n CHARACTER	Scrolls the screen to the left by n columns. (Command sequence) 1st byte: ESC 2nd byte: 93H 3rd byte: Number of columns to scroll, n $1 \leq n \leq$ (maximum number of columns on the screen) Does nothing when an illegal number of columns is specified.	Does nothing.	Nothing is executed in old ASCII version (M25Ø3ØCA). In new ASCII version (M25Ø3ØCB), the data in the third parameter is displayed as is.

CONOUT SPECIFICATIONS (6)

CODE	FUNCTION	MODE 4	MODE 5	REMARKS
ESC 94H	SET SCROLL STEP	Does nothing. (The number of lines to be scrolled may be specified in the same way as for screen 5. This value is effective only when screen 5 is selected. On screen 4, the number of lines to scroll is always 1. Actual scrolling takes place when ESC + 96H or ESC + 97H is executed.	Specifies the number of lines to be scrolled. (Command sequence) 1st byte: ESC 2nd byte: 94H 3rd byte: Number of lines, n (1 ≤ n ≤ 15) Does nothing when an illegal number of columns is specified.	
ESC 95H	SET SCROLL MODE	Does nothing.	Does nothing.	
ESC 96H	SCROLL UP n LINE	Moves up the screen on which the cursor rests by one line. The cursor is held in the original position on the screen.	Moves up the screen on which the cursor rests by n lines. The cursor is held in the original position on the screen. The value of n equals the value specified in ESC + 94H. The default value is 1.	
ESC 97H	SCROLL DOWN n LINE	Moves down the screen on which the cursor rests by one line. The cursor is held in the original position on the screen.	Moves down the screen on which the cursor rests by n lines. The cursor is held in the original position on the screen. The value of n equals the value specified in ESC + 94H. The default value is 1.	

CONOUT SPECIFICATIONS (7)

CODE	FUNCTION	MODE 4	MODE 5	REMARKS
ESC 0A0H	INS LED ON	Same as in system mode.	Same as in system mode.	
ESC 0A1H	INS LED OFF	Same as in system mode.	Same as in system mode.	
ESC 0A2H	CAPS LOCK LED ON	Same as in system mode.	Same as in system mode.	
ESC 0A3H	CAPS LOCK LED OFF	Same as in system mode.	Same as in system mode.	
ESC 0A4H	NUM LED ON	Same as in system mode.	Same as in system mode.	
ESC 0A5H	NUM LED OFF	Same as in system mode.	Same as in system mode.	
ESC 0B0H	FUNCTION KEY CHECK MODE ON	Same as in system mode.	Same as in system mode.	
ESC 0B1H	FUNCTION KEY CHECK MODE OFF	Same as in system mode.	Same as in system mode.	

ESC 0C0H	CHANGE KANJI MODE	Switches the screen mode between kanji and non-kanji.
		(Command sequence)
		1st byte: ESC
		2nd byte: 0C0H
		3rd byte: Mode
		    non-zero = kanji mode
		    zero = non-kanji mode
		If the mode is switched from non-kanji to kanji, the screen is initialized in the screen mode 4 in which 60 half-size character images (30 kanji images) can be displayed in a line and in which the screen is ready for phrase translation. When the mode is switched from kanji to non-kanji, screen mode 0, VS1=25, VS2=23, VS1, block blink cursor, and tracking mode are selected.
			Switches the screen mode between kanji and non-kanji.	

ESC 0C0H	SCREEN ALL CLEAR AND MASKING GUIDE	Clears the entire screen and masks all system areas.	Same as in mode 4.	

CONOUT SPECIFICATIONS (8)

CODE	FUNCTION	MODE 4	MODE 5	REMARKS
ESC 0C2H	HORIZONTAL DOT LINE WRITE	Draws a horizontal dot line on the LCD screen. (Command sequence) 1st byte: ESC 2nd byte: 0C2H 3rd byte: Starting point (vertical) n1 4th byte: Starting point (horizontal) <H> n2 5th byte: Starting point (horizontal) <L> 6th byte: Length<H> n3 7th byte: Length<L> $0 \leq$ n1 ≤ 63 $0 \leq$ n2 ≤ 479 $1 \leq$ n3 ≤ 480 This function does nothing when the line goes beyond the screen.	Same as in mode 4.	Nothing is executed in old ASCII version (M2530CA). In new ASCII version (M2530CB), the data specified in the third and subsequent parameters is displayed as is.
ESC 0C3H	HORIZONTAL DOT LINE ERASE	Draws a horizontal space dot line on the LCD screen. (Command sequence) 1st byte: ESC 2nd byte: 0C3H 3rd byte: Starting point (vertical) n1 4th byte: Starting point (horizontal) <H> n2 5th byte: Starting point (horizontal) <L> 6th byte: Length<H> n3 7th byte: Length<L> $0 \leq$ n1 ≤ 63 $0 \leq$ n2 ≤ 479 $1 \leq$ n3 ≤ 480	Same as in mode 4.	Nothing is executed in old ASCII version (M2530CA). In new ASCII version (M2530CB), the data specified in the third and subsequent parameters is displayed as is.

CONOUT SPECIFICATIONS (8)

CODE	FUNCTION	MODE 4	MODE 5	REMARKS
		This function does nothing when the line goes beyond the screen.		
ESC 0C4H	VERTICAL DOT LINE WRITE	Draws a vertical dot line on the LCD screen. (Command sequence) 1st byte: ESC 2nd byte: $\emptyset$C4H 3rd byte: Starting point (vertical) n1 4th byte: Starting point (horizontal) < H > n2 5th byte: Starting point (horizontal) < L > 6th byte: Length < H > n3 $\emptyset \le$ n1 $\le$ 63 $\emptyset \le$ n2 $\le$ 479 1 $\le$ n3 $\le$ 64 This function does nothing when the line goes beyond the screen.	Same as in mode 4.	Nothing is executed in old ASCII version (M25$\emptyset$3$\emptyset$CA). In new ASCII version (M25$\emptyset$3$\emptyset$CB), the data specified in the third and subsequent parameters is displayed as is.

CONOUT SPECIFICATIONS (9)

CODE	FUNCTION	MODE 4	MODE 5	REMARKS

ESC 0C5H	VERTICAL DOT LINE ERASE	Draws a space dot line in the vertical direction on the LCD screen.
		
		(Command sequence)
		1st byte: ESC
		2nd byte: 0C5H
		3rd byte: Starting point (vertical) n1
		4th byte: Starting point (horizontal) $< H > n2$
		5th byte: Starting point (horizontal) $< L >$
		6th byte: Length $< H > n3$
		7th byte: Length $< L >$
		    $\emptyset \le n1 \le 63$
		    $\emptyset \le n2 \le 479$
		    $1 \le n3 \le 480$
		This function does nothing when the line goes beyond the screen.
			Same as in mode 4.	Nothing is executed in old ASCII version (M25030CA). In new ASCII version (M25030CB), the data specified in the thrid and subsequent parameters is displayed as is.

CONOUT SPECIFICATIONS (9)

CODE	FUNCTION	MODE 4	MODE 5	REMARKS
ESC 0C6H	DOT LINE WRITE	Draws a line of user-specified dot pattern on the LCD screen. (Command sequence) 1st byte: ESC 2nd byte: ØC6H 3rd byte: Starting position (horizontal) < H > ml 4th byte: Starting position (horizontal) < L > 5th byte: Starting position (vertical) < H > m2 6th byte: Starting position (vertical) < L > 7th byte: Ending position (horizontal) < H > nl 8th byte: Ending position (horizontal) < L > 9th byte: Ending position (vertical) < H > 10th byte: Ending position (vertical) < L > 11th bute: Mask pattern 1 12th byte: Mask pattern 2 13th byte: Operation where |ml-nl| < 16383 |m2-n2| < 16383 The pattern must be specified in bit image in 16 bits from mask pattern 1, bit 7 through mask pattern 2, bit Ø. Line segments are masked sequentially by this mask pattern in the mode specified by operation. The dot coordinates of a slanted line are automatically calculated by the function. (Operation) Operation specifies the mode in which the dot coordinates corresponding to the 1 bits of the (16) mask pattern are to be masked. The operation codes are: 01H = Off 02H = On 03H = Complement See separate sheets for dot coordinates.		

CONOUT SPECIFICATIONS (1∅)

CODE	FUNCTION	MODE 4	MODE 5	REMARKS
ESC 0C7H	PSET/PRESET	Sets and resets the specified point on the LCD screen. (Command sequence) 1st byte: ESC 2nd byte: ∅C7H 3rd byte: Function code 4th byte: Plot position (vertical) nl 5th byte: Plot position (horizontal) < H > n2 6th byte: Plot position (horizontal) < L > ∅ ≤ nl ≤ 63, ∅ ≤ n2 ≤ 479 Function code: 1 = PSET (ON) ∅ = PRESET (OFF)	Same as in mode 4.	
ESC 0C8H	CHARACTER SIZE WIDE	Sets the screen character display size to double width.	Same as in mode 4.	
ESC 0C9H	CHARACTER SIZE NARROW	Sets the screen character display size to full width.	Same as in mode 4.	
ESC 0CBH	SET KEISEN MODE	Sets and resets the ruler line mode for the ruler line data. (Command sequence) 1st byte: ESC 2nd byte: ∅CBH 3rd byte: nonzero = ruler line mode on. zero = ruler line mode off. The ruler line data includes the following: (SHIFT JIS CODE) 83F9H – 83FCH	Same as in mode 4.	

6-56

CONOUT SPECIFICATIONS (1Ø)

CODE	FUNCTION	MODE 4	MODE 5	REMARKS
		849FH - 84AEH 84B1H - 84BBH 84C3H - 84F4H F74ØH - F747H		

CONOUT SPECIFICATIONS (11)

CODE	FUNCTION	MODE 4	MODE 5	REMARKS

ESC 0CCH	CONTROL GUIDE DISPLAY SET/RESET	Turns on or off display on the system area.
		(Command sequence)
		1st byte: ESC
		2nd byte: 0CCH
		3rd byte: BIT 7 .... Shift mode (1-6 coloumns)
		    BIT 6 .... Input data display (15-53 columns)
		    BIT 5 .... Entry subguide (54-60 columns)
		    BIT 4 .... Convert mode guide (11-12 columns)
		    BIT 3 .... Guide line
		Data is masked if the corresponding bit is on, and displayed if the bit is off. More than one mode can be specified.		

ESC 0CDH	CLEAR GUIDE	Clears vertical dot lines 46 through 64 (the area below the guide line).	Does nothing.	

ESC 0CEH	CHANGE CONVERT MODE	Switches between the convert and nonconvert modes.
		(Command sequence)
		1st byte: ESC
		2nd byte: 0CEH
		3rd byte: Change information
		    BIT 0 ... ON: Convert mode OFF: Nonconvert mode
		    BIT 6 ... ON: Enables convert mode switching from the keyboard.
		    BIT 7 ... ON: Disables convert mode switching from the keyboard.
		Bit 7 takes precedence over bit 6 when both bits are on. The keyboard buffer is cleared each time the mode is switched from convert to nonconvert and vice versa. Actual switching takes place when CONIN or CONST is executed. The guide line display is also updated at this time.		

CONOUT SPECIFICATIONS (12)

CODE	FUNCTION	MODE 4	MODE 5	REMARKS

ESC 0D0H	DISPLAY MODE SET	Switches the screen mode between normal (screen 4) and zoom (screen 5).
		
		(Command sequence)
		1st byte: ESC
		2nd byte: 0D0H
		3rd byte: Mode
		    04H: Normal mode 05H: Zoom mode
		4th byte: Number of columns
		    1-60 in normal mode
		    1-100 in zoom mode
		Switching into the same mode is regarded as changing the number of columns and the cursor is placed in the home position.		

ESC 0D1H	SELECT DISPLAY SCREEN	Does nothing.	Does nothing.	

ESC 0D2H	DIRECT DISPLAY OF PHYSICAL SCREEN	Displays a character in the specified position on the real screen.
		
		(Command sequence)
		1st byte: ESC
		2nd byte: 0D2H
		3rd byte: Row position (1-3 lines)
		4th byte: Column position (1-60 columns)
		5th byte: Attribute
		    zero: Full width Nonzero: Double width
		6th byte: Display specification
		    zero: User area Nonzero: Guide area
		7th byte: Character code HIGH
		8th byte: Character code LOW
		When the 7th byte is a code from 20H to 7FH or from 0A0H to 0DFH, this function regards the code as consisting only of one byte and ignores the 8th byte. When the 7th byte is a code from 00H to 1FH, the function takes it as an error and does nothing. The display position may be set to any location on the screen with the line and column numbers.		

CONOUT SPECIFICATIONS (12)

CODE	FUNCTION	MODE 4	MODE 5	REMARKS
		When the 6th byte indicates the user area, this function displays the character in 16 × 22 dot matrix according to the specified attributes. When the 6th byte indicates the guide area, the function displays the character in 16 × 18 dot matrix ignoring the attribute. When the 3rd line is specified as the user area, it is treated as the guide line.		
ESC 0D3H	SELECT FUNCTION KEY DISPLAY	Does nothing.	Does nothing.	
ESC 0D4H	LOCATE TOP OF SCREEN	Does nothing.	Does nothing.	
ESC 0D5H	LOCATE END OF SCREEN	Does nothing.	Does nothing.	

CONOUT SPECIFICATIONS (13)

CODE	FUNCTION	MODE 4	MODE 5	REMARKS
ESC 0D6H	SELECT CURSOR KIND	Does nothing. (Set to the nonblink underline cursor.)	Does nothing. (Set to the 4×4 dot blinking block cursor.)	
ESC 0D7H	FIND CURSOR	Does nothing.	Does nothing.	
ESC 0E0H	SET DOWN LOAD CHARACTER	Does nothing.	Does nothing.	
ESC 0F0H	KEYBOARD REPEAT ON/OFF	Controls the keyboard repeat function (receiving key code repeatedly while a key is held). (Command sequence) 1st byte: ESC 2nd byte: 0F0H 3rd byte: 0 (repeat off) 1 (repeat on) The default is repeat on.	Same as in mode 4.	
ESC 0F1H	SET KEYBOARD REPEAT START TIME	Specifies the keyboard repeat start time (interval between the time the first data is entered and the time the second data is taken when the key is held). (Command sequence) 1st byte: ESC 2nd byte: 0F1H 3rd byte: n (n/64 SEC) (1 ≤ n ≤ 127) The default is approx. 656 ms.	Same as in mode 4.	
ESC 0F2H	SET KEYBOARD REPEAT INTERVAL TIME	Specifies the keyboard repeat interval time. (Command sequence) 1st byte: ESC 2nd byte: 0F2H 3rd byte: n (n/256 SEC) (1 ≤ n ≤ 127) The default is approx. 70 ms.	Same as in mode 4.	

CONOUT SPECIFICATIONS (14)

CODE	FUNCTION	MODE 4	MODE 5	REMARKS
ESC 0F3H	SET ARROW KEY CODE	Defines the arrow key codes. (Command sequence) 1st byte: ESC 2nd byte: 0F3H Default 3rd byte: Code of → --- 1CH 4th byte: Code of ← --- 1DH 5th byte: Code of ↑ --- 1EH 6th byte: Code of ↓ --- 1FH See "Arrow Key Function Chart" for details.		See "Arrow Key Function Chart".
ESC 0F4H	SET SCROLL KEY CODE	Defines codes for SHIFT + arrow key combinations. (Command sequence) 1st byte: ESC 2nd byte: 0F4H Default 3rd byte: Code of SHIFT + → --- 80H 4th byte: Code of SHIFT + ← --- 80H 5th byte: Code of SHIFT + ↑ --- 0F8H 6th byte: Code of SHIFT + ↓ --- 0F9H See "Arrow Key Function Chart" for details.		See "Arrow Key Function Chart".
ESC 0F5H	SET CTRL KEY CODE	Defines codes for CTRL + arrow key combinations. (Command sequence) 1st byte: ESC 2nd byte: 0F4H Default 3rd byte: Code of CTRL + → --- 0FFH 4th byte: Code of CTRL + ← --- 0FEH 5th byte: Code of CTRL + ↑ --- 0FAH 6th byte: Code of CTRL + ↓ --- 0FBH See "Arrow Key Function Chart" for details.		See "Arrow Key Function Chart".
ESC 0F6H	CLEAR KEY BUFFER	Same as in system mode.	Same as in system mode.	
ESC 0F7H	SET KEY SHIFT	Same as in system mode.	Same as in system mode.	

CONOUT SPECIFICATIONS (15)

CODE	FUNCTION	MODE 4	MODE 5	REMARKS
ESC 0F8H	SCREEN DUMP OF 44 DOT-LINE	Takes a dump of 1 to 44 vertical dot lines (not including the guide line) on the LCD screen.	This function only accepts the sequence but carries out no actual dump. (Actual dump is initiated when the user later sets the screen to mode 4 with ESC + 0D0H.)	
ESC 0F9H	SCREEN DUMP OF 64 DOT-LINE	Takes a dump of 1 to 64 vertical dot lines (including the guide line) on the LCD screen.	This function only accepts the sequence but carries out no actual dump. (Actual dump is initiated when the user later sets the screen to mode 4 with ESC + 0D0H.)	

Note: Starting and ending positions in the ESC Ø6CH (Dot Line Write sequence)

ESC+Ø6CH draws a line across the specified two points assuming that the LCD screen exists at the coordinates shown below. Values are taken to be negative when MSB is 1. Negative values are represented in two's complement form.

Graphics coordinates

(0,0)   (479,0)
                         Horizontal
                              +
            LCD

(0,63)   (479,63)

+ Vertical

Text mode Coordinates

(1,1)   (80,1)
                         Horizontal
                              +
            LCD

(1,8)   (80,8)

+ Vertical

6-64
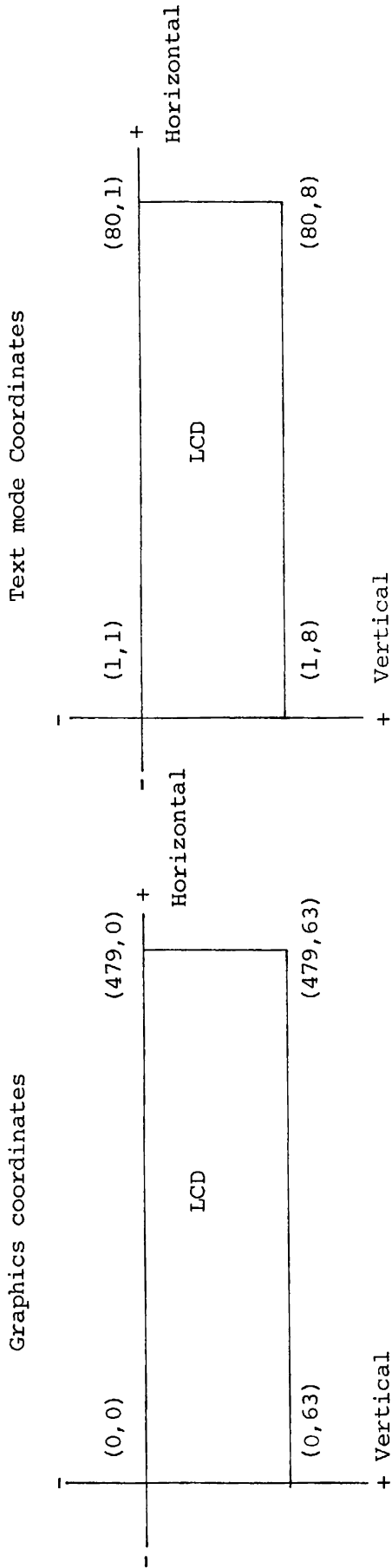
The arrow keys (including the shift keys) may be set to any codes using the above listed ESC sequences. Some special codes are used to control the screen directly. The codes and functions are listed below.

CODE	FUNCTION	DEFAULT KEY
00H . . 1BH	See the CONOUT Specifications	
1CH	CURSOR RIGHT	→
1DH	CURSOR LEFT	←
1EH	CURSOR UP	↑
1FH	CURSOR DOWN	↓
20H . . 7FH	See the CONOUT Specifications	
80H	Does nothing.	SHIFT/ → SHIFT/ ←
81H . . F7H	See the CONOUT Specifications	
F8H	Scroll Up One Line	SHIFT/ ↑
F9H	Scroll Down One Line	SHIFT/ ↓
FAH	Page Up	CTRL/ ↑
FBH	Page Down	CTRL/ ↓
FCH	TOP OF SCREEN	
FDH	BOTTOM OF SCREEN	
FEH	Switch To VS1	CTRL/ ←
FFH	Switch To VS2	CTRL/ →

Keys are disabled when they are set to 80H. When a key is set to one of the codes F8H through FFH, it is disabled and the screen can be controlled directly by the user.

ESC Sequence Parameter Table

The table below lists the ESC sequences for which the MAPLE does
nothing and discards any parameters.

(ESC sequences and the number of parameters that are ignored)

NUMBER	CODE	KANA MODE	KANJI MODE	REMARKS
1	ESC "%"	-	1	Access CG ROM directly
2	ESC "C"	1	1	Set character
3	ESC "L"	2	2	Change CRT Color In the new ASCII version (M2503ØCB) kana mode, the parameters are displayed as they are (garbage data).
4	ESC 80H	1	-	Code conversion In the new ASCII version (M2503ØCB) kana mode, the parameters are displayed as they are (garbage data).
5	ESC 81H	1	-	Code conversion In the new ASCII version (M2503ØCB) kana mode, the parameters are displayed as they are (garbage data).
6	ESC 82H	2	-	Code conversion In the new ASCII version (M2503ØCB) kana mode, the parameters are displayed as they are (garbage data).
7	ESC 83H	2	-	JIS C6226 → SHIFT JIS
8	ESC 84H	2	-	SHIFT JIS → JIS C6226
9	ESC 90H	-	2	Partial Scroll Up
10	ESC 91H	-	2	Partial Scroll Down
11	ESC 92H	1	-	Scroll Right In the new ASCII version (M2503ØCB) kana mode, the parameters are displayed as they are (garbage data).

NUMBER	CODE	KANA MODE	KANJI MODE	REMARKS
12	ESC 93H	1	-	Scroll Left In the new ASCII version (M2503CB) kana mode, the parameters are displayed as they are (garbage data).
13	ESC 95H	-	1	Set Scroll Mode
14	ESC 0C2H	5	-	Horizontal Line Write In the new ASCII version (M2503CB) kana mode, the parameters are displayed as they are (garbage data).
15	ESC 0C3H	5	-	Horizontal Line Erase In the new ASCII version (M2503CB) kana mode, the parameters are displayed as they are (garbage data).
16	ESC 0C4H	4	-	Vertical Line Write In the new ASCII version (M2503CB) kana mode, the parameters are displayed as they are (garbage data).
17	ESC 0C5H	4	-	Vertical Line Erase In the new ASCII version (M2503CB) kana mode, the parameters are displayed as they are (garbage data).
18	ESC 0CBH	1	-	Ruler Line Mode On/Off
19	ESC 0CCH	1	-	Set/Reset Guide Display
20	ESC 0CEH	1	-	Change Convert Mode
21	ESC 0D1H	-	1	Switch between VS1 and VS2
22	ESC 0D3H	-	1	Display Function Key
23	ESC 0D6H	-	1	Cursor Type
24	ESC 0E0H	-	9	User Defined Character

Note: "-" indicates that the MAPLE takes some action.

Chapter 7 System Functions

The MAPLE provides the following six system functions in
addition to the standard CP/M functions:

    1) Password

    2) Auto Start String

    3) Menu

    4) Resident

    5) System Display

    6) Auto Power Off

This chapter describes the six system functions.

7.1 Password

The operation of and specifications for the Password
function are described in "OS Specifications". This
section describes how to set or cancel a password in
an application program. The contents of the password
specified in the following work area is held intact
until the next system initialization:

- PASFLG: Overseas version = 0F01DH

  Japanese-language version = 0ED1DH

  This flag indicates whether a password is
  defined or not.

  = 00H: No password defined.

  ≒ 00H: Password defined.

- PASWRD: Overseas version = 0F01EH

  Japanese-language version = 0ED1EH

  This 8-byte area is loaded with the password in the
  complemented form.

Note: When the password is canceled with the PASFLG set
      to 00H, all of the eight bytes starting at the
      PASWRD must also be padded with "?" marks. This is
      because MTOS will copy this password onto tape.

7.2 Auto Start String

The Auto Start String function loads a predefined auto
start string into the key buffer when a warm boot is
initiated by a power on and processes it as if it were
entered from the keyboard. This function is useful for
users who wish to run a specific program every time they
start their system or those who want to use the MAPLE as
a turn-key system.

This section shows how to define and cancel an auto
start string in an application program. Refer to "OS
specifications" for the operation of and specifications
for the Auto Start String function.

- AUTOSTRT: Overseas version = 0F3D6H

       Japanese-language version = 0F14BH

      The auto start string is loaded in the buffer

      area at the above location in the following

      format:
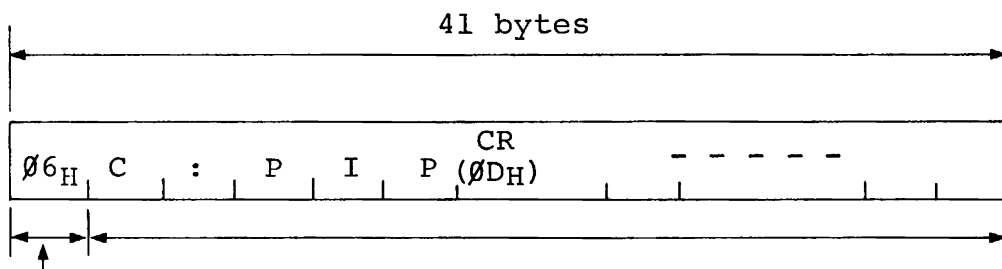
```
               41 bytes
 |<--------------------------------------->|
 |                                         |
 | 06H  C  :  P  I  P  CR          - - - - |
 |                    (0DH)                |
 |<>|<------------------------------------>|
   |
   |    A string of up to 40 bytes can be defined
   |    beginning at the second byte.
   |
   |__Indicates the length of the specified string
        (00H - 28H).  00H indicates that no auto start
        string is defined.
```

An auto start string may consist of up to 40 bytes including control codes if the string is to be defined directly in the work area. The length of the actual string, however, must be shorter than 40 bytes if control codes are included because a control code is displayed by two characters on the system display.

The contents of in the work area is cleared during system initialization.

* Auto start string and menu display
In Overseas Version 1.0 and Japanese-language Version, the auto start string is treated in the same way as keyed in data on the menu screen command line. This necessitates the user to be aware of where the auto start string will work (on the menu screen or as a CCP command).

In Overseas Version 2.3 (B), on the other hand, if an auto start string is stored in the keyboard buffer, it always works as a CCP command, whether it is displayed on the menu screen or not. The user, therefore, need only define the auto start string as a CCP command.

The figure below shows how WBOOT processes the auto start string.

```
                    Power SW turned on
                    in restart mode
                            │
─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─┼─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─
  WBOOT                     │
                         ╱──┴──╲
                        ╱ Auto  ╲        Y
                       ╱start str-╲───────────────┐
                       ╲ing defined╱              │
                        ╲   ?    ╱                │
                         ╲──┬──╱                  │
                          N │            ┌────────┴────────┐
                            │            │ Load auto       │
                            │            │ start string    │
                            │            │ into keyboard   │
                            │            │ buffer          │
                            │            └────────┬────────┘
                            │◄─────────────────────┘
                            │                     ┆
                            │                     ┆  Overseas version 2.3 (B)
                         ╱──┴──╲                  ┆
                        ╱ Menu  ╲      N          ┆
                       ╱specified?╲───────────────┤
                        ╲        ╱                ┆
                         ╲──┬──╱                  ┆
                          Y │                     ┆
                   ┌────────┴────────┐            ┆
                   ║ Display menu    ║            ┆
                   └──┬──────┬───────┘            ┆
          File        │      │  ESC key           ┆
  Select  selected    │      │  entered           ┆
  resident            │      │                    ┆
                 ┌────┴──────┴────┐               ┆
                 │ Load command   │               ┆
                 │ line data into │               ┆
                 │ keyboard buffer│               ┆
                 └────────┬───────┘               ┆
        │                 │              │        ┆
        ▼                 ▼              ▼        ▼
  Execute program                    To CCP
  at 1ØØH
```

7-5
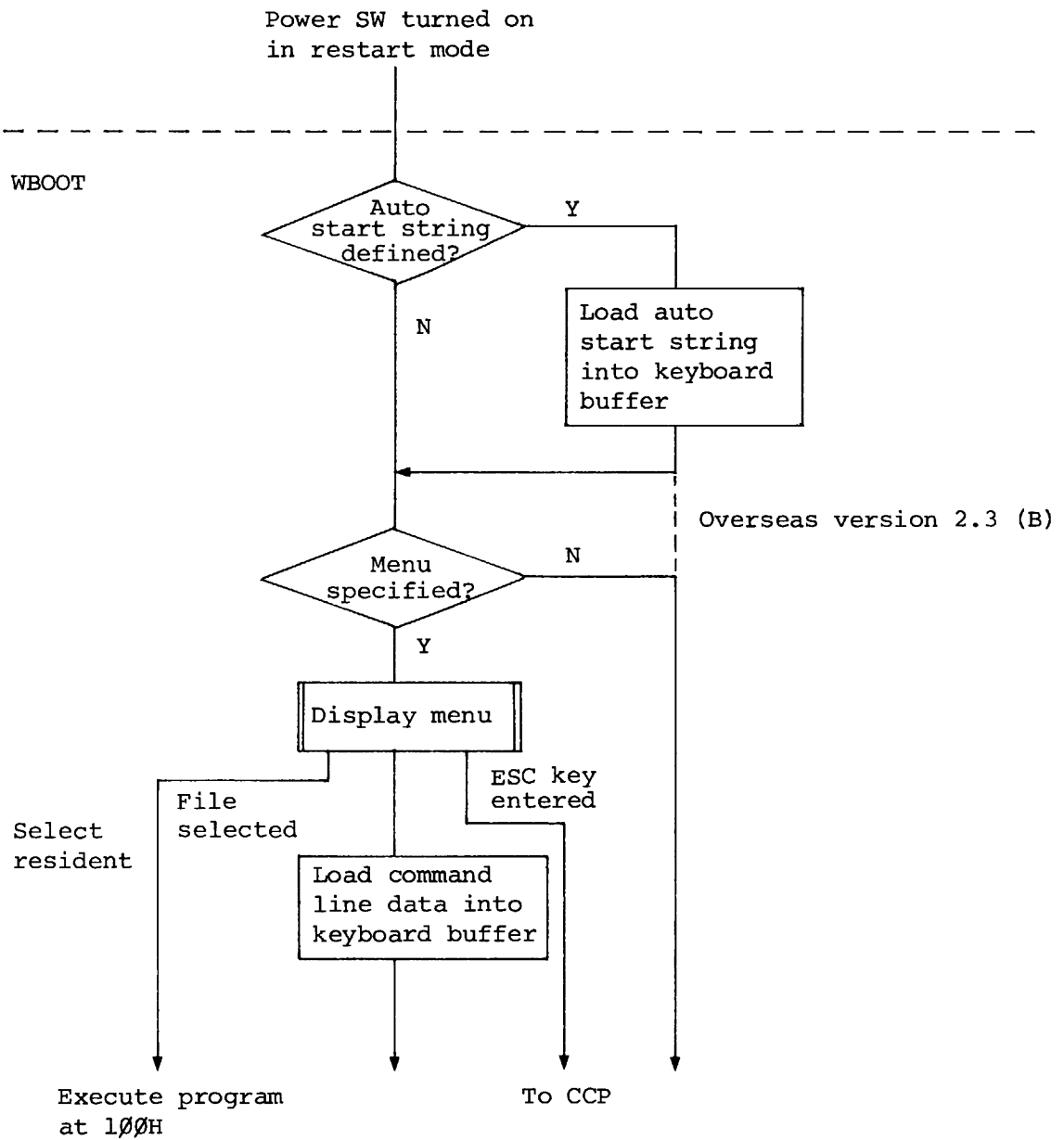
This section explains how to define and cancel a menu in an application program. Refer to "OS Specifications" for the operation of and specifications for the menu function.

- MENUFG: Overseas version = 0F02AH

    Japanese-language version = 0ED2AH

    This flag indicates whether a menu is to be displayed during WBOOT processing.

    = 00H: Menu displayed.

    ≠ 00H: No menu displayed.

    MENUFG is initialized to 00H.

- MENUDRV: Overseas version = 0F02BH

    Japanese-language version = 0ED2BH

    The 8-byte area starting at the above address is loaded with the ASCII codes corresponding to the drives of which the directory is to be loaded. Lowercase letters are converted to uppercase letters. The MAPLE supports drives A through H (A through I for Overseas version 2.3 (B)). Specify letters from A through H or I. If a letter is specified more than

once, the MAPLE will display the directory of
the corresponding drive the number of times
equal to the number of occurrence of the
letter. Any letters other than A through H
(or I) are ignored. This area is initialized
to "CBA_____" (ICBA_____" for Overseas
Version 2.2 (B)).

- FTYPETBL: Overseas version = 0F036H

    Japanese-language version = 0ED37H

    The 12-byte area starting at the above
    address is loaded with the file types of the
    files that are to be displayed in the menu.
    Specify up to four 3-character file types. A
    file type of three blanks is treated as
    undefined.

- FNAMETBL: Overseas version = 0F042H

    Japanese-language version = 0ED43H

    The 40-byte area starting at the above
    address is loaded with the file names (10
    characters including the drive name) of the
    COM files that are to be displayed in the
    menu. Specify up to four file names.
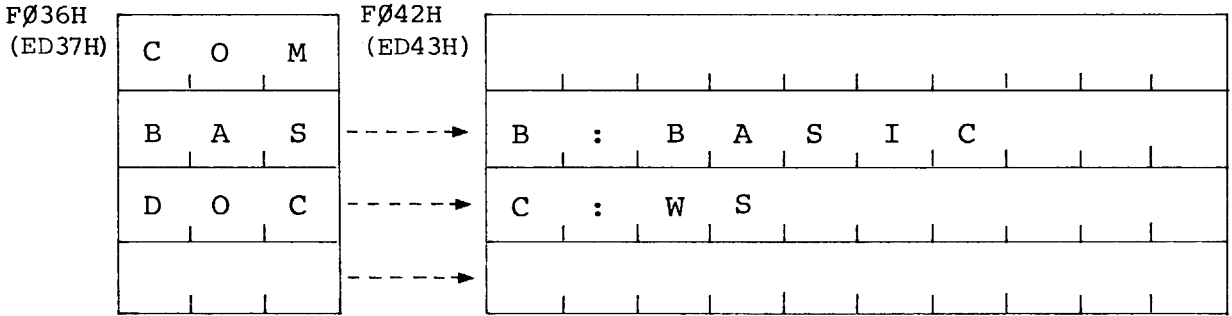
```
        FTYPETBL                      FNAMETBL

F036H ┌─────────────┐    F042H  ┌──────────────────────────────────┐
(ED37H)│ C    O    M │    (ED43H)│  │    │    │    │    │    │    │  │
       ├─────────────┤           ├──────────────────────────────────┤
       │ B    A    S │ - - - - ▶ │ B  :    B    A    S    I    C     │
       ├─────────────┤           ├──────────────────────────────────┤
       │ D    O    C │ - - - - ▶ │ C  :    W    S                    │
       ├─────────────┤           ├──────────────────────────────────┤
       │             │ - - - - ▶ │  │    │    │    │    │    │    │  │
       └─────────────┘           └──────────────────────────────────┘

             ⇓                                    ⇓

     FTYPETBL is                      Initialized to all blanks.
     initialized to
     "COM"; the
     remaining 9
     bytes are padded
     with blanks.
```

In the above example, the menu displays files which have a file type of either ".COM," ".BAS," or ".DOC." When the user select a file having a file type of ".BAS" with cursor keys, the message "B:BASIC___" is displayed at the beginning of the command line, followed by the name of the selected file. The menu function then waits for key entry. When a file with a file type of ".DOC" is selected, the message "C:WS_____" is displayed at the beginning of the command line. When a file with a file type of ".COM" is selected, no data is taken from FNAMETBL and only the specified file name is displayed on the command line starting at its beginning.

The contents of the above work areas can be set by menu file specification in System Display. The work areas are initialized during each system initialization and their contents are preserved until the next system initialization.

Only the BS edit function is effective on the command line.

If the address MTPAFG is loaded with a value other than
zero, the comment "(resident)" is displayed following
the first file (the file on the upper left position on
the first page) of the menu.
When this file is selected from the menu, the OS will load
no transient program but transfers control directly to
the program already in memory at address 100H. This
function is used to eliminate the time required to load
a program from a disk drive. The resident function is
enabled only when the menu is displayed. BASIC takes
advantage of this function.

- MTPAFG: Overseas version = 0F035H

        Japanese-language version = 0ED36H

        Indicates whether the resident function is to

        be enabled during menu processing.

         = 00H: Resident function disabled.

        ≒ 00H: Resident function enabled.

        MTPAFG is initialized to 00H.

        If the program to be executed at address 100H

        is programmed to set this work area to zero at

        its beginning, it displays the contents of the

MTPAM at the beginning of the menu when it
terminates execution with the WBOOT routine.
If the contents of the MTPAM is selected in
the menu, the program at address 100H starts
execution immediately.

This work area is automatically cleared to
zero by the OS when the menu function is
exited.

- MTPNM: Overseas version = 0F0B4H

Japanese-language version = 0F229H
This 9-byte area is loaded with the message to
be displayed at the beginning of the menu.
The message must be terminated by an 0FFH.
The user can specify not longer than 8
characters. This area must always be filled
when MTPAFG is set to 00H. BASIC assumes the
following message in MTPANM:

F4B4H
(F229H)

```
 ┌─────────────────────────────────┐
 │ B   A   S   I   C   FF_H         │
 └─────────────────────────────────┘
```
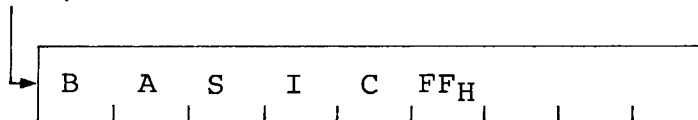
The following message will then be displayed
at the beginning of the menu:

```
┌─────────────────────────┐
│  BASIC   (resident)     │
└─────────────────────────┘
```

7.5 System Display

Refer to "OS Specifications" for the use of the system display function.

As explained in Chapter 5, "keyboard," this function is invoked as a subroutine from a BIOS keyboard routine when the HELP key is pressed while holding down the CTRL key. This means that if the application program executes neither BDOS nor BIOS keyboard routines (CONST, CONIN, etc.), this function will not be started whenever the CTRL/HELP keys are pressed.

The system display function provides the following subfunctions:

1) Password processing
2) Alarm/wake processing
3) Auto start string processing
4) Menu processing
5) MCT processing
6) Manual MCT processing

7.5.1 Password

The password system display function defines, cancels, or displays a password using the work areas described in 7.1.

7.5.2 Alarm/Wake

The alarm/wake system display function sets, resets, and displays the status of the alarm/wake function using the TIMDAT internal BIOS function. This system display function, however, has the following restrictions compared with the alarm/wake BIOS function:

(1) The time can be set only in minutes (the second field is set to 0). (The BIOS function allows the time to be set in 10-second units.)

(2) The day of the week is not supported (the day of the week field in the time descriptor is set to 0FFH). Only the month, the day, and the time are supported.

(3) Only WAKE1 is supported (specifying WAKE2 is invalid).

For further information, see Chapter 4, "BIOS Functions" and Chapter 8 "Alarm/Wake Function."

7.5.3 Auto Start String

The auto start string function defines, cancels, or displays an auto start string using the work areas described in 7.2.

7.5.4 Menu

The menu system display function defines, cancels, or displays a menu using the work areas described in 7.3.

7.5.5 MCT

The MCT system display function sets the following two modes:

- Stop/Nonstop mode
- Verify mode

1) Stop/nonstop mode

The MCT system display function specifies whether tape blocks are to be fed in the stop or nonstop mode when writing a file onto tape using the work areas described below. The mode used for writing a file on MCT tape is used when the file is read from the MCT tape (the mode, however, can be changed by rewriting the work areas).

See Chapter 14, "MTOS/MIOS Operations" for details on the stop/nonstop modes and related work areas.

- DFTATR: Overseas version = 0F2E0H

    Japanese-language version = 0F01DH

- TACATR: Overseas version = 0F78FH

    Japanese-language version = 0F70CH

    In both work areas:

        A 1 in bit 7 specifies the stop mode.

        A 0 in bit 7 specifies the nonstop mode.

    DFTATR is initialized to '11000010B'.

    Both work areas must be updated simultaneously when the mode is to be changed.

2) Verify mode

The MCT system display function also specifies whether the contents of the tape are to be verified after a write operation. When verify mode is specified, after closing the written file, the MTOS/MIOS rewinds the tape, reads the file blocks on the tape, and compares the CRC bytes with those in memory for each tape block. It does not verify data itself.

See Chapter 14, "MTOS/MIOS Operations" for details on the verify mode.

- VERFDFLT: Overseas version = 0F07BH

    Japanese-language version = 0F036H

- VERFFG: Overseas version = 0F07CH

    Japanese-language version = 0F737H

    The use of both work areas are the same:

    = 0FFH turns on the verify mode.

    ≠ 0FFH: turns off the verify mode.

    These work areas are initialized to 00H. They must be updated simultaneously when the mode is to be changed.

7.5.6 Manual MCT Operation

The system display function executes as follows when the user controls MCT operations with PF keys:

1) PF1 (FF)

Executes the internal MIOS function 06H (FF).

2) PF2 (PLAY)

Executes the following functions sequentially:

i) MIOS function 0BH (HEAD ON).

ii) MIOS function 04H (PLAY).

iii) 7805 command 72H (with 10000000B as parameter) to turn on the speaker.

3) PF3 (STOP)

Executes the following functions sequentially:

i)   7805 command 72H (with 00000000B as parameter) to turn off the speaker.

ii)   MIOS function 03H (STOP).

iii) 7805 command 0CH (HEAD OFF)

4) PF4 (REWIND)

Executes the MIOS function 08H (REWIND).

5) PF5 (RESET COUNTER)

Executes the MIOS function 02H to reset the tape counter
to 0.

6) PF6 (REMOVE)

Executes the MTOS function 252 (REMOVE).

7) PF7 (MOUNT)

Executes the MTOS function 253 (MOUNT).

8) PF8 (DIRINIT)

Executes the MTOS function 255 (MAKDIR).

9) PF9 (ERASE)

Executes the MIOS function 15H (ERASE) to erase the
tape.

Care must be taken with the following when operating the MCT manually:

1)   No manual operation on the MCT is allowed when a file on the MCT is open. Whether a file is opened or not can be identified by checking the following work area:

- OPNMOD: Overseas version = 0F361H

Japanese-language version = 0F0A8H

= 00H: No file is open.

= 01H: A file is opened in the read mode.

= 02H: A file is opened in the write mode.

2)   Operations other than remove are not allowed when a MCT is mounted but no files are opened. This is because any operation other than remove will affect the counter value.

3)   Any operation is allowed when no MCT is mounted.

Whether the MCT is in the mount or remove state can be identified by checking the following work area:

- TAPMOD: Overseas version = 0F2DDH

    Japanese-language version = 0F01AH

        = 00H: Remove state

        = 01H: Mount state

4) While the Z80 proceeds to the next instruction immediately after calling an MIOS function, the slave CPU continues to execute the MIOS function after it is invoked. Accordingly, the calling program, after calling an MIOS function, must monitor the state of the MCT and terminate the MIOS function at a necessary point. Terminating an MIOS function can also be done using an MIOS function (e.g., a function equivalent to PF3 (STOP)).

See Chapter 14, "MTOS/MIOS Operations" for details on the MTOS and MIOS functions and Chapter 13 "Slave CPU Operations" for detailed discussion of the slave CPU functions.

7.5.7 Other Information Displayed by System Display Function

1) Clock on first line

The system display function reads and displays the present time using the TIMDAT BIOS function.

2) Disk size on third line

The RAM disk size in bytes is stored in the following work area in binary format:

- YSIZERAM: Overseas version = 0F6A8H

        Japanese-language version = 0F42BH

3) User BIOS size on fourth line

The user BIOS size in 256-byte units is stored in the following work area in binary format:

- USERBIOS: Overseas version = 0F00BH

        Japanese-language version = 0ED0BH

4) Tape counter value on fourth line

The system display function reads the tape counter value using the MIOS function 01H (REDCT).

7.6 Auto Power Off

To save power, the MAPLE automatically turns its power
off in the continue mode if it receives no data from the
keyboard within a predetermined time while it waits for
keyed-in data with the CONIN BIOS function. When its
power switch is turned off and back on again, the MAPLE
resumes its operation at the point where it was waiting
for keyed-in data. This feature is called the auto
power off function.

The auto power off function and its interval can be
specified in the following work areas:

- ATSHUTOFF: Overseas version = 0F026H

        Japanese-language version = 0ED26H

        Contains (in binary form) the interval in

        minutes between the time the last key is

        pressed and the time the power automatically

        turns off. A 00H in this area disables the

        auto power off function. The initial

        (default) value is 0AH (10 minutes).

- ATSOTIME: Overseas version = 0F027H

        Japanese-language version = 0ED27H

        The 2-byte area which contains the interval in

        seconds between the time the last key is

pressed and the time the power automatically
turns off. It has the following relationship
with ATSHUTOFF:

ATSOTIME = ATSHUTOFF x 60

The initial (default) value is 258H (600
seconds).

When the auto power off interval is to be altered, both
fields must always be changed simultaneously.

The auto power off function may not be available for
application programs which do not use CONIN for
receiving console input but perform console input in
their own way polling the keyboard port with CONST. To
make the auto power off function available for such
programs, add the routines described below to the
application program and have it execute the auto power
off function by itself.
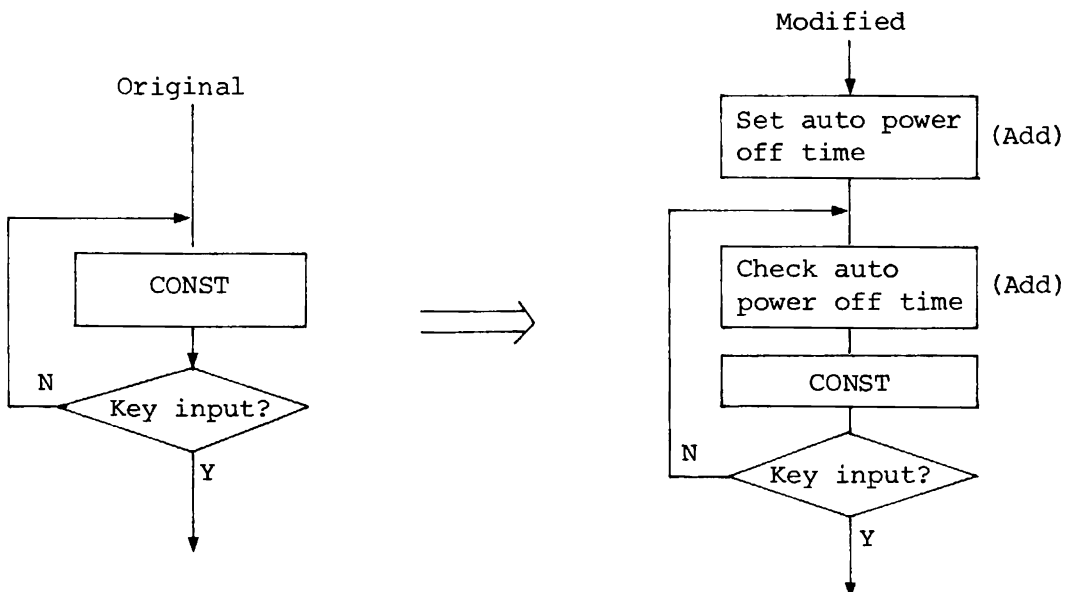
1) Routine for setting the auto power off time

```
        LD   HL,(ATSOTIME)      ; Get auto power off
                                  interval (in seconds)

        LD   DE,(TIMER∅)        ; Get current 1-second
                                  clock time

        ADD  HL,DE

        LD   (TIMEEND),HL       ; Set 1-clock timer value
                                  at which auto power off
                                  is to occur
```

2) Routine for checking the auto power off time

```
        LD   A,(ATSHUTOFF)      ; Check whether auto power
                                  off time is defined

        OR   A

        JR   Z,AAAA             ; Do nothing if not defined

        LD   HL,(TIMEEND)       ; Get 1-clock timer value
                                  for auto power off

        LD   DE,(TIMER∅)        ; Current 1-clock timer
                                  value

        SBC  HL,DE              ; Match?

        LD   C,∅                ; Load continue mode power
                                  off parameter

        CALL Z,POWEROFF         ; If match, call POWEROFF
                                  BIOS function for carry-
                                  ing out continue mode
                                  power off

        EI                     ; Execution resumes at EI
                                  instruction when power
                                  is turned on again.

  AAAA:
```

3) Work area descriptions

Label Name	Address	Size	Description
ATSHUTOFF	Overseas version = ØFØ26H Japanese-language version = ØED26H	1	Contains the auto power off time in minutes. A ØØH in this area disables the auto power off function.
ATSOTIME	Overseas version = ØFØ27H Japanese-language version = ØED27H	2	Contains the auto power off time in seconds. ATSOTIME has the following relationship with ATSHUTOFF: ATSOTIME = ATSHUTOFF x 6Ø
TIMERØ	Overseas version = ØFØ71H Japanese-language version = ØED72H	2	16-bit counter that is incremented by 1 every one second.
TIMEEND	Overseas version = ØF6DCH Japanese-language version = ØF46BH		Loaded with the time at which auto power off is to occur. Filled by the application program.

Chapter 8   Alarm/Wake Feature

8.1   General

The MAPLE is furnished with a 7508 4-bit CPU which
controls the software timer (clock) and generates
interrupts to the Z80 CPU at specified intervals.   The
software timer is supported by the alarm/wake OS feature.
The alarm/wake feature is divided into the following
three functions:

     1) Alarm function

     2) Wake1 function

     3) Wake2 function

These functions are identified by software using a flag;
only one type of interrupt is generated by the 7508 CPU
for these functions.   The 7508 checks for an alarm/wake
time every 10 seconds even if the MAPLE is in the power
off state, that is, the alarm/wake feature remains
available when MAPLE power is off.   However, the
alarm/wake processing differs depending on whether MAPLE
power is off or on.   The next section explains how
alarm/wake processing proceeds in both power off and on
states (refer to "OS Specifications" for details).

8.2 Alarm Function

1) What to set

   (1) Alarm time (month/day/hour/minute/second (10-
        second units))

   (2) Alarm message (up to 40 alphanumeric, kana,
        and graphics characters)

2) How to set

   (1) Use the System Display (second cannot be
        specified).

   (2) Use the BIOS TIMDAT function (see Chapter 4,
        "BIOS Calls").

   (3) Load the work areas time data and issue a time
        setting command directly to the 7508 CPU (see
        Section 8.7 and Chapter 11, "7508
        Explanations").

3) Alarm function in power-on state

The alarm function sounds an alarm and displays the time and message using the VRAM system screen. This guarantees that no user data on the screen be destroyed. When the display is ended, the user data displayed immediately before the alarm message is restored.

The time display can be terminated when:

    (1) The ESC key is pressed.

    (2) 50 seconds has elapsed.

    (3) The POWER switch is turned off.

    (4) A power failure occurs.

4) Alarm function in power-off state

The alarm function performs the same operations as in the power-on state after the MAPLE is powered on. After the display is terminated, the original screen before power is turned on is restored. If power is switched off and back on again while the alarm function is displaying the alarm time and message, then the normal power-on sequence occurs.

8.3 Wakel Function

1) What to set

    (1) Wake time (month/day/hour/minute/second (10-
        second units))

    (2) The name of program to be executed when a wake
        condition occurs.

2) How to set

    Same as the alarm function in 8.2.

3) Wakel function in power-on state

The wake function treats the wake string as an alarm
string and performs the same operations as the alarm
function.

4) Wakel function in power-off state

When power is turned on, the wake function loads the
wake string into the key buffer for execution as power-
on commands.

- When the MAPLE is in the restart mode power-off state

    The wakel function executes WBOOT and displays the

    Menu, then enables the wake string for execution

    under CCP control.

    (In Overseas version 2.3 (B), the function enables

    the wake string for execution under CCP without

    displaying the menu).

- When the MAPLE is in the continue mode power-off state

    The wakel function ignores the wake string and

    returns the MAPLE into the state before it is powered

    off and continues processing.

The wakel function, when used with the BIOS POWEROFF
function, may find many applications in periodic data
collection and other automatic (unattended) operations
without operator's intervention.

8.4 Wake2 Function

1) What to set

    (1)   Wake time (month/day/hour/minute/second (10-second units))

    (2)   The address of the routine to be executed when a wake condition occurs.

2) How to set

    (1)   Use the BIOS TIMDAT function (see Chapter 4, "BIOS Calls").

    (2)   Load the work areas time data and issue a time setting command directly to the 7508 CPU (see Section 8.7 and Chapter 11, "7508 Explanations").

3) Wake2 function in power-on state

The wake2 function calls the specified address. See programming note 5) below for the routine to be specified at this address.

4) Wake2 function in power-off state

- When the MAPLE is in the restart mode power-off state
  When power is turned on, the wake2 function calls
  the specified address, then returns the MAPLE into
  the state (restart mode) before power is turned
  off.

- When the MAPLE is in the continue mode power-off state
  The wake2 function returns the MAPLE into the state
  before it is powered off, then causes a jump to the
  specified address. If the destination of the jump
  is a RET instruction, control is returned to the
  point in the program at which the MAPLE was powered
  off in the continue mode.

5) Wake2 function programming notes

(1) Neither BDOS nor BIOS system call can be used in the routine to be called by the wake2 function.

(2) The routine to be executed by the wake2 function must end with a RET instruction.

(3) When the wake2 function is invoked in the power-off state, only power to the main board is turned on and no power is supplied to the I/O devices (e.g., RS-232C, serial port, and ROM capsules). Furthermore, if this condition occurs in the continue mode, the routine to be called by the wake2 function must turn on the power to these devices before executing the RET instruction. See the next page for the procedure for turning on the power to the I/O devices.

(4) The event which called the wake2 function can be identified by examining the following work areas:

- ZSTARTFG: Overseas version = 0F389H

          Japanese-language version = 0F0C9H

          Identifies the source of the invocation of the routine.

          01H: POWER switch on.

          02H: Alarm

          03H: Wake1

          04H: Wake2

- CNTNFG: Overseas version = 0F330H

    Japanese-language version = 0F050H

    Identifies the power-off state mode.

    = 00H: Continue mode.

    ≠ 00H: Restart mode.

ZSTARTFG	CNTNFG	State from which control is passed to the routine via wake2 function
≠04H	NC	Power-on state.
=04H	00H	Continue mode power-off state
=04H	≠00H	Restart mode power-off state

The routine called by the wake2 function must examine the above work areas to identify the power-off state and immediately set the work areas as follows:

    ZSTARTFG = 00H

    CNTNFG = 0FFH

When called in the continue mode power-off state, the routine must execute the following code before executing the RET instruction:

```
LD    HL,(ATSOTIME)  ⎤
LD    DE,(TIMER∅)    ⎥  Sets the new auto
ADD   HL,DE          ⎥  shut-off time.
LD    (TIMEEND),HL   ⎦


LD    A,(SPOPN)      ⎤
OR    A              ⎥  Checks to determine
JR    NZ,AAAA        ⎥  whether serial or
                     ⎥  RS-232C interface has
LD    A,(RSCLSF)     ⎥  been used and, if so,
OR    A              ⎥  turns on its driver.
JR    NZ,BBBB        ⎦


AAAA:                      ── Turns on the driver.
      LD    A,(CTLR2)        ↓
      SET   4,A
      OUT   (2),A
      CALL  ST1ML  ◄──────────── 1m-second software timer.
      SET   3,A
      OUT   (2),A
      CALL  ST1∅∅ML ◄─────────── 1∅∅m-second software
      RES   4,A                  timer.
      SET   5,A
      LD    (CTLR2),A
      OUT   (2),A
```

```
BBBB:
        LD      A,(RSCLSF)          ⌐
        OR      A                   │
        JR      NZ,CCCC             │

        LD      A,ØBEH              │
        OUT     (ØDH),A             │   Initializes 8251 if
        LD      A,Ø4ØH              │   the RS-232C interface
        OUT     (ØDH),A             │   has been used.
        CALL    ST1ØØML             │
        LD      A,(SVRSMOD)         │
        OUT     (ØDH),A             │
        LD      A,(SVRSCMD)         │
        OUT     (ØDH),A             ⌐


CCCC:
        XOR     A                       Turns off ROM capsule
        LD      (PROMPWR),A             power.



        LD      HL,MTIMEBUF
        LD      DE,YPOFDS
        LD      BC,4
        LDIR


        LD      A,(IER)
        OUT     (Ø4H),A


        LD      A,(CNTNILVL)
        LD      (INTLEVEL),A


        POP     HL                  ⌐
        POP     DE                  │
        POP     BC                  │   Restores registers.
        POP     AF                  │

        EX      AF,AF'              │
        EXX                         │
        POP     IY                  │
        POP     IX                  ⌐


        RET
```

Work area address chart

Work area name	Overseas version	Japanese-language version
ATSOTIME	FØ27H	ED27H
TIMERØ	FØ71H	ED72H
TIMEEND	F6DCH	F46BH
SPOPN	F35AH	FØ82H
RSCLSF	F2C8H	EFF8H
CTLR2	FØB2H	ED92H
SVRSMOD	F6DØH	F45DH
SVRSCMD	F6D1H	F45CH
PROMPWR	F1CAH	EEE3H
MTIMEBUF	F4BDH	F232H
YPOFDS	FØD9H	EDB9H
IER	FØB3H	ED93H
CNTNILVL	F385H	FØC5H
INTLEVEL	FØBAH	ED9AH

8.5 Alarm/Wake Function Processing Flow

1) Alarm/Wake processing in the power-off state

Alarm/wake interrupt generated by 75Ø8
(DI state).

75Ø8 automatically turns on main power.

```
┌─────────────────┐
│ ALRMFG ← 8ØH    │
│ LARMCT ← ØAH    │
│ ALRMST ← Ø1H    │
└─────────────────┘
        │
┌─────────────────┐
│ Call ALMHKO     │
└─────────────────┘
        │
┌─────────────────┐
│ ZSTARIFG        │
│ ALRMTP+1        │
└─────────────────┘
        │
      ◇ ALRMTP ◇
```

WAKE1 ──── 02H   03H ──── WAKE2

Ø1H ALARM

| Set MODEFG bit Ø on |
| EI |
| Display alarm message |
| DI |

◇ CNTNFG ◇ =ØØH

≠ØØH

| Call specified address |

| RESET SP to continue mode SP value |

◇ PW SW switched from off to on during display? ◇ Y / N

| Call ALMHK1 |

Normal power-on
processing
(- Continues
processing if in
continue mode)

Return to state
before power-off

Jump to specified
address (continue
processing with
RET instruction).

8-13

2) Alarm/Wake processing in the power-on state

Alarm/wake interrupt generated by 75Ø8
(EI state but interrupts from 75Ø8 are
disabled).

```
                    ◇ MODEFG ◇ ── on ──→   ┌ Terminates processing if
                      bit 0,1              └ alarm message is being displayed.
                         │ off
                    ◇ ALRMFG ◇ ── N ──→    ┌ Does nothing for second and
                      = ØØH                 │ subsequent interrupts (1Ø in
                         │ Y                └ total).
                ┌─────────────────┐
                │ ALRMFG ← 8ØH    │
                │ ALRMCT ← ØAH    │
                │ ALRMST ← Ø1H    │
                └─────────────────┘
                ┌─────────────────┐
                │ Call            │
                │ ALMHK 2         │
                └─────────────────┘
                ┌─────────────────┐
                │ YALMST          │
                │     YALMDS      │
                └─────────────────┘
                    ◇ YALMDS ◇ ── N ──→    ┌ Displays nothing if alarm message
                      = ØØH                └ is flagged off (by software).
                         │ Y
                ┌─────────────────┐         ┌ Enables interrupts because 75Ø8
                │ Enable 75Ø8     │ ─────   │ will specify the termination of
                │ interrupts      │         └ display of the alarm message.
                └─────────────────┘
                ┌─────────────────┐
                │ Display alarm   │
                │ message         │
                └─────────────────┘
                ┌─────────────────┐
                │ Disables 75Ø8   │
                │ interrupt       │
                └─────────────────┘
                ┌─────────────────┐
                │ Call            │
                │ ALMHK 3         │
                └─────────────────┘
                ┌─────────────────┐
                │ Check for power-off │
                │ state during alarm  │
                │ message display     │
                └─────────────────┘
                         │
                    Interrupt
                    postprocessing
```

3) Alarm display processing

```
                        ┌──────────────┐
                        │      EN      │
                        └──────┬───────┘
                               │
                        ┌──────┴───────┐
                        │ YPOFDS←bit5 on│
                        │ YALMST←ØØH    │
                        │ MODEFG←bit1 on│
                        └──────┬───────┘
                               │
  WAKE 2        =Ø3H    ┌──────┴───────┐
 ◄─────────────────────│   ALRMTP     │
                        └──────┬───────┘
                          =Ø3H │        Alarm or wake 1
 ┌──────────────┐       ┌──────┴───────┐
 │Call specified│       │Display alarm │
 │address       │       │screen using  │
 └──────┬───────┘       │system screen │
        │               └──────┬───────┘
        │                      │
        │               ┌──────┴───────┐
        │               │  Save        │
        │               │  key-related │
        │               │  work areas  │
        │               └──────┬───────┘
        │                      │
        │        N      ┌──────┴───────┐
        │      ◄────────│ Alarm dis-   │   - ESC key pressed.
        │               │ play termin- │   - 5Ø seconds has elapsed.
        │               │ ation        │   - POWER switch set on to off.
        │               │ condition?   │   - Power failure.
        │               └──────┬───────┘
        │                    Y │
        └──────────────────────┤
                        ┌──────┴───────┐
                        │   Call       │
                        │   ALMHK 4    │
                        └──────┬───────┘
                        ┌──────┴───────┐
                        │ Restore key- │
                        │ related work │
                        │ area         │
                        └──────┬───────┘
                        ┌──────┴───────┐
                        │ Restore screen│
                        │ into original │
                        │ state         │
                        └──────┬───────┘
                        ┌──────┴───────┐
                        │YPOFDS bit5 off│
                        │MODEFG bit1 off│
                        └──────┬───────┘
                        ┌──────┴───────┐
                        │      EX      │
                        └──────────────┘
```

4) Alarm processing during 1-second interrupt processing

Since alarm/wake interrupts occur every one second, a total of 10 times during the 10-second period, the interrupt handler ignores the second and subsequent interrupts. Accordingly, the OS examines the ALRMFG flag for 10 seconds (for 10 interrupts) since the first interrupt occurred using the 1-second interrupt processing routine and indicates the results to the alarm/wake processing routine.

```
                    │
                    ▼
             ╱ ALRMFG  ╲        N
            ╱  bit 7 on? ╲───────────┐
            ╲           ╱            │
             ╲         ╱             │
                  │ Y                │
                  ▼                  │
          ┌──────────────┐           │
          │  (ALRMCT) ←  │           │
          │  (ALRMCT)-1  │           │
          └──────────────┘           │
                  │                  │
                  ▼                  │
      N    ╱             ╲           │
      ◄────╱  ALRMCT = Ø  ╲          │
           ╲             ╱           │
                  │ Y                │
                  ▼                  │
          ┌──────────────┐           │
          │  Set ALRMFG  │           │
          │  bit 7 off   │           │
          └──────────────┘           │
                  │                  │
                  ├──────────────────┘
                  ▼
```
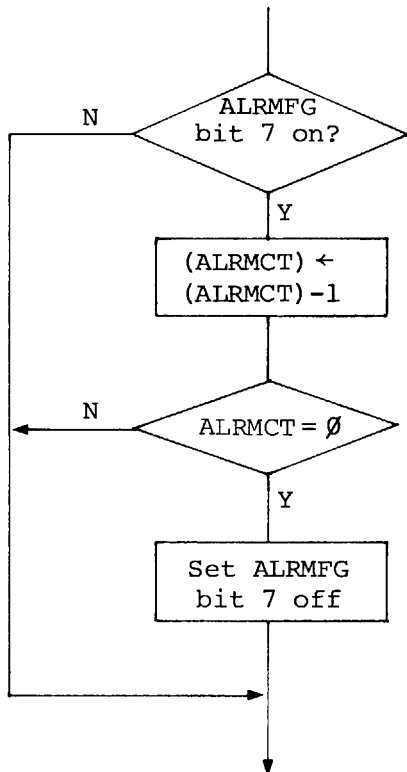
The first alarm/wake interrupt sets ALRMFG and ALRMCT flags as follows:

ALRMFG = 8ØH

ALRMCT = ØAH

After 1Ø seconds, ALRMFG is set to ØØH. The interrupt handler ignores any alarm/wake interrupts while ALRMFG is nonzero.

Summary of work areas used by the alarm/wake functions

Work area name (Address)	Size (in bytes)	Description
ALRMTP (Overseas version = ØFØ6CH, Japanese-language version = ØED6DH)	1	Identifies the type of alarm/wake functions. = ØØH: Undefined = Ø1H: Alarm = Ø2H: Wake1 = Ø3H: Wake2
ALRMAD (Overseas version = ØFØ6DH, Japanese-language version = ØED6EH)	2	Contains the starting address of the alarm message or wake string.
ALRMST (Overseas version = ØFØ6FH, Japanese-language version = ØED7ØH)	1	Indicates whether an interrupt occurred or not for the currently set alarm/wake time. = ØØH: Not occurred. = ØH: Occurred.
ALRMFG (Overseas version = ØFØ7ØH, Japanese-language version = ØED71H)	1	Indicates the time count from the timer when an alarm/wake interrupt occurred (up to 1Ø seconds). Bit 7: Set by the first interrupt and cleared after 1Ø seconds.
MODEFG (Overseas version = ØFØB8H, Japanese-language version = ØED98H)	1	Indicates the current system status. Bit Ø = 1: Alarm/wake processing invoked from power-off state is in progress. Bit 1 = 1: Alarm/wake processing invoked from power-on state is in progress.

Work area name (Address)	Size (in bytes)	Description
YALMDS (Overseas version = ØFØDBH, Japanese-language version = ØEDBBH)	1	Indicates the alarm/wake disable state. Bit 7 = 1: Disabled because BIOS is in execution. Bit 6 = 1: Disabled because password is being entered. Bit 5 = 1: Disabled because alarm/wake message is being displayed. Bit 4 = 1: Disabled because system message is being displayed. Bit 3 = 1: Disabled by BASIC. Bit 2 = 1: Disabled by scheduler. Bit 1 = 1: Disabled by MTOS. Bit Ø = 1: Reserved (for applications).
YALMST (Overseas version = ØFØDCH, Japanese-language version = ØEDBCH)	1	Indicates that an alarm/wake interrupt occurred when the alarm/wake functions are disabled. The meanings of the bits are identical to those of YALMDS.
ALRMCT (Overseas version = ØF4E6H, Japanese-language version = ØF25BH)	1	Contains the number of alarm/wake interrupts. The 75Ø8 generates an interrupt every one second for 1Ø seconds (1Ø in total) for one alarm/wake time.

8.6 How to Augment the Alarm/Wake Functions Using Hooks

As shown in Section 8.5, the alarm/wake functions has five hooks. The user can extend the alarm/wake functions by making patches in these hooks. This section shows how to make patches for these hooks.

Hook addresses

		Address	Label	Contents
Overseas Ver.		ØEF8CH	ALMHKØ:	JP RETURN
Japanese Ver.		ØEBD8H		
Overseas Ver.		ØEF8FH	ALMHK1:	JP RETURN
Japanese Ver.		ØEBDBH		
Overseas Ver.		ØEF92H	ALMHK2:	JP RETURN
Japanese Ver.		ØEBDEH		
Overseas Ver.		ØEF95H	ALMHK3:	JP RETURN
Japanese Ver.		ØEBE1H		
Overseas Ver.		ØEF98H	ALMHK4:	JP RETURN
Japanese Ver.		ØEBE4H		
Overseas Ver.		ØEEB7H	RETURN:	RET
Japanese Ver.		ØEBØBH		

The above entries can be hooked to any user-supplied routines by changing the address portion of the JP RETURN instruction.

Programming notes that the user must take when changing hook addresses follow.

(1) Take care with bank control.

The hook entries are always called when the system is in the system bank state (addresses 0000H through 7FFFH are allocated for ROM and 8000H through 0FFFFH for RAM). No problem will occur as long as the jump addresses in the hook table point to memory addresses 8000H and higher; however, the active bank need be switched to the user bank whenever hook entries are entered if they point to addresses between 0000H and 7FFFH. Normally no user-supplied routine can be placed in addresses between 0000H and 7FFFH.

(2) Reserve a user stack area.

Since control is transferred to the hook with the stack pointer pointing to the stack for interrupt routines, if the routine pointed to by the hook is to use a stack area (e.g., when using CALL and/or PUSH instructions), it must reserve its own stack area and restores the stack pointer to the original value when it terminates execution.

(3) Save the contents of registers and work areas.

Control is passed to the hook without saving the contents of registers and work areas. Accordingly, if a

user routine specified in the hook is to alter registers
or system work areas, it must save the contents of the
registers and work areas to be changed on entry and
restores them on exit (of course, it can safely alter
the contents of work areas which expect alteration by
the user).

(4) Do not change the interrupt status.
Since ALMHK0-ALMHK4 are invoked when the CPU is in one
of the interrupt states listed below, no user-supplied
routine specified in the hook can change the interrupt
state.   If a user-supplied routine need to change the
interrupt state, it must restore the MS into the
original interrupt state before terminating processing.

     ALMHK0:   DI state

     ALMHK1:   DI state

     ALMHK2:   EI state (7508 interrupts are disabled.)

     ALMHK3:   EI state (7508 interrupts are disabled.)

     ALMHK4:   EI state

(5) Disable interrupts when changing an address in the hook.
The system is highly likely to crash if an interrupt
using a hook entry occurs while the address in that
entry is being changed.   Since alteration of jump
addresses in the hook is normally done by the user
program in the TPA, the user program can and should

inhibit such interrupts with a DI instruction to avoid possible system crash. The program, however, must executes an EI instruction after terminating its execution.

(6) Do not call any system routine from the hook. The hook does not know from what system state it is called because it is invoked by interrupts. It may be called while a system routine (BDOS, BIOS, or internal OS routine) is being executed. A system crash will occur if a routine in the hook calls a system routine in such a situation.

(7) Do not perform an I/O operation. For the same reason given above, the routines in the hook must not perform any I/O operations such as display on the screen, communication through the RS-232C interface, etc.

(8) Since the jump table in the hook is initialized by system initialize or reset processing (placed into the state described on page 8-19), when the hook routines are to be made resident in memory, write to that effect in the manual. After system initialize or reset processing is performed, run a program for setting up the hook jump table. (Reset processing initializes only the hook jump table and keeps the user BIOS and RAM disk contents

intact.)

(9) Generally, the routines to be executed in the hook should be placed in the user BIOS area. This makes them resident in memory and solves the problem discussed in (1).

As explained in Sections 8.2, 8.3, and 8.4, alarm/wake
settings can easily be made by means of System Display
or BIOS calls. When alarm/wake settings are to be made
in interrupt processing routines as scheduled by a
scheduler, however, there is no way but to define
alarm/wake information directly to the 7508 CPU for the
reason given in paragraph (6) in 8.6.

The 7508 subsystem is provided with the following four
functions (commands) associated with the alarm/wake
feature:

- ALARM (WAKE) SET
- ALARM (WAKE) READ
- ALARM (WAKE) ON
- ALARM (WAKE) OFF

See Chapter 11, "7508 CPU" for details on the above
functions and the interface to the 7508.

This section describes the procedure for defining the
alarm, wake1, and wake2 information directly to the 7508
CPU.

1) Alarm setting procedure

(1) Disable interrupts from the 7508.

```
LD    A, (IER)
RES   0, A
OUT   (4), A
```

(2) Issue ALARM (WAKE) SET to the 7508 (to set the
alarm/wake time).

(3) Set up the work areas.

-    Load ALRMTP with 01H.

-    Load ALRMMSG (0F3FFH for overseas version and
     0F174H for Japanese-language version) with an
     alarm message in the following format:

41 bytes

4Ø bytes
(Actual message text is
loaded here)

Message length (Ø - 4Ø) in binary. A Ø indicates
that no message text is defined.
-    Load ALRMST with ØØH.
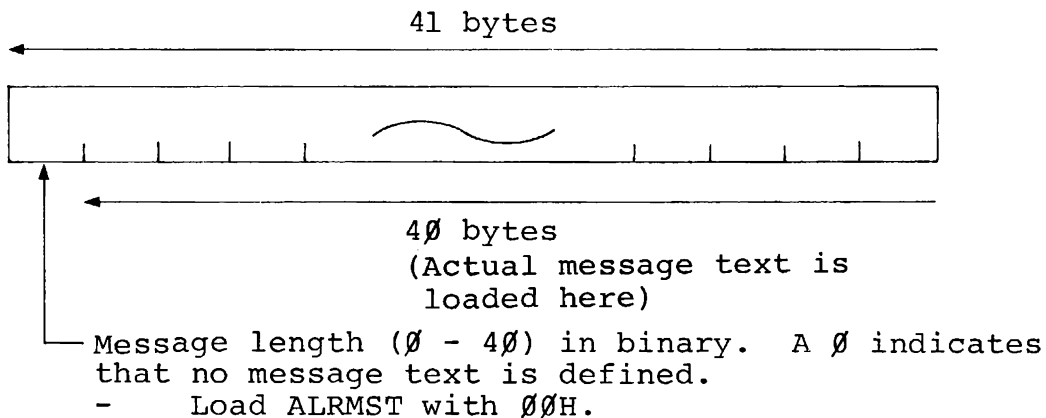
(4) Issue ALARM (WAKE) ON to the 7508 (to enable alarm/wake interrupt).

(5) Enables interrupts from the 7508.

    LD   A, (IER)

    OUT   (4), A

Take steps (1) through (5) in sequence.

2) Wake1 setting procedure

(1) Take the same steps as in alarm setting procedure except step 3):

(3) Set up the work areas.

- Load ALRMTP with 02H.
- Load ALRMMSG with a wake string in an appropriate format (a control code is counted as one byte).
- Load ALRMST with 00H.

3) Wake2 setting procedure

(1) Take the same steps as in alarm setting procedure except step 3):

(3) Set up the work areas.

- Load ALRMTP with 03H.
- Load ALRMAD with the address of the processing routine to be executed when a wake interrupt occurs.
- Load ALRMST with 00H.

Normal alarm processing displays an alarm message
immediately when an alarm interrupt occurs.
When displaying the alarm/wake message, it uses the speaker
and screen which are controlled by the slave CPU (63Ø1).
The slave CPU does many I/O operations in addition to
alarm/wake operation. If an alarm/wake interrupt occurs
while the slave CPU is performing an I/O operation and the
associated interrupt processing routine attempts to use
the slave CPU, the alarm/wake operation overlaps the
executing operation, causing a system hang-up.

The MAPLE takes the following measure to solve this
problem:
Since the slave CPU runs only when BIOS is performing an
I/O operation, the BIOS preprocessing routine sets a
flag on and the BIOS postprocessing routine resets that
flag. During the time this flag is on, the alarm/wake
interrupt processing routine displays no alarm message
when an alarm/wake interrupt occurs. It does nothing
but flags to indicate that an interrupt has occurred.
The BIOS postprocessing routine examines this flag to
see whether an alarm/wake interrupt has occurred and

displays an alarm/wake message if the flag is on. The
BIOS preprocessing and postprocessing routines are
called PREBIOS and PSTBIOS, respectively.

The flag indicating whether the alarm/wake message
is to be displayed or not is stored in the YALMDS work
area. The flag indicating that an alarm/wake interrupt
has occurred while the display of the alarm/wake message
is disabled is stored in the YALMST work area.

The figure on the next page shows the relationship
between PREBIOS, PSTBIOS, and BIOS, and the relationship
of YALMDS and YALMST to BIOS.
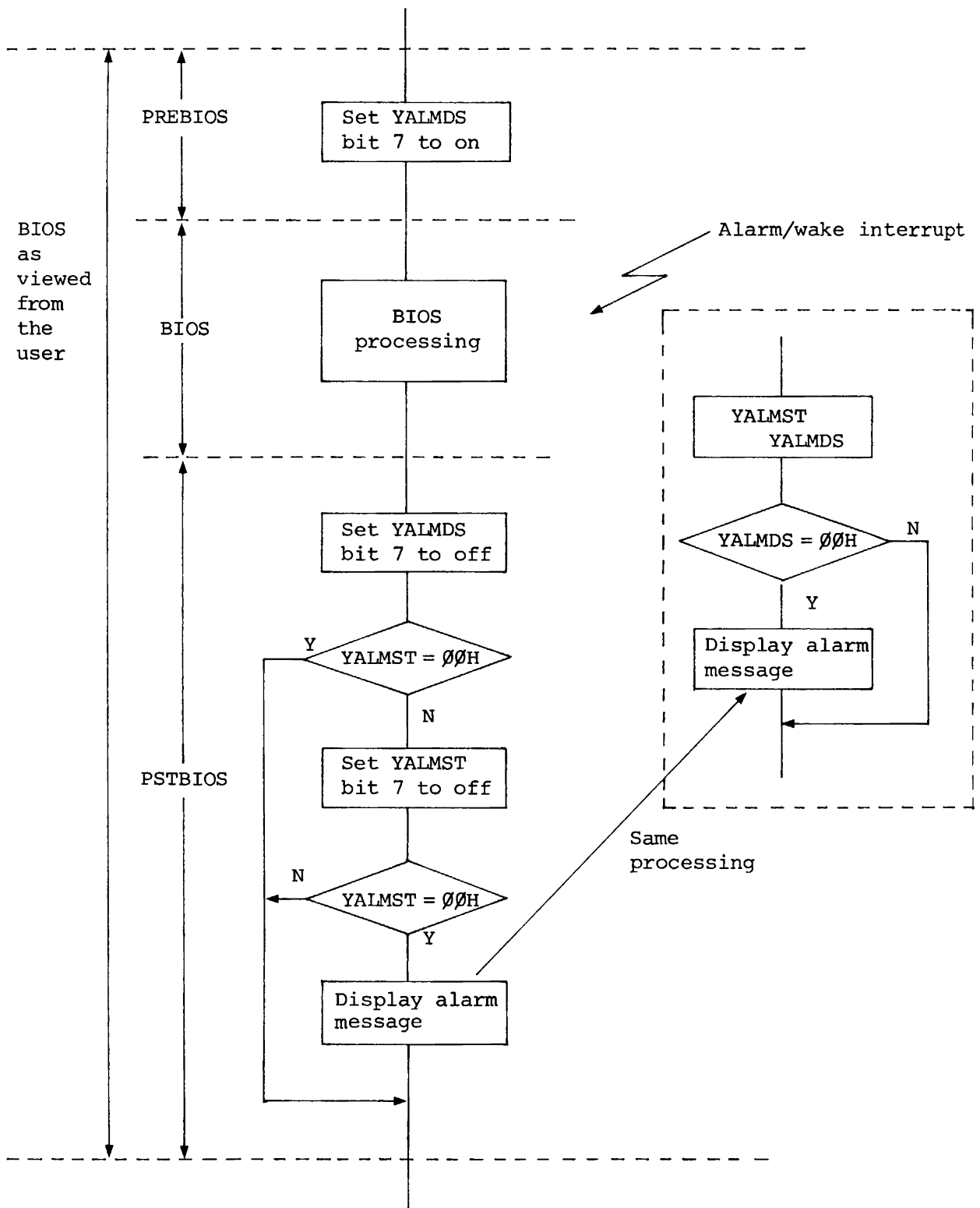
8.9 Method of Inhibiting Alarm Message Display from Application Program

Some application programs may not want the alarm message to be displayed during execution of some specific operations. The alarm operation can be disabled by executing the DI instruction or by inhibiting interrupts from the 7508 CPU. These measures, however, will also inhibit other interrupts (e.g., keyboard and power switch off interrupts). To avoid this, the application program can and must perform the same operations as PREBIOS and PSTBIOS do as explained in 8.8.

Display of the alarm message can be disabled using the YALMDS work area. As explained on page 8-18, YALMDS specifies what mode inhibits the alarm/wake interrupt. Bit 0 of YALMDS is reserved for application programs. The procedure on the next page shows how to inhibit alarm/wake operation from the application program.

When an alarm/wake interrupt occurs while alarm message display is disabled (YALMST contains a nonzero value), set only YALMST bit 7 to on (to make the system believe that the interrupt occurred during BIOS processing) and make a dummy call to BIOS. PSTBIOS will then check YALMST bit 7 and display an alarm message.

```
                            ┌──────────────┐
                            │ Set YALMDS   │
                            │ bit Ø to on  │
                            └──────────────┘
      - - - - - - - - - - - - - - -│
                         ▲         │
                         │         │
                         │         │
                         │         │
                         │         │
   Display of alarm message        │
   due to alarm/wake               │
   interrupt is disabled           │
   during this interval.           │
                         │         │
                         │         │
                         │         │
      - - - - - - - - - - - - - - -│
                            ┌──────────────┐
                            │ Set YALMDS   │
                            │ bit Ø to off │
                            └──────────────┘
                                   │
                             ◇ YALMST = ØØH ◇
                                   │
                                   │ N
                            ┌──────────────┐
                            │ Set YALMST to│
                            │ 8ØH (set only│        ┌──────────────┐
                            │ bit 7 to on  │        │   PREBIOS    │
                            │ and others to│        └──────────────┘
                            │ off)         │               │
                            ├──────────────┤        ┌──────────────┐
                            │ Output ESC   │        │ BIOS CONOUT  │
                            │ and "("      │        └──────────────┘
                            │ through BIOS │               │
                            │ CONOUT       │        ┌──────────────┐   ┌──────────────┐
                            └──────────────┘        │   PSTBIOS    │   │ Display alarm│
                                                    └──────────────┘   │ message      │
                                                                       └──────────────┘
```
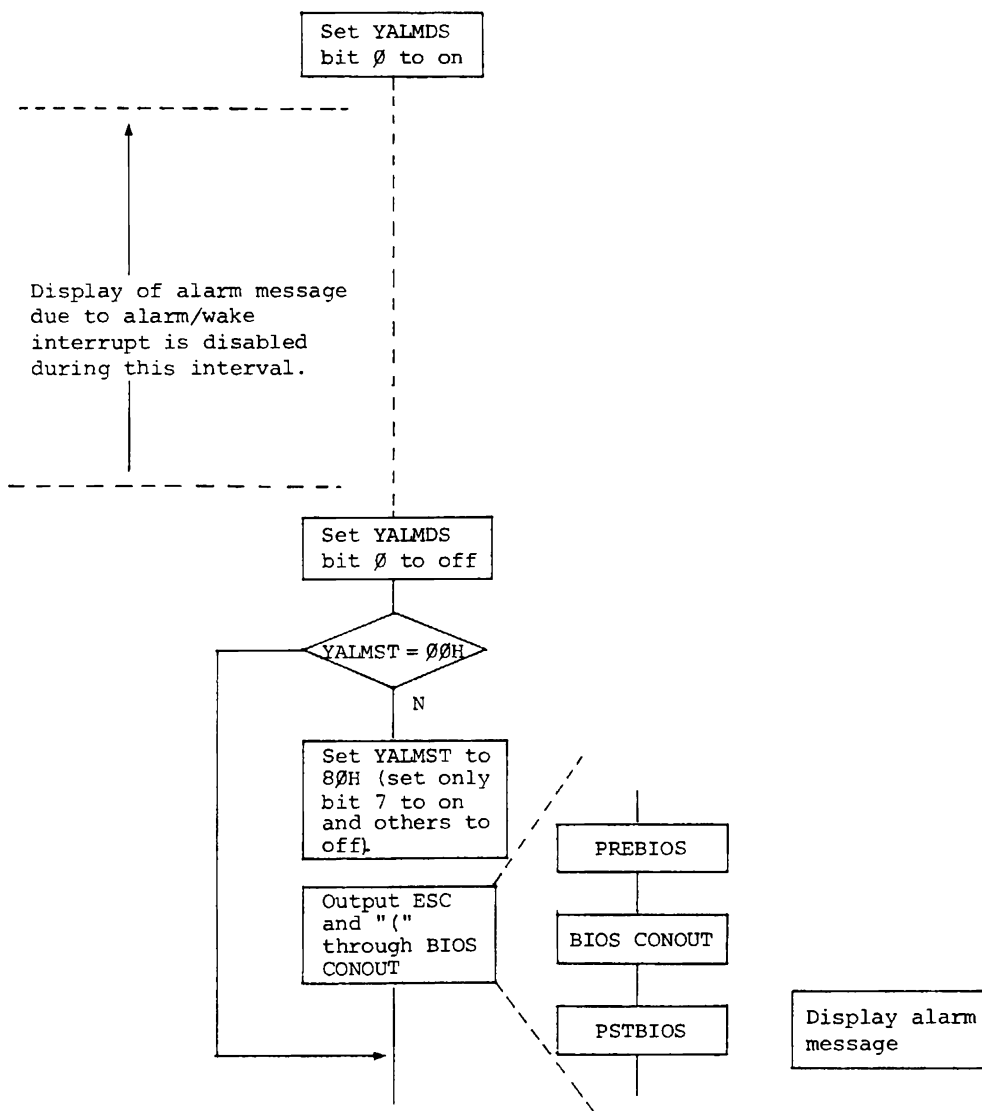
The BIOS CONOUT function only passes ESC and "(" ,but no operation results.

8.10 How to Disable System Display Function for Displaying Alarm/Wake Message

When an application program, e.g., scheduler, controls alarm/wake function, malfunctions will result if alarm/wake is set or reset from System Display. To avoid this, the MAPLE OS provides a work area for inhibiting the control of the alarm/wake functions through the System Display.

ALRMDS: Overseas version = 0F06BH

Japanese-language version = 0ED6CH

= 00H enables the control of the alarm/wake functions through the system display function.

≠ 00H disables the control of the alarm/wake functions through the system display function.

ALRMDS defaults to 00H.

ALRMDS is set to 00H by a system initialize.

8.11 Precautions on the Use of the Alarm/Wake Functions

(1) An alarm/wake interrupts are deferred up to 10 seconds in the power-off state. This is because the system checks the alarm/wake time only once every 10 seconds when the MAPLE is in the power-off state.

(2) Since display of the alarm message is inhibited while an BIOS operation is in progress as explained in 8.8, display of the alarm message will be put off accordingly. This should normally be negligible; however, it will be in the order of seconds if the MCT is running.

(3) The Overseas Version B allows the user to change the interval during which the alarm message is displayed (default is 50 seconds) in the range from 1 to 255 seconds.

ALRMPROD (0F2F9H): Load a number from 1-255. Do not specify 0 because the value 0 is interpreted as 0 second or 256 seconds.

Chapter 9   Power On/Off Function

There are two modes in the MAPLE power-off state:
restart and continue modes. In the restart mode, WBOOT
is executed after MAPLE power is turned on. In the
continue mode, processing that was executing when power
was turned off is resumed.

Switching to the power-on state from one of these modes
or switching to the power-off state from the power-on
state are normally carried out by turning on and off the
POWER switch.

The power to the MAPLE is controlled by the 7508 slave
CPU. Switching the POWER switch on or off is indicated
by the 7508 to the Z80 in the form of an interrupt. The
Z80 identifies the source of the power interrupt and take
necessary actions. (The MAPLE power can also be controlled
by the application program using the alarm/wake or BIOS
POWEROFF function.) A special power-off control is the
power failure interrupt which occurs when the battery
voltage falls below approximately 4.7 volts and places
the MAPLE into the continue mode power-off state.

This power failure interrupt is also controlled by the
7508 CPU.

The MAPLE is placed into the power-on state from the power-off state by:

(1)   Setting the POWER switch to ON.

(2)   Generating an alarm/wake interrupt.

The MAPLE power-on sequence is controlled by the 7508. The 7508 power-on sequence proceeds as follows:

(1)   Turns on Z80 power.

(2)   Set the active bank to the system bank (ROM in addresses 0000H through 7FFFH and RAM in addresses 8000H through 0FFFFH).

(3)   Starts the Z80 at address (ROM) 0H.

The processing starting at address 0H is called the "address 0 start" (or "STARTER"). Address 0 start is entered by the following two conditions in addition to the above two:

(3) By pressing the RESET switch (on the left side of the MAPLE).

(4) By pressing the 7508 RESET switch (on the bottom of the MAPLE).

The MAPLE power is turned on by the above four events or
conditions. The address 0 start processing routine can
identify the source of a power-on sequence by checking
the status returned from the 7508 CPU.
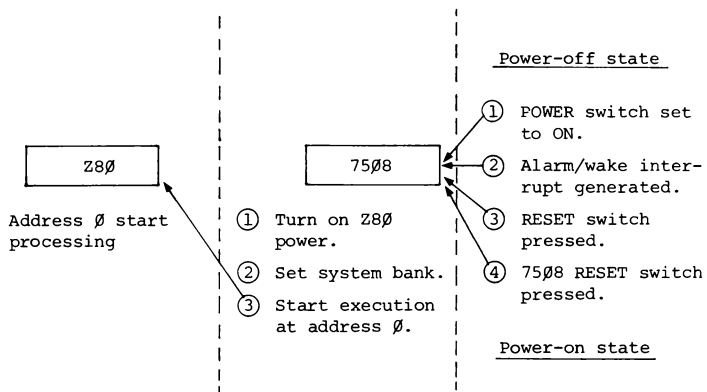
The operations of the address 0 start processing routine as invoked by the above four conditions are described below.

1) When the POWER switch is set to ON
- Initiates a restart mode power-on sequence (WBOOT).
- Initiates a continue mode power-on sequence (continues processing at the point where power was turned off).

2) When an alarm/wake interrupt occurs
- When the interrupt is generated by an alarm condition
    - Turns power off after displaying an alarm message.
    - Performs the same sequence as 1) after displaying an alarm message.
- When the interrupt is generated by a wake condition
    - Performs the same sequence as 1) if the wake1 function is invoked.
    - Calls or causes a jump to the specified address if the wake2 function is invoked.

3) When the RESET switch is pressed
- Causes a jump to CBOOT after carrying out reset processing.

4) When the 7508 RESET switch is pressed
- Performs the same sequence as 3) after carrying out system initialize processing.

```
                          ┌─────────────┐
                          │  Address Ø  │
                          │    start    │
                          └──────┬──────┘
                          ┌──────┴──────┐
                          │  Set Z8Ø into│
                          │  interrupt   │
                          │  mode 2      │
                          └──────┬──────┘
                          ┌──────┴──────┐
                          │  Set up     │
                          │  interrupt  │
                          │  vector     │
                          └──────┬──────┘
                          ┌──────┴──────┐
                          │  Read 75Ø8  │
                          │  status     │
                          └──────┬──────┘
```

| 75Ø8 RESET switch pressed | RESET switch pressed | POWER switch set to ON | Alarm/wake interrupt |

```
                   ◇ SHIFT/          ┌──────────┐        ┌──────────┐
          ◄──Y──── GRAPHICS key      │ 63Ø1 Hot │        │ 65Ø1 Hot │
                   pressed? ◄─        │  Start   │        │  Start   │
                      │               └────┬─────┘        └────┬─────┘
┌──────────┐          │ N            ┌─────┴─────┐       ┌─────┴─────┐
│ System   │    ┌─────┴──────┐       │ Check     │       │ Beep for  │
│initialize│    │ Check      │       │ password  │       │ alarm/wake│
│processing│    │ password   │       └─────┬─────┘       └─────┬─────┘
└────┬─────┘    └─────┬──────┘     ◇ Continue ◇──N──┐   WAKE1 ◇Identify◇ WAKE2
     │          ┌─────┴──────┐      mode? ◇         │    ◄───alarm or wake───►
     │          │ Reset      │         │ Y          │        │ ALARM
     └────────► │ processing │    ┌────┴────┐  ┌────┴────┐ ┌──┴────────┐  ┌──────────┐
                └─────┬──────┘    │ Beep in │  │ Beep in │ │ Display   │  │ Turn     │
                      │           │continue │  │ restart │ │ alarm     │  │ screen   │
                      │           │ mode    │  │ mode    │ │ message   │  │ display on│
                      │           └────┬────┘  └────┬────┘ └──┬────────┘  └────┬─────┘
                      │           ┌────┴────┐  ┌────┴────┐  ◇Power switch◇  ◇Continue◇─Y─┐
                      │           │Restore  │  │ Enable  │ Y turned on during  mode?      │
                      │           │RS-232C  │  │ 75Ø8    │ ◄─display?          │ N        │
                      │           │and serial│ │interrupts│     │          ┌───┴────┐    │
                      │           │interface │ └────┬────┘     │ N         │Execute │    │
                      │           │to previous│ ┌───┴────┐      │          │at      │    │
                      │           │state     │ │Initialize│     │          │specified│   │
                      │           └────┬────┘  │screen   │     │          │adress  │    │
                      │           ┌────┴────┐  └────┬────┘     │          └───┬────┘    │
                      │           │Turn     │  ┌────┴────┐      │              │         │
                      │           │screen   │  │Compute  │      │              │         │
                      │           │display on│ │RAM disk │      │              │         │
                      │           └────┬────┘  │check sum│      │              │         │
                      │           ┌────┴────┐  └────┬────┘      │              │         │
                      │           │Restore  │       │           │              │         │
                      │           │interrupt│       │           │              │         │
                      │           │mask to  │       │           │              │         │
                      │           │previous │       │           │              │         │
                      │           │state    │       │           │              │         │
                      │           └────┬────┘       │           │              │         │
                      │           ┌────┴────┐       │           │              │         │
                      │           │Restore  │       │           │              │         │
                      │           │SP to    │       │           │              │         │
                      │           │previous │       │           │              │         │
                      │           │value    │       │           │              │         │
                      │           └────┬────┘       │           │              │         │
                      │           ┌────┴────┐       │           │              │         │
                      │           │Restore  │       │           │              │         │
                      │           │registers│       │           │              │         │
                      │           │to       │       │           │              │         │
                      │           │previous │       │           │              │         │
                      │           │values   │       │           │              │         │
                      │           └────┬────┘       │           │              │         │
```

| To CBOOT (then WBOOT) | Continue processing before power was turned off | To WBOOT | To previous power-off state | To specified address |

9-5

9.2 Software-driven Power-on Sequence

As explained in 9.1, the MAPLE is normally powered on by two procedures. The first procedure (turning on the POWER switch) need be carried out manually. The latter procedure makes use of the alarm/wake feature which can be controlled by software. This feature (more exactly the wakel function) allows the user to control MAPLE power from this program with great ease.

See Chapter 8 "Alarm/Wake Feature" for how to set up the alarm/wake feature.

As seen from the flowchart on the preceding page, processing after power is turned on by the wakel function differs depending on which power-off state (i.e., restart or continue mode) the MAPLE was previously in. Since the wakel function is used in situations in which no human being attend the MAPLE (restart or continue mode power-on), adequate care must be taken when determining in which mode the MAPLE is to be powered off. See 9.3 for power-off sequences.

9.3 Power-off Sequence

The MAPLE enters the power-off state either in the restart or continue mode. This can be initiated by the following four methods:

Method	Initiated by	Controlled by	Power-off state
Setting POWER switch to OFF	Operator	Power SW on interrupt from 75Ø8	Restart mode
Setting POWER switch to OFF while holding CTRL key	Operator	Power SW on interrupt from 75Ø8	Continue mode
Power fail	OS	Power fail interrupt from 75Ø8	Continue mode
Auto Power Off	OS	CONIN BIOS function	Continue mode
POWEROFF BIOS function	Application	BIOS	Restart of continue mode

In either method, the Z80 CPU issues a power shut-down command to the 75Ø8 CPU to direct the 7508 CPU to turn Z8Ø power off. If there are any Z80 controlled I/O devices running, the Z80 CPU turns their power off before issuing this command.

The figure below outlines the flow of the power-off
sequence.
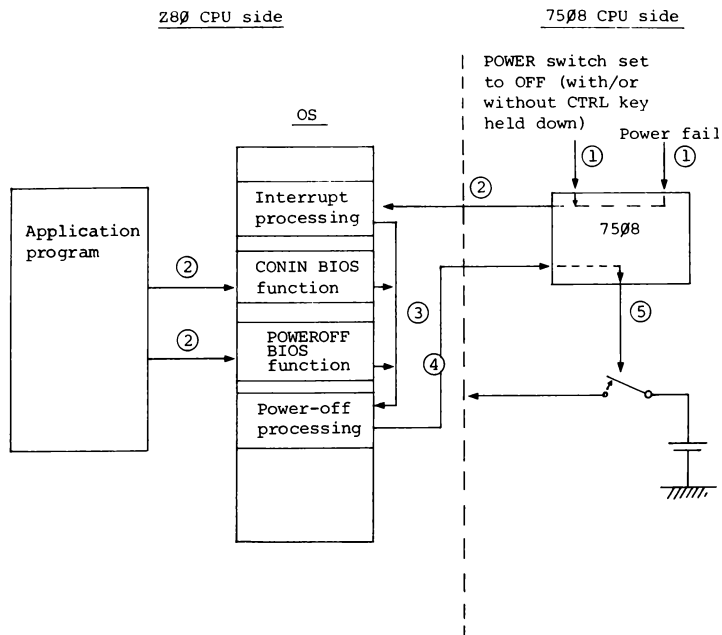
The power-off sequence proceeds in the order of steps
(1) through (4) to eventually turns main power off in
step (5).

The basic power-off sequence is the same in both restart and continue modes except flag set/reset processing. The OS examines the pertinent flag to identify the power-on mode (restart or continue) and takes preparatory actions accordingly for the next power-on sequence.

If one of the five power-off conditions occur while the OS is taking in a password or displaying an alarm message when the MAPLE is in the power-off state, the OS restores the MAPLE into the previous power-off state, irrespective of which power-off condition occurred.

Power-off conditions occurring while the system menu is being displayed always place the MAPLE into the restart mode. This is because it is likely that floppy disks or ROM capsules are replaced, changing the system environments.

When a power-off condition occurs while the system display function is in execution, the OS restores the MAPLE into the state before the system display function is executed, then carries out the required power-off sequence (restart or continue mode).

```
Set POWER switch                Power fail    Auto Power Off      POWEROFF BIOS function
   to OFF                                     (through CONIN
                                              BIOS function)

  ╱CTRL key?╲___off___
  ╲         ╱
      │on
                    │
Continue mode    Restart mode   Continue mode   Continue mode   Restart mode  Continue mode

─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─┬─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─

Power-off sequence        ┌──────────────┐
                          │      DI      │
    ⇓                     └──────────────┘
                          ┌──────────────┐
                          │ Flush buffer │
                          │ for external │
                          │ FD           │
                          └──────────────┘
                          ┌──────────────┐
                          │ Save         │
                          │ registers    │
                          └──────────────┘
                          ┌──────────────┐
                          │   Save SP    │
                          └──────────────┘
              =ØØH      ╱              ╲
            ◄───────── ╲ (FRCECNTN)   ╱
                        ╲            ╱
                             │=ØØH
               Y        ╱           ╲      N
            ◄────────── ╲ Continue  ╱──────────►
                         ╲  mode?  ╱
┌──────────────┐                        ┌──────────────┐
│ Set and clear│                        │ Set and clear│
│ contine mode │                        │ restart      │
│ flag         │                        │ flag         │
└──────────────┘                        └──────────────┘
                          ┌──────────────┐
                          │ Turn off ROM │
                          │ capsule power│
                          └──────────────┘
                          ┌──────────────┐
                          │ RS-232C,     │
                          │ Serial Power │
                          │ off          │
                          └──────────────┘
                           ╱          ╲   N
                           ╲ Power fail╱──────┐
                            ╲        ╱        │
                               │Y             │
                          ┌──────────────┐    │
                          │ Displays     │    │
                          │ "CHARGE      │    │
                          │ BATTERY"     │    │
                          └──────────────┘    │
                          ┌──────────────┐◄───┘
                          │ Turn off     │
                          │ screen       │
                          │ display      │
                          └──────────────┘
                          ┌──────────────┐
                          │ Turn off     │
                          │ loudspeaker  │
                          └──────────────┘
                          ┌──────────────┐
                          │ Stop MCT and │
                          │ turns off    │
                          │ MCT head     │
                          └──────────────┘
                          ┌──────────────┐
                          │  Halt 63Ø1   │
                          └──────────────┘
                          ┌──────────────┐
                          │ Direct 75Ø8  │
                          │ to turn 28Ø  │
                          │ power off    │
                          └──────────────┘
```

9.4 Power Fail Sequence

Since the MAPLE is driven by a battery, it may cause hardware malfunctions (CPU operation, memory read or write, etc.) if the battery voltage falls below a certain level. To avoid this, the 7508 CPU always monitors the battery voltage and, if the voltage falls below approximately 4.7 volts, it generates a power fail interrupt to the Z80 CPU and subsequently take the following actions:

(1) Generates power fail an interrupt to the Z80 CPU every one second.

(2) Switches from main (1100 mAH) to sub- (90 mAH) battery.

(3) If a power-off command is received from the Z80 CPU within 50 seconds after the first power fail interrupt request, turns off Z-80 power. The 7508 performs the normal power-on sequence when it receives the next power-on command. (The OS power fail processing is carried out in the continue mode.)

(4) If no power-off command is received from the Z80 CPU within 50 seconds after the first power fail interrupt request, the 7508 CPU forces Z80 power to off. When the 7508 CPU receives the next power-

on command, the 7508 performs the same power-on
sequence that is activated when the RESET switch is
pressed. Consequently, the system status at the
power fail time is cleared off (the RAM disk and
user BIOS areas are preserved).

When the OS accepts a power fail interrupt, it carries
out the power-off sequence as shown in the flow in 9.3.
During the power-off sequence, the OS displays the
message "CHARGE BATTERY" on the screen for 20 to 30
seconds and turns power off in the continue mode (before
the 7508 forces the power-off sequence).

Considerations on power fail interrupts are given below.
(1) The battery voltage level at which the 7508 CPU
senses a power fail condition can be changed (see
Chapter 11, "7508 CPU" for details). Notice that no
power failure can be sensed if the set voltage level is
too low.
(2) When an I/O device (especially MCT) is activated,
the battery voltage level may drop momentarily, causing
a power fail interrupt.
(3) If a DI instruction is executed in the application
program with no subsequent EI instruction or 7508

interrupts are disabled for an extended period, the
MAPLE can accept no power fail interrupt and probably
cannot turn off in the continue mode for the reason
given in paragraph (4) on the preceding page.

9.5 Software-activated Power-Off

As mentioned in 9.3, the application program can turns
MAPLE power off at appropriate times using the POWEROFF
BIOS function. The application program can specify the
continue or restart mode in the C register when calling
POWEROFF.

```
C reg. = 00H:          Continue mode
C reg. = 01H:          Restart mode
```

One caution must be observed when calling POWEROFF in
the continue mode: the CALL statement calling POWEROFF
must be immediately followed by an EI statement. This
is because when the MAPLE is subsequently powered on in
the continue mode, program execution will continue at
the instruction immediately following the CALL statement
with the MAPLE in the DI state.

```
LD   C, 00H
```

```
CALL POWEROFF ;MAPLE is powered off at this point.
EI              ;Execution resumes at this point in
                 the DI state when MAPLE is powered
                 on again.
```

9.6 Turning Power Off Always in the Continue Mode

As shown in the processing flow on page 9-10, the MAPLE
can be powered off always in the continue mode by
loading an appropriate value into the FRCECNTN flag
during the power-off sequence.

```
FRCECNTN                   = 00H  Normal mode
Overseas version = F4E8H          (Powered off either in
Japanese-language version         the restart or continue
                 = F25EH          mode depending on the
                                  power-off conditions)
                          ≠ 00H   Always powered off in
                                  continue mode.
```

The default value of FRCECNTN is 00H. The contents of
FRCECNTN are preserved until the RESET switch is
pressed.

BASIC always tries to turn MAPLE power off in the

continue mode using this flag except when waiting for a command (i.e., when executing a BASIC program).

9.7 Changing the Key for Specifying the Continue Mode

Normally the MAPLE is powered off in the continue mode when the user turns the POWER switch to OFF while holding down the CTRL key. The user, however, can power off the MAPLE in the continue mode by turning the POWER switch to OFF while holding down a key other than CTRL by modifying the contents of a work area.

CNTNKEY: Overseas version = 0F008H

          Japanese-language version = 0ED08H

    Bit 0 = CTRL (Left side)

    Bit 1 = SHIFT (Left side)

    Bit 2 = CAPS LOCK

    Bit 3 = CTRL (Right side)

    Bit 4 = SHIFT (Right side)

    Bit 5 = GRPH

    Bit 6 = Not used.

    Bit 7 = Not used.

The MAPLE is turned off in the continue mode if one of the keys whose corresponding bits in CNTNKEY are 1 is held down when the POWER switch is set off.

CNTNKEY defaults to 00001001B and the MAPLE is turned off in the continue mode if the CTRL key on either side is pressed. The contents of CNTNKEY are preserved until cleared by a system initialize. If CNTNKEY is set to 00H, the MAPLE is powered off in the continue mode when the POWER switch is turned off, with or without a function key.

9.8 Relationship between Power-off Interrupts and BIOS

The MAPLE is powered off by three types of interrupts from the 7508 CPU as discussed in 9.3. Since the power-off sequence activated by these interrupts stops any running I/O devices or displays the "CHARGE BATTERY" message on the screen, a system hang-up may result, disabling continue mode power-off, if these interrupts occur while BIOS is performing an I/O operation as explained in 8.8. To avoid this, the OS uses PREBIOS and PSTBIOS as the alarm/wake functions do to defer the actual power-off sequence in case a power-off interrupt occurs during execution of a BIOS function, until the BIOS function terminates (after which case the power-off sequence is carried out by PSTBIOS).

The following two flag areas are used to achieve the
above objective:

YPOFDS: Overseas version = 0F0D9H

           Japanese-language version = 0EDB9H

           Indicates the reason while 7508 interrupts are
           disabled.

           Bit 7 = 1: Disabled because BIOS is in execution.

           Bit 6 = 1: Disabled because password is being
                      entered.

           Bit 5 = 1: Disabled because alarm/wake message
                      is being displayed.

           Bit 4 = 1: Disabled because system message is
                      being displayed.

           Bit 3 = 1: Disabled by BASIC.

           Bit 2 = 1: Disabled by scheduler.

           Bit 1 = 1: Disabled by MTOS.

           Bit 0 = 1: Reserved (for applications).

YPOFST: Overseas version = 0F0DAH

           Japanese-language version = 0EDBAH

           Indicates that a power-off interrupt has
           occurred while 7508 interrupts are disabled.

           The meanings of the bits are identical to
           those in YPOFDS.

BIOS
as
viewed
from
the
user

PREBIOS

Set YPOFDS
bit 7 to on

Power off interrupt

BIOS

BIOS
processing

YPOFST
← YPOFDS

Y

YPOFST=ØØH

YPOFDS=ØØH

N

Y

Set YPOFDS
bit 7 to off

Power off
processing

PSTBIOS

N

YPOFST=ØØH

Same
processing

Y

Power off
processing

Set YPOFDS
bit 7 to off
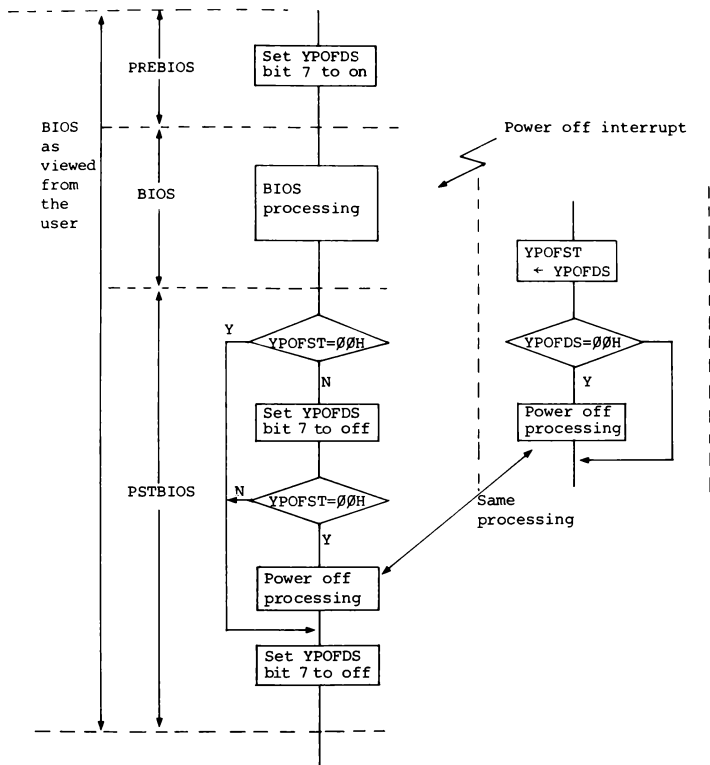
9.9 Method of Inhibiting Power-off Sequence from Application Program

Some application programs may not want MAPLE power to be turned off (naturally triggered by an interrupt) during the execution of some specific operations. This can be accomplished by the OS using the DI instruction or by inhibiting interrupts from the 7508 CPU. These measures, however, will also inhibit other interrupts (e.g., keyboard). To avoid this, the application program can and must perform the same operations as PREBIOS and POSTBIOS do as explained in 9.8.

The interrupt-triggered power-off sequence can be disabled using the YPOFDS flag. As explained on page 9-17, YPOFDS specifies what conditions inhibit power-off interrupts. Bit 0 of YPOFDS is reserved for application programs. The procedure given on the next page shows how to inhibit power-off interrupts from the application program.

When a power-off interrupt occurs while the interrupt-triggered power-off sequence is disabled (YPOFST contains a nonzero value), set only YPOFST bit 7 to on (to make the system believe that the interrupt occurred during BIOS processing) and make a dummy call to BIOS. PSTBIOS will then check YPOFST bit 7 and carry out a power-off sequence.
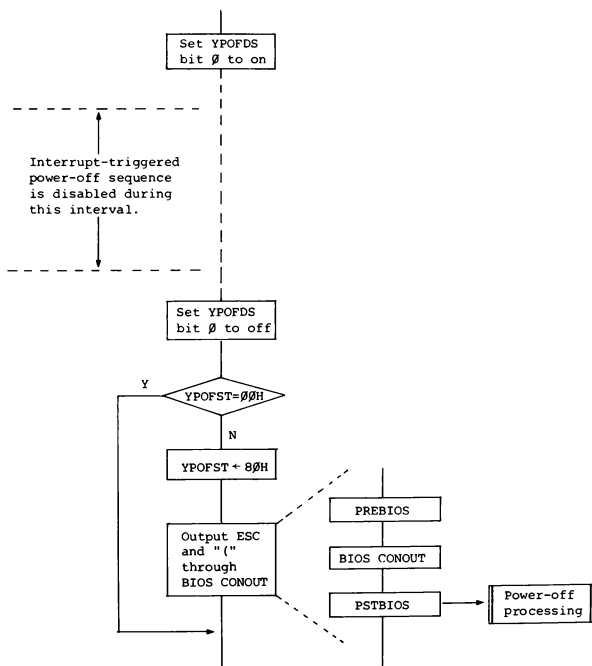
The ESC sequence, ESC and "(", output through the BIOS CONOUT function does not processing.

Chapter 10   Interrupt Processing

10.1   Interrupt Levels

The MAPLE responds to the following six levels of
interrupts:

Precedence	Type	Description
High	7508 interrupt	Interrupt from the 7508 4-bit CPU.
	8251 interrupt	RS-232C receive interrupt.
	CD interrupt	RS-232C CD interrupt.
	ICF interrupt	Interrupt from the bar code reader.
	OVF interrupt	FRC overflow interrupt.
Low	EXT interrupt	Interrupt from external devices.

When two interrupts occur simultaneously, only the
interrupt of the higher precedence is taken.

10.2 Interrupt Processing

(1) Z80 status

The Z80 CPU handles interrupts in the following
operating conditions:

    Interrupt mode: 2

    Contents of the interrupt register: 0EFH

The interrupts described in the previous section
generate the following values (the low-order address of
the interrupt processing routine):

Interrupt level	Low-order address
7508 interrupt	0F0H
8251 interrupt	0F2H
CD interrupt	0F4H
OVF interrupt	0F6H
ICF interrupt	0F8H
EXT interrupt	0FAH

The interrupt JUMP vector consists of the following
entries:

0EFF0H   DW [ ] : 7508 interrupt processing routine
                     starting address.

0EFF2H   DW [ ] : 8251 interrupt processing routine
                     starting address.

0EFF4H   DW [ ] : CD interrupt processing routine
                     starting address.

0EFF6H   DW [ ] : OVF interrupt processing routine
                     starting address.

0EFF8H   DW [ ] : ICF interrupt processing routine
                     starting address.

0EFFAH   DW [ ] : EXT interrupt processing routine
                     starting address.

(2) Control flow during interrupt processing
As shown above, the interrupt vector begins at address
0EFFH. This is because the MAPLE uses tow banks;
unexpected results may occur unless the interrupt vector
is placed somewhere in the resident area. Accordingly,
two areas are provided for processing interrupts: one in
RAM (INTRAM) and the other in ROM (INTROM). Although
all entries of the interrupt vector contain addresses in

INTRAM, interrupt processing requiring a large amount of memory (7508 or 8251 interrupt processing) is actually performed in INTEROM after bank switching. The control flow for interrupt processing is illustrated below.
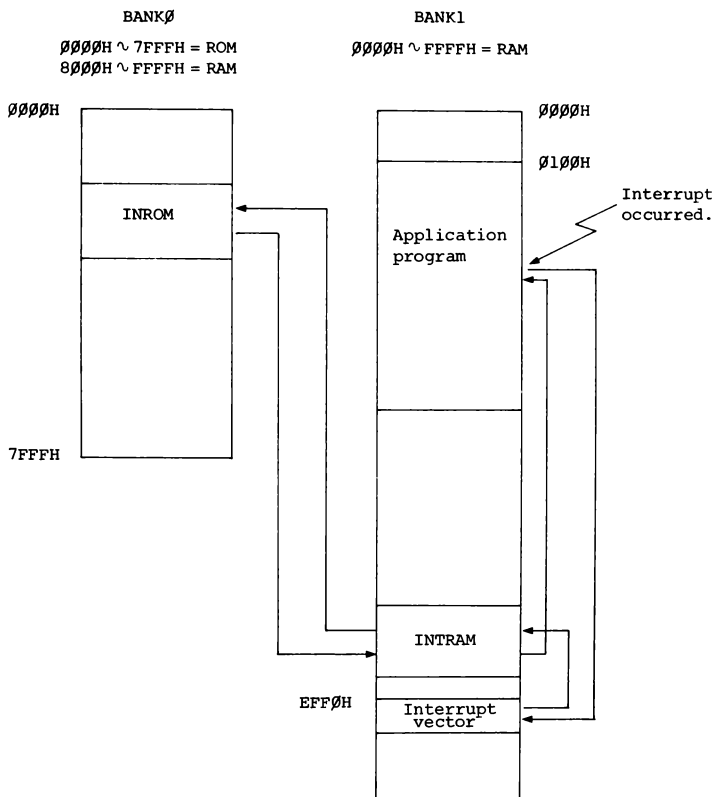
1) An interrupt occurred.

2) Transfers control to the pertinent interrupt processing routine in INTRAM as routed by the interrupt vector.

3)* Transfers control to the pertinent interrupt processing routine in INTROM after bank switching.

4)* After interrupt processing is completed, sets up the original bank and returns control to the INTRAM location immediately before the point where the bank switching to the INTROM bank occurred.

5) After interrupt processing is completed, returns control to the point where the interrupt occurred with the RETI instruction.

*: Only for 7508 or 8251 interrupts.

The MAPLE OS supports none of the CD, ICF, OVF, and EXT interrupts. Instead, the OS provides a hook which is called by the interrupt processing routine so that the user can easily extend the portion of the interrupt processing routine that carries out the actual interrupt processing tasks.

Hook table contents

Address	Hook name
0EFA1H	CDHOOK: JP RETURN
0EFA4H	ICFHOOK: JP RETURN
0EFA7H	OVFHOOK: JP RETURN
0EFAAH	EXTHOOK: JP RETURN
0EFB7H	RETURN: RET

CPU control can be routed to the desired program by changing the address currently pointing to RETURN. See the description about individual interrupts in sections 10.5 and later for how control is passed to the hook table.

(3) Enabling/disabling interrupts

Interrupts can be enabled or disabled from the
application program by sending control data to I/O port
4 in one of the following two ways:

    1) Using the MASKI (WBOOT + 57H) BIOS function.

    2) Sending data directly to port 4.

The user is recommended to use the MASKI function. See
Chapter 4, "BIOS Functions" for MASKI. The following
paragraphs explain the latter method.

[Port 4]

Bits 0 through 5 of port 4 correspond to the interrupt
sources. Required interrupts can be enabled or disabled
by sending control data in the corresponding bit pattern
to port 4.

Bit	Interrupt	Meaning of bit data
0	7508 interrupt	0: Disabled 1: Enabled
1	8251 interrupt	0: Disabled 1: Enabled
2	CD interrupt	0: Disabled 1: Enabled
3	OVF interrupt	0: Disabled 1: Enabled
4	ICF interrupt	0: Disabled 1: Enabled
5	EXT interrupt	0: Disabled 1: Enabled

Bits 6 and 7 are not used.

When sending control data directly to port 4, the application program must save the same data in a work area labeled IER in advance. To do this, the application program must follow the procedure given below.

     IER .... Loaded with the latest data sent to
                  port 4.
     (Overseas version = ØFØB3H
      Japanese-language version = ØED93H)

     LD   A,(IER)   ; Get previous data.
     ┌─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ┐
     │ Perform bit manipulation │   ; Update data.
     │ on A reg. and enables   │
     │ or disables interrupts.   │
     └─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ┘
     LD   (IER),A   ; Save latest
                                        data.
     OUT   (4),A

All interrupts are automatically disabled when the RESET switch is pressed. Subsequently, only 7508 interrupts are enabled by the OS.

Bits of port 4 can be read to identify the interrupt source. The meaning of bits differ with the type of the interrupt. See sections 10.3 and later for details.

10.3 7508 Interrupts

(1) When generated

7508 interrupts are generated by the 7508 4-bit CPU when
the following conditions occur:

1) The keyboard is turned on or off.

2) The POWER switch is turned on or off.

3) The Z80 RESET switch (on main unit left side
   panel) is pressed.

4) The 7508 RESET switch (on the MAPLE's bottom
   panel) is pressed.

5) The battery voltage drops below a certain level
   (power failure).

6) An alarm/wake time is reached.

7) A 1-second interrupt (generated every second)
   occurs.

(2) Interrupt processing

The 7508 and Z80 CPUs exchange commands and data via a
serial data line. When the 7508 sends one of the above
interrupt requests to the Z80, the Z80 interrupt
handling routine reads the status from the 7508 to
identify the interrupt source and invokes the
corresponding interrupt processing routine.

Bit 0 of port 4 is set to 1 when the interrupt from the 7508 is accepted and set to 0 when the interrupt status is read from the 7508.
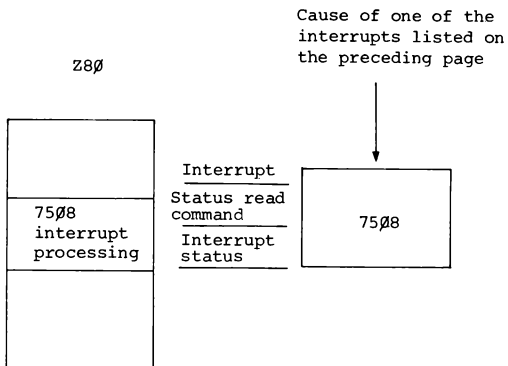
See Chapter 11, "7508 CPU" for details on communication between the Z80 and 7508 CPUs.

(3) Modifying an interrupt processing routine from applications

As described in (1), 7508 interrupts exercise control over the basic MAPLE operations (e.g., keyboard on/off, system reset, power on/off, etc.). It is highly likely that the entire system would not function if the user attempts to perform his own interrupt processing by changing the interrupt vector address. Therefore, the application programs are disallowed to modify any 7508 interrupt routines.

However, since the system alarm/wake interrupt processing routines are given hooks, it is possible for the user to add his own interrupt processing routines using the hooks (see Chapter 8, "Alarm/Wake Feature").

10.4 8251 Interrupts

(1) When generated

An 8251 interrupt is generated when the 8251 receives
data from the RS-232C. More exactly, it is generated
when the RxRDY of the 8251 is set to 1.

(2) Interrupt processing

The Z80 CPU can communicate with the 8251 through port
addresses 0CH and 0DH.

Port	Read mode	Write mode
∅CH	8251 receive data	8251 send data
∅DH	8251 status	command to 8251

The 8251 RxRDY is reset when the interrupt processing
routine receives RS-232C interrupt from the 8251 and
gets the received data from port 0CH. Bit 1 of port 4
indicates the 8251 RxRDY state.

No special sequence is required to access port 0CH and
0DH.
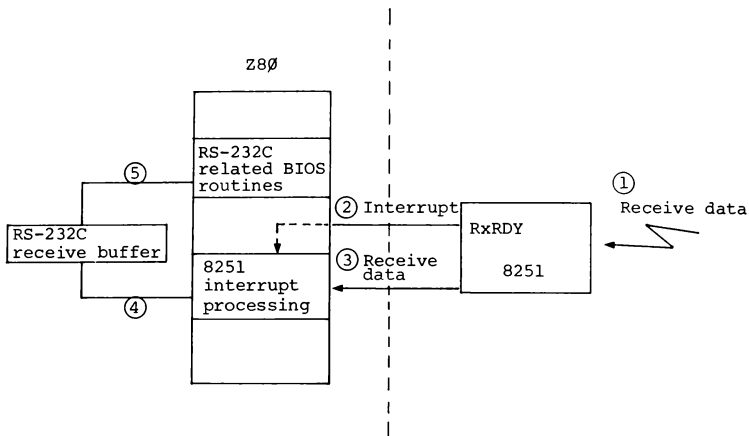
1) The 8251 receives RS-232C data.

2) 8251 RxRDY is set to 1 and an interrupt is
   generated to the Z80.

3) The interrupt routine gets receive data via port
   address 0CH.

4) The interrupt routine places the received data into
   the RS-232C receive buffer.

5) The BIOS gets data from the receive buffer and passes
   it to the application program.

The user can have his own interrupt routine perform
steps 3) through 5) by modifying the pertinent interrupt
vector address.

10.5 CD Interrupts

(1) When generated

CD interrupts are generated when the CD line at the RS-232C connector reaches -8V (-3V to -15V).

(2) Interrupt processing

CD interrupts must be disabled by the interrupt processing routine because successive interrupts can occur if when the CD line is held at -8V. (Follow the procedure below.)

```
LD    A, (IER)
RES   2, A
LD    (IER), A
OUT   (4), A
```

The OS interrupt processing routine does nothing for CD interrupts but simply calls an entry in the hook table. (See the flow chart on the next page.)

```
   ┌──────────────┐
   │      EN      │
   └──────────────┘
          │
   ┌──────────────┐
   │  Switch SP   │ ─ ─ ─ ─ ┐
   └──────────────┘         │
          │                 │
   ┌──────────────┐         │
   │  Save AF and │         │
   │  HL contents │         │
   └──────────────┘         │
          │                 │
   ┌──────────────┐         │
   │  Disable CD  │         │
   │  interrupts  │         │
   └──────────────┘         │
          │                 │
   ┌──────────────┐         │
   │     CALL     │         │
   │    CDHOOK    │         │
   └──────────────┘         │
          │                 │
   ┌──────────────┐         │
   │  Restore AF  │ ─ ─ ┘   │
   │   and HL     │         │
   └──────────────┘         │
          │                 │
   ┌──────────────┐         │
   │  Replace SP  │ ─ ─ ─ ─ ┘
   └──────────────┘
          │
   ┌──────────────┐
   │      EI      │
   └──────────────┘
          │
   ┌──────────────┐
   │     RETI     │
   └──────────────┘
```
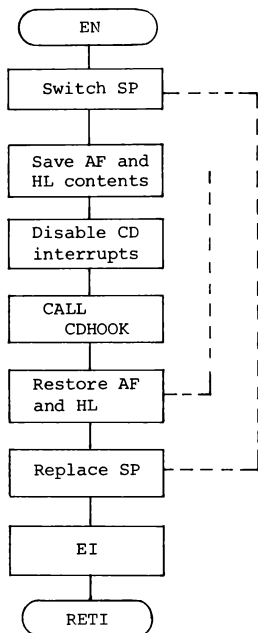
(3) Modifying interrupt processing from an application

Control is routed to the user routine for CD interrupt
service by modifying the interrupt vector or using
the CDHOOK shown above. The following paragraphs focus
on the use of the CDHOOK. See 10.9 for the former
method.

. CDHOOK address

    0EFA1H CDHOOK: JP RETURN

                   |

               Address containing a RET

               instruction

Control can be passed to the user processing routine by
changing the contents of 0EFA2H and 0EFA3H.

* Programming notes to be taken when using an interrupt hook
1) Change the jump address of the hook while the
   corresponding interrupt is disabled.
2) Set the jump address to 8000H or higher. (Use the
   last portion of the TPA or the user BIOS area if the
   hook is to be made resident.)
3) Set the stack pointer to a user stack area.
   (This is because the system stack area may be full
   when control is sent to the hook.)
4) Use registers after saving their old values.
5) Reset the jump address of the hook to the original
   address when it is no longer needed. Otherwise, the
   modified address will be preserved until the next
   system reset occurs.
6) Do not use the EI command.
7) Do not use system calls (BDOS or BIOS call).

10.6 OVF Interrupts

(1) When generated

OVF interrupts are generated when an overflow condition occurs in the counter running at the 614.4 kHz clock rate (Free Running Counter). The period of the clock is approximately 100 ms.

$$\frac{1}{614.4 \times 10^3} \times 2^{16} \fallingdotseq 100 \text{ ms}$$

(2) Interrupt processing

As with CD interrupts, the OS does nothing for OVF interrupts but supplies a hook. When an OVF interrupt occurs, bit 2 of port 01H must be set to 1 to reset the INTR signal set by the FRC overflow interrupt.
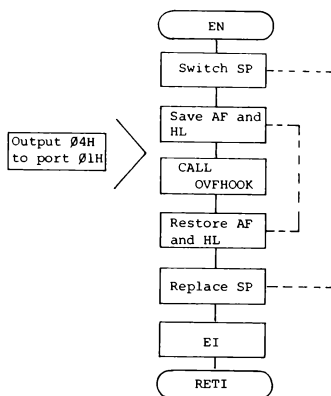
```
         ┌─────────┐
         (   EN    )
         └────┬────┘
        ┌─────┴─────┐
        │ Switch SP ├ ─ ─ ─ ┐
        └─────┬─────┘       │
        ┌─────┴─────┐       │
        │ Save AF and├ ─ ┐  │
        │ HL        │    │  │
        └─────┬─────┘    │  │
        ┌─────┴─────┐    │  │
        │ CALL      │    │  │
        │ OVFHOOK   │    │  │
        └─────┬─────┘    │  │
        ┌─────┴─────┐    │  │
        │ Restore AF├ ─ ┘  │
        │ and HL    │       │
        └─────┬─────┘       │
        ┌─────┴─────┐       │
        │ Replace SP├ ─ ─ ─ ┘
        └─────┬─────┘
        ┌─────┴─────┐
        │    EI     │
        └─────┬─────┘
         ┌────┴────┐
         (  RETI   )
         └─────────┘
```

┌──────────────┐
│ Output Ø4H   │
│ to port Ø1H │
└──────────────┘

(3) Modifying interrupt processing from an application
This paragraph describes the way to use the OVFHOOK
routine shown in the above chart. See Section 10.9 for
the procedure for modifying the interrupt vector.

o OVFHOOK address

         0EFA7H   OVFHOOK: JP RETURN
                               ↑
                        Address containing a RET
                        instruction

OVF interrupt processing by a user routine can be
specified by modifying the contents of 0EFA8H and
0EFA9H. Follow the programming notes given on 10.16
when using the hook.

10.7 ICF Interrupts

(1) When generated

ICF interrupts are generated when the state of the data signal at the bar code reader connector changes (negative-to-positive or positive-to-negative transition).

(2) Interrupt processing

When a change in the data state occurs, an interrupt is generated and the FRC contents at that moment is immediately loaded into the ICR (Input Capture Register).

The FRC is a 16-bit counter running at the 614.4 kHz clock rate (this counter is also used for OVF interrupts). The contents of the ICR can be obtained by reading I/O ports 2 (lower 8 bits) and 3 (higher 8 bits).

Ports 2 and 3 must be read in that order since the ICF interrupt signal is reset whenever I/O port 3 is read. ICF interrupts can occur successively until port 3 is read.

The timing at which an ICF interrupt is generated
depends on the data placed at bits 1 and 2 of I/O port
0.

Port Ø		Timing at which interrupt occurs
bit 2	bit 1	
Ø	Ø	No interrupt is generated when a change in the data signal state.
Ø	1	Generated at a falling edge.
1	Ø	Generated at a rising edge.
1	1	Generated at both falling and rising edges.

Load data into I/O port 0 using the following procedure:

```
LD    A,(CTLR1)
```

```
Code for manipulating A reg.
bits 1 and 2
```

```
LD    (CTLR1),A
OUT   (Ø),A
```

This interrupt is used to measure the relative bar code
width (time).
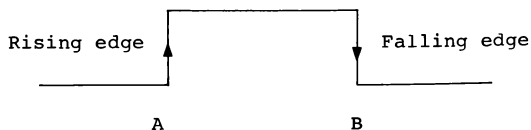
For example, if interrupts are generated on negative-to-positive and positive-to-negative transitions (see the above figure), the bar code width (A - B) can be calculated by saving the ICR contents at interrupt A into a work area and subtracting the ICR contents at interrupt B from the value in that work area.

The ICR is a 16-bit counter running at 614.4 kHz, as described before, and wraps around every 100 ms or so (it cannot measure time intervals longer than 100 ms). The interrupt processing routine, however, can measure intervals longer than 100 ms, by using OVF interrupts,

that is, by counting the number of overflows.
The OS only calls the hook table and does nothing for ICF interrupts. The flow chart on the next page shows that procedure.
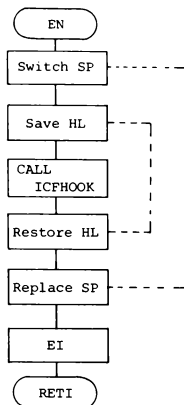
```
        ( EN )
          |
   +-------------+
   | Switch SP   |--------------+
   +-------------+              |
          |                     |
   +-------------+              |
   | Save HL     |----+         |
   +-------------+    |         |
          |           |         |
   +-------------+    |         |
   | CALL        |    |         |
   |   ICFHOOK   |    |         |
   +-------------+    |         |
          |           |         |
   +-------------+    |         |
   | Restore HL  |----+         |
   +-------------+              |
          |                     |
   +-------------+              |
   | Replace SP  |--------------+
   +-------------+
          |
   +-------------+
   |     EI      |
   +-------------+
          |
      ( RETI )
```

(3) Modifying interrupt processing from an application

This paragraph describes the way to use the ICFHOOK
shown in the above chart. See Section 10.9 for the
procedure for modifying the interrupt vector.

o ICFHOOK address

        0EFA4H   ICFHOOK: JP RETURN

                            |

                          Address containing an RET

                          instruction

ICF interrupts can be processed using a user interrupt
processing routine by modifying the contents of 0EFA5H
and 0EFA6H. Follow the programming notes on page 10-16
when using the hook. The interrupt processing routine
must read I/O ports 2 and 3 when handling ICF
interrupts. (This is to reset the ICF interrupt
signal.)

10.8 EXT Interrupts

(1) When generated

EXT interrupts are generated when the INTEXT line of
the system bus which can be accessed from the MAPLE's
rear side is set to 0V.

(2) Interrupt processing

EXT interrupts must be disabled in the interrupt
processing routine because successive interrupts can
occur if INTEXT is held at 0V. (Follow the procedure
below.)

```
    LD    A, (IER)
    RES   5, A
    LD    (IER), A
    OUT   (4), A
```

The OS only calls the hook table and does nothing.

```
      ( EN )
         |
   [ Switch SP ]
         |
   [ Save HL ]
         |
   [ CALL
      EXTHOOK ]
         |
   [ Restore HL ]
         |
   [ Restore SP ]
         |
   [ EI ]
         |
   ( RETI )
```
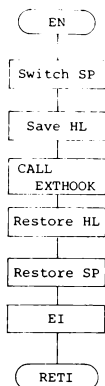
(3) Modifying interrupt processing from an application

This paragraph describes how to use OVFHOOK shown in the above chart. See Section 10.9 for the procedure for modifying the interrupt vector.

o EXTHOOK address

        0EFAAH   EXTHOOK: JP RETURN
                             ↑
                           Address containing an RET
                           instruction

The EXT interrupt routine can be processed by a user-supplied interrupt processing routine by modifying the contents of 0EFABH and 0EFACH.

Observe the programming notes on page 10-16 when using hooks. When processing an EXT interrupt, the user interrupt processing routine must disable subsequent EXT interrupts at its beginning.

10-24

10.9 Procedure for Modifying Interrupt Vectors

This section describes the procedure for modifying the interrupt vectors at address 0EFF0H or higher to specify user routines for interrupt service.

(1) Modifying an interrupt vector

1) Place an interrupt routine in RAM (at location 8000 or higher).

2) Save the current interrupt inhibit status. (Use the BIOS MASKI function.)

3) Save the current interrupt vector status.

4) Inhibit the interrupt whose interrupt vector is to be modified. (Use the BIOS MASKI function.)

5) Modify the interrupt vector.

6) Enable the interrupt that has been inhibited above. (Use the BIOS MASKI function.)

(2) Restoring the interrupt vector to the original state.

1) Inhibit the interrupt whose interrupt vector is to be restored. (Use the BIOS MASKI function.)

2) Restore the interrupt vector to the original state. (Use the contents saved in step (1)-3.)

3) Restore the original inhibit status (the contents saved in step (1)-2.) (Use the BIOS MASKI function.)

10.10 Programming Notes on Interrupt Processing

(1) Changes made in interrupt vectors or hooks are preserved until the RESET switch is pressed. Therefore, be sure to restore the changed vectors that are no longer needed to the original vectors.

(2) User interrupt routines must be placed at locations 8000 or higher. Use the user BIOS area if the interrupt routines are to be made resident.

(3) The user is disallowed to make modifications for the 7508 interrupt routines.

(4) Define a user stack area in the user interrupt processing routine. Save the contents of registers before using the registers and replace them before terminating processing.

(5) Do not use system calls (BDOS or BIOS) in any of the interrupt processing routines.

(6) User interrupt routines hooked by modifying their interrupt vectors must end with an EI or RETI instruction.

(7) OS-supplied interrupt processing routines use up to two levels (four bytes) of the user stack area. The application program in the TPA must take these four bytes into consideration when defining a stack area.

Chapter 11   7508 CPU

This chapter describes functions and use of the 7508 4-bit sub-CPU.

11.1   7508 CPU Functions

The 7508 CPU performs the following functions:

(1) Serving keyboard functions such as keyboard scan and auto repeat.

(2) Controlling the POWER switch.

(3) Controlling the RESET switch.

(4) Serving the one-second interval timer function.

(5) Measuring the battery voltage.

(6) Serving the alarm function.

(7) Turning on and off the main CPU switch.

(8) Reading temperature data.

(9) Serving the calendar and clock functions.

(10) Reading data from an AD converter.

(11) Controlling the DIP switches.

(12) Transferring serial data to and from main CPU.

(13) Controlling the DRAM refresh mode.

In addition to generating interrupts, the 7508 CPU transfers commands and data to and from the Z80 CPU via

a serial data line using a handshake technique.

The processing results for functions (1) through (6) on the previous page are returned to the Z80 in the form of interrupts.

11.2 Interface to Z80

The Z80 CPU uses the following ports when interfacing with the 7508 CPU:

Port	Read/Write	Meaning
06H	Read	Data from the 7508.
	Write	Data to the 7508.
05H	Read	Bit 3 carries the control signal for the serial bus to the 7508. 1: Accessible. 0: Inaccessible.
01H	Write	Used to reset the above control signal. 1: Resets. 0: Does nothing.

The flowchart on the next page illustrates the procedure for transferring commands or data to and from the 7508 CPU using the above I/O ports.

When sending a command or data to the 75Ø8:

```
              ┌─────────┐
             (    EN     )
              └────┬────┘
                   │
     ┌─────────────┤
     │       ┌──────────────┐
     │       │  IN  A,(5)   │ ─────────────── Check whether 75Ø8 is ready.
     │       └──────┬───────┘
     │              │
     │           ◇──────◇
  =Ø │          ╱ A reg.  ╲
     └─────────◇  Bit 3    ◇
                ╲         ╱
                 ◇───┬──◇
                  =1 │
          ┌──────────────────┐
          │ Load command     │
          │ or data into     │
          │ A reg.           │ ─────────────── Send command or data to 75Ø8.
          └────────┬─────────┘
                   │
          ┌──────────────────┐
          │  OUT  (6),A      │
          └────────┬─────────┘
                   │
          ┌──────────────────┐
          │  LD   A,2        │
          │  OUT  (1),A      │ ─────────────── Reset.
          └────────┬─────────┘
                   │
              ┌─────────┐
             (    EX     )
              └─────────┘
```
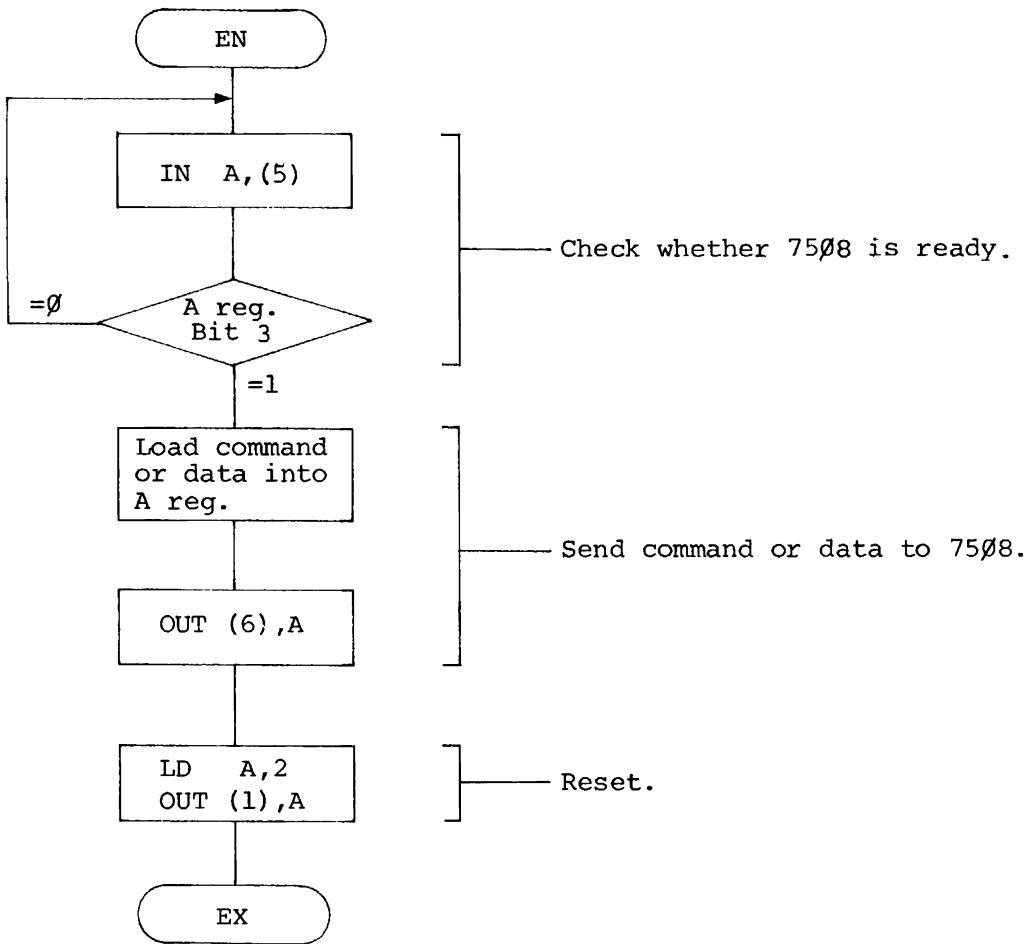
When one or more parameters are to be sent following the
command, the above procedure is repeated the number of
times equal to the number of command and parameter
bytes.

11-4

When sending a command and receiving data as the
response to the command:

```
                    ┌─────────┐
                   (    EN     )
                    └─────────┘
                         │
                  ┌──────────────┐        ┌───── See flowchart on
                  │ Send command │ ───────┤       previous page.
                  └──────────────┘        └─────
                         │
  ┌─────────────┐        │
  │ LD    A,2   │   ┌──────────────┐       ┌─────
  │ OUT   (1),A │   │  IN   A,(5)  │ ──────┤
  └─────────────┘   └──────────────┘       └───── Checks whether 75Ø8
                         │                         is ready.
              =Ø    ╱ A reg. ╲
         ┌─────────<   Bit 3   >
         │          ╲         ╱
         │              │ =1
         │         ┌──────────────┐       ┌─────── Receive data.
         │         │  IN   A,(6)  │ ──────┤
         │         └──────────────┘       └───────
         │              │
         │   N      ╱          ╲
         └─────────< End of data? >
                    ╲          ╱
                         │ Y
                    ┌─────────┐
                   (    EX     )
                    └─────────┘
```
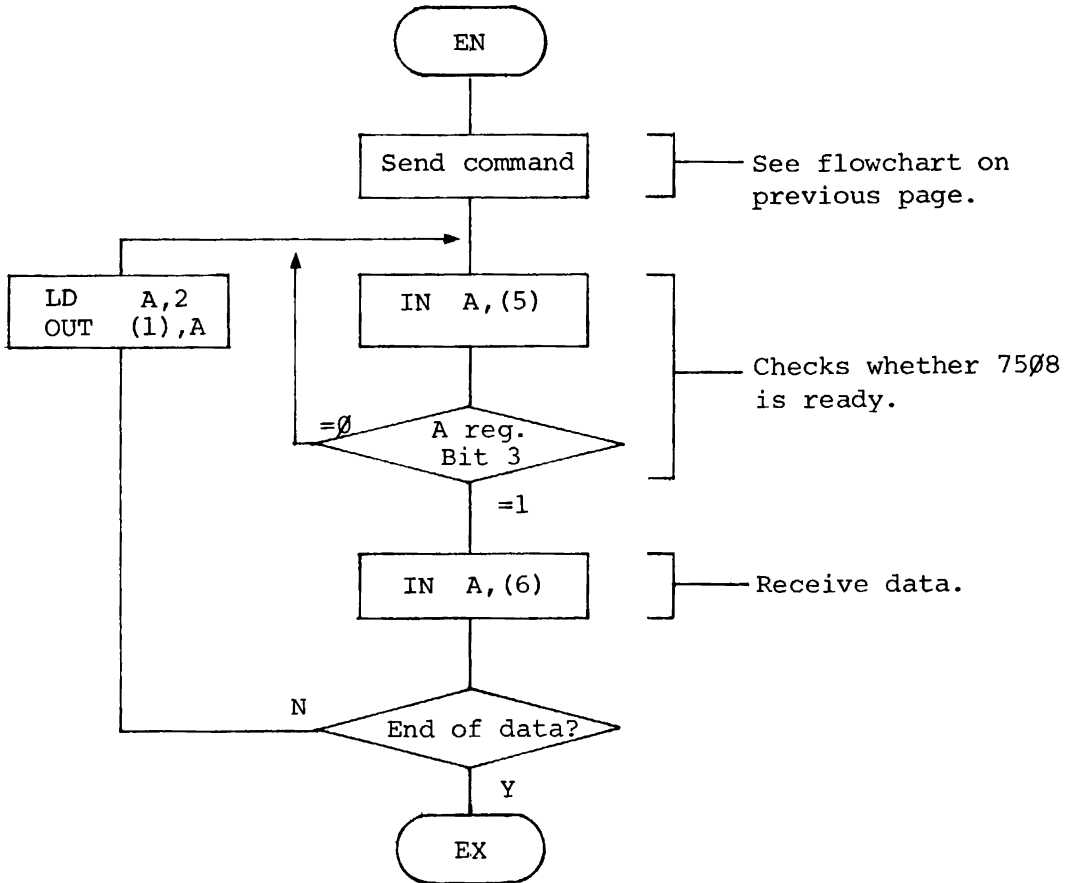
<u>Points to be noted when using the 7508 directly in an</u>
<u>application program</u>

1) Disable 7508 interrupts while transferring a

    command or data to or from the 7508 CPU. (Use the

    DI instruction or the BIOS MASKI call.)

    If a 7508 interrupt occurs while the application

    program is communicating with the 7508 CPU, the Z80

    CPU may not receive correct return information

    or, at the worst case, it may hang up because the

    Z80 CPU will call for a new 7508 service from its

    interrupt handling routine and consequently the

    original command to the 7508 CPU will be ignored.

2)   Complete the send or receive sequence for a command
     before proceeding with the next command. Normal
     processing cannot be guaranteed unless the
     application program sends or receives the required
     number of data bytes; otherwise, a system
     hangup would result in the worst case.

11.3 7508 Commands

The table below lists the commands that the 7508 CPU receives from the Z80 CPU.

Command Chart

Command function	Code	Command function	Code
Power off Z80	Ø1H	Read time.	Ø7H
Read 75Ø8 status.	Ø2H	Set alarm.	19H
Reset keyboard.	Ø3H	Read alarm.	Ø9H
Set keyboard repeat start time	Ø4H	Disable alarm.	29H
Set keyboard repeat interval.	14H	Enable alarm.	39H
Read keyboard repeat start time	24H	Read battery voltage.	ØCH
Read keyboard repeat interval.	34H	Read temperature.	1CH
Disable keyboard auto repeat.	Ø5H	Read analog input 1.	2CH
Enable keyboard auto repeat.	15H	Read analog input 2.	3CH
Disable key-in interrupt	Ø6H	75Ø8 power-on reset	ØFH
Enable key-in interrupt.	16H	Read DIP-SW	ØAH
Disable one-second interrupt.	ØDH	Set power failure detect voltage.	ØBH
Enable one-second interrupt.	1DH	Set full charge voltage.	1BH
Set time.	17H	Read power or trigger switch.	Ø8H

(1) Power off Z80

    Code: 01H

    Send data: None.

    Receive data: None.

    Function: Turns off power to the Z80.

    Note: This command is not used in application
          programs. It is used by the POWEROFF BIOS
          function.

11-7

(2) Read 7508 status

Code: 02H

Send data: None

Receive data: 1 byte (7508 status)

Function: Reads the 7508 status. It is used to
read the 7508 status when an interrupt occurs to
identify the interrupt source. The meanings of the
status byte are as follows:

- 0BEH and below: Interrupts from the keyboard.

- 0C0H and above: Interrupts from sources other
than the keyboard.

- 0BFH: End of status.

1) Interrupts from the keyboard

The status byte 0BEH and below indicate
interrupts from the keyboard. The correspondence
between the keys and status values are shown on
the next page. For example, status code 73H is
returned when the space key (No. 71 on the
keyboard) is pressed.

The 7508 returns only one status code when an
ordinary key is pressed and released. For a key
No. 43, 57, 70, 72, 68, or 69, however, the 7508
returns a status code (0B2H − 0B7H) when the key
is pressed and returns another code (0A2H − 0A7H)
when it is released.

2) Interrupts from sources other than the keyboard

The status byte ØCØH and above indicate interrupts from sources other than the keyboard. Each bit of the status byte has the meaning listed below. When two or more interrupts occur simultaneously, the corresponding bits are set to 1.

Correspondence between the key numbers and key codes

1		2	3	4	5	6	7	8	9		10	11	12	13
14	15	16	17	18	19	20	21	22	23	24	25	26	27
28	29	30	31	32	33	34	35	36	37	38	39	40	41	42
43	44	45	46	47	48	49	50	51	52	53	54	55	56
57	58	59	60	61	62	63	64	65	66	67	68	69
70	71	72

Higher / Lower	0	1	2	3	4	5	6	7	8	9	A	B
0	2	1	29	46	62	21	37	54	12			
1	3	14	30	47	63	22	38	55	13			
2	4	15	31	48	64	23	39	56			OFF 43	ON 43
3	5	16	32	49	65	24	40	71			OFF 57	ON 57
4	6	17	33	50	66	25	41	58			OFF 70	ON 70
5	7	18	34	51	67	26	42	59			OFF 72	ON 72
6	8	19	35	52	10	27	44	60			OFF 68	ON 68
7	9	20	36	53	11	28	45	61			OFF 69	ON 69
8												
9												

<u>Status</u>

Bit 7: Always set to 1.

Bit 6: Always set to 1.

Bit 5: Set to 1 when a one-second interrupt

occurs.

Bit 4: Set to 1 when the RESET switch on the

left-side panel of the MAPLE main unit

is pressed.

Bit 3: Set to 1 when the INITIALIZE RESET

switch on the rear panel of the main

unit is pressed.

Bit 2: Set to 1 when a power fail interrupt

occurs.

Bit 1: Set to 1 when an alarm interrupt occurs.

Bit 0: Indicates the POWER switch state.

1: Power turned on.

0: Power turned off.

This status information is used to distinguish
between address 0 start interrupts and power-on
interrupts.

i) The source of an address 0 start interrupts
(Z80 CPU starting at address 0) can be
identified by examining the status bits in the
sequence shown below.

```
Status
   |
< Bit3 = 1 >──Y──▶  INITIALIZE RESET Switch
   |                pressed.   ────▶  System initialize
   N                                  processing
   |
< Bit4 = 1 >──Y──▶  RESET Switch
   |                pressed.   ────▶  Reset processing
   N
   |
< Bit1 = 1 >──Y──▶  Alarm interrupt
   |                           ────▶  Alarm/wake processing
   N                                  in power-off state
   |
   └──────────▶  POWER switch turned on.----▶POWER switch
                                               on processing
```
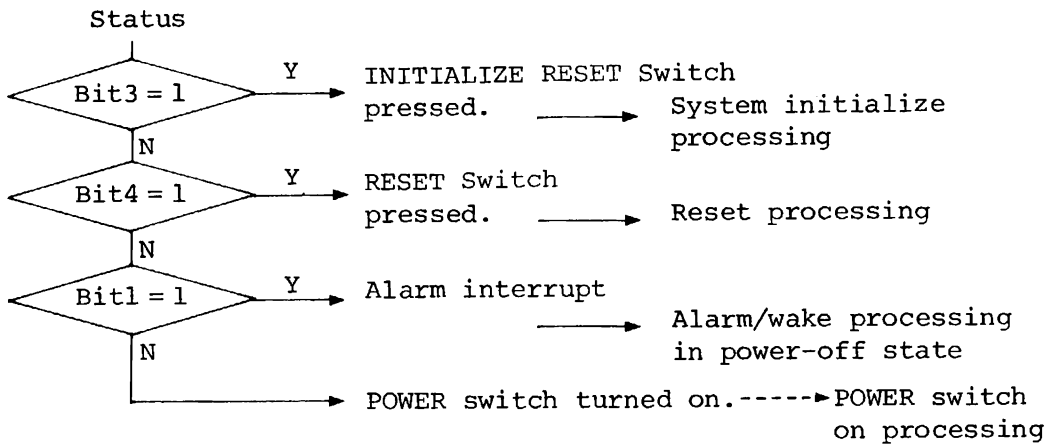
ii) 16 status values may be returned by power-on
interrupts. Since status byte bit 0 always
indicates the state of the POWER switch, the
correct interrupt source cannot be determined
unless the POWER switch state immediately before
the interrupt is known. The table below is used
to identify the interrupt source for status
values of 0C0H through 0C7H. The interrupt
sources for status values of 0E0H through 0E7H
correspond to 0C0H through 0C7H on a one-to-one
basis and their meanings are identical except
that they also indicate the occurrence of a 1-
second interrupt.

Status	POWER switch state before interrupt	
	OFF	ON
CØH		POWER switch turned off.
C1H	POWER switch turned on.	·
C2H	Alarm interrupt	POWER switch off and alarm interrupts occurred simultaneously.
C3H	POWER switch turned on.	Alarm interrupt.
C4H	Power fail interrupt	POWER switch turned off.
C5H	Power fail interrupt	Power fail interrupt
C6H	Power fail interrupt	POWER switch turned off.
C7H	Power fail interrupt	Power fail interrupt

*1: Since power is already turned on, the interrupt handling routine need nothing but to set the flag.

*2: This state cannot occur.

Interrupts are handled as follows when more than one status bit is 1:

- The power fail interrupt has the highest priority.

- Processing of the alarm interrupt may be deferred since a total of 10 alarm interrupts are generated.

For interrupt status values 0E0H to 0E7H, the interrupt handling routines need only perform one-second interrupt processing in addiction to the interrupt processing associated with status values 0C0H to 0C7H.

3) When the status value is 0BFH

The 7508 sub-CPU has a 7-byte buffer for storing keyed in data. It returns status code 0BFH when its key buffer holds no keyboard data. To read all data in the keyboard buffer, the application program need only execute this command repeatedly until a 0BFH code is received.

```
                        Status
                          |
                          |
              _____/ _____
             \   0C0H or above?        /──────────────────┐
              _____ _____/                    |
                          | \                              |
                          |                                ▼
                          |                         Interrupt from a
              _____/ _____             source other than
             \    0BFH?                /            keyboard (see 2)).
              _____ _____/
      ┌───────────────────┘   |
      |                       Y
      ▼                       |
                              ▼
Keyboard data present        End
(see 1)).  Data may
also be received from
another source.
```
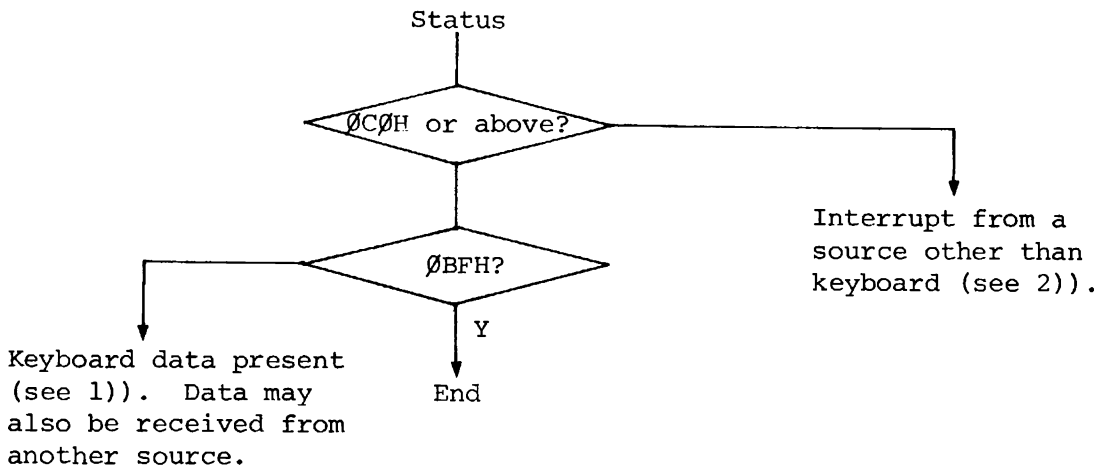
(3) Reset keyboard

Code: 03H

Send data: None.

Receive data: None.

Function:

1) Initializes the keyboard as follows:

- Sets the keyboard repeat start time to 656 ms.

- Sets the keyboard repeat interval to 70 ms.

- Clears the buffer.

- Enables interrupts from the keyboard.

2) Scans the keyboard and places the information concerning the currently pressed key.

11-13

(4) Set keyboard repeat start time

> Code: Ø4H
>
> Send data: 1 byte
>
> Receive data: None.

Function: This command defines the interval between the time when a key is first pressed (one key code is loaded into the buffer) and the time when the auto repeat function is to be started. This function causes a key code to be read repeatedly as long as the corresponding key is held pressed.

The send data is made up of one byte and has the following format:

bit7   bitØ

| 1 | | | | | | | |

Specifies the repeat start time in 1/64 second increments. The maximum repeat start time is approximately two seconds $(1/64 \times 2^7)$.

Always set to 1.
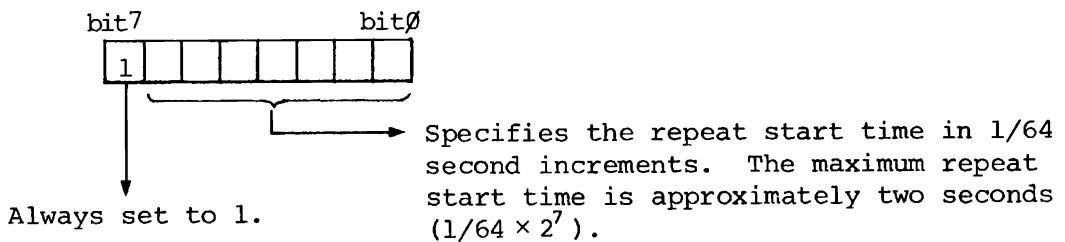
(5) Set keyboard repeat interval

Code: 14H

Send data: 1 byte

Receive data: None.

Function: Defines the interval at which the key code of a key being held pressed is to be entered repeatedly.

The send data consists of one byte and has the following format:

```
bit7              bit0
  ┌─┬─┬─┬─┬─┬─┬─┬─┐
  │1│ │ │ │ │ │ │ │
  └─┴─┴─┴─┴─┴─┴─┴─┘
```

Specifies the auto repeat interval in 1/256 second increments. The maximum period is approximately 0.5 second $(1/256 \times 2^7)$.
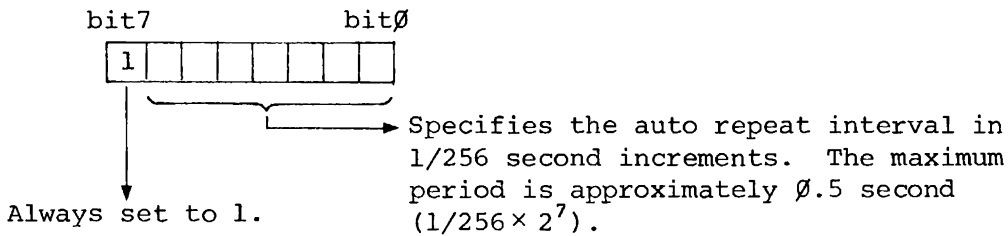
Always set to 1.

(6) Read keyboard repeat start time

Code: 24H

Send data: None.

Receive data: 1 byte

Function: Returns the currently set keyboard repeat start time. The send data specifies the keyboard repeat start time in 1/64 second increments. Bit 7 is always set to 0.

(7) Read keyboard repeat period

   Code: 34H

   Send data: None.

   Receive data: 1 byte

   Function: Returns the currently set keyboard repeat
      interval. As with command (5), send data
      specifies the repeat interval in 1/256 second
      increments. Bit 7 is always set to 0.

(8) Disable keyboard auto repeat

   Code: 05H

   Send data: None.

   Receive data: None.

   Function: Disables the keyboard auto repeat
      function.

(9) Enable keyboard auto repeat

   Code: 15H

   Send data: None.

   Receive data: None.

   Function: Enables the keyboard auto repeat
      function.

(10) Disable key-in interrupts

   Code: 06H

   Send data: None

   Receive data: None

Function: Disables key-in interrupts to the Z80
CPU. When a key is pressed after this command is
executed, only the key code is placed in the 7508
buffer and no interrupt request is sent to the Z80
CPU. When a command (11) is subsequently
executed, a key-in interrupt is generated at this
moment to the Z-80 unless the buffer is empty.

(11) Enable key-in interrupts

Code: 16H

Send data: None.

Receive data: None.

Function: Enables key-in interrupts to the Z80 CPU.

(12) Disable one-second interrupts

Code: 0DH

Send data: None.

Receive data: None.

Function: Disables one-second interrupts.

(13) Enable one-second interrupts

Code: 1DH

Send data: None.

Receive data: None.

Function: Enables one-second interrupts.

(14) Set time

Code: 17H

Send data: 8 bytes

Receive data: None.

Function: Specifies the year, month, day, hour,
minute, second, and day of the week for the
calendar/clock controlled by the 7508 CPU. The
send data has the following format:

	bit 7		4	3	bit 0
①	1	Ø Ø Ø		Tens digit of year	
②	1	Ø Ø Ø		Units digit of year	
③	1	Tens digit of month		Units digit of month	
④	1	Tens digit of day		Units digit of day	
⑤	1	Tens digit of hour		Units digit of hour	
⑥	1	Tens digit of minute		Units digit of minute	
⑦	1	Tens digit of second		Units digit of second	
⑧	1	Ø Ø Ø		Day of the week	

All items are defined in BCD notation. The
calendar/clock is updated when the last parameter
byte is received. Any item whose bits are set to
all 1s is not updated (this allows partial

11-18

update). The day of the week is automatically

updated within the range 0 through 6.

Since the 7508 CPU makes no check on the set

data, the contents of the calendar/clock will not

be guaranteed if logically invalid data is given.

Bit 7 of send data bytes is always set to 1. The

time is represented in 24-hour system.

(15) Read time

Code: 07H

Send data: None.

Receive data: 8 bytes

Function: Reads the contents of the 7508

calendar/clock. The format of the send data is

shown below. All items are specified in BCD.

	bit 7 4	3 bit 0
①	0 0 0 0	Tens digit of year
②	0 0 0 0	Units digit of year
③	Tens digit of month	Units digit of month
④	Tens digit of day	Units digit of day
⑤	Tens digit of hour	Units digit of hour
⑥	Tens digit of minute	Units digit of minute
⑦	Tens digit of second	Units digit of second
⑧	0 0 0 0	Day of the week

(16) Set alarm

Code: 19H

Send data: 6 bytes

Receive data: None.

Function: Sets the month, day, hour, minute, second, and day of the week for the alarm. The format of send data is as follows:

	bit 7		4	3		bit 0
①	1	Tens digit of month		Units digit of month		
②	1	Tens digit of day		Units digit of day		
③	1	Tens digit of hour		Units digit of hour		
④	1	Tens digit of minute		Units digit of minute		
⑤	1	∅ ∅ ∅		Units digit of second		
⑥	1	∅ ∅ ∅		Day of the week		

All items are specified in BCD. Items whose bits
are all 1s are "don't care." (Setting the minute
field to all 1s causes alarm interrupts to be
generated every minute.) The second must be set
in ten second increments.

The time is represented in 24-hour system. Bit 7
of send data bytes is always set to 1. Since the
7508 makes no check on the set data, the contents
of the alarm will not be guaranteed if logically
invalid data is sent to the 7508 CPU.

Command (19) must be executed after this command
to enable the alarm function.

(17) Read alarm

Code: 09H

Send data: None.

Receive data: 6 bytes

Function: Reads the currently set alarm time
  (month, day, hour, minute, second, and day of the
  week).

The send data must be specified in the following format:

	bit 7 … 4	3 … bit Ø
①	Tens digit of month	Units digit of month
②	Tens digit of day	Units digit of day
③	Tens digit of hour	Units digit of hour
④	Tens digit of minute	Units digit of minute
⑤	Ø Ø Ø Ø	Units digit of second
⑥	Ø Ø Ø Ø	Day of the week

All items are specified in BCD notation.

(18) Disable alarm

Code: 29H

Send data: None.

Receive data: None.

Function: Disables alarm interrupts to the Z80 CPU.

(19) Enable alarm

   Code: 39H

   Send data: None.

   Receive data: None.

   Function: Enables alarm interrupts to the Z80 CPU.
      It must be executed at least once after the alarm
      is set by command (16). (This command may be
      executed before setting the alarm time.)

(20) Read battery voltage

   Code: 0CH

   Send data: None.

   Receive data: 1 byte

   Function: Reads the main battery voltage in
      digital form. The relationship between the
      voltage and receive data is shown in the figure
      on the next page.

(21) Read temperature

   Code: 1CH

   Send data: None.

   Receive data: 1 byte

   Function: Reads the current temperature in digital
      form. The relationship between the temperature and
      receive data is shown in the figure on the next
      page.

Battery voltage and receive data (7508)

- The receive data is linearly proportional to the battery voltage.
- The combined errors including the scatters of resistance and standard voltage are shown below.
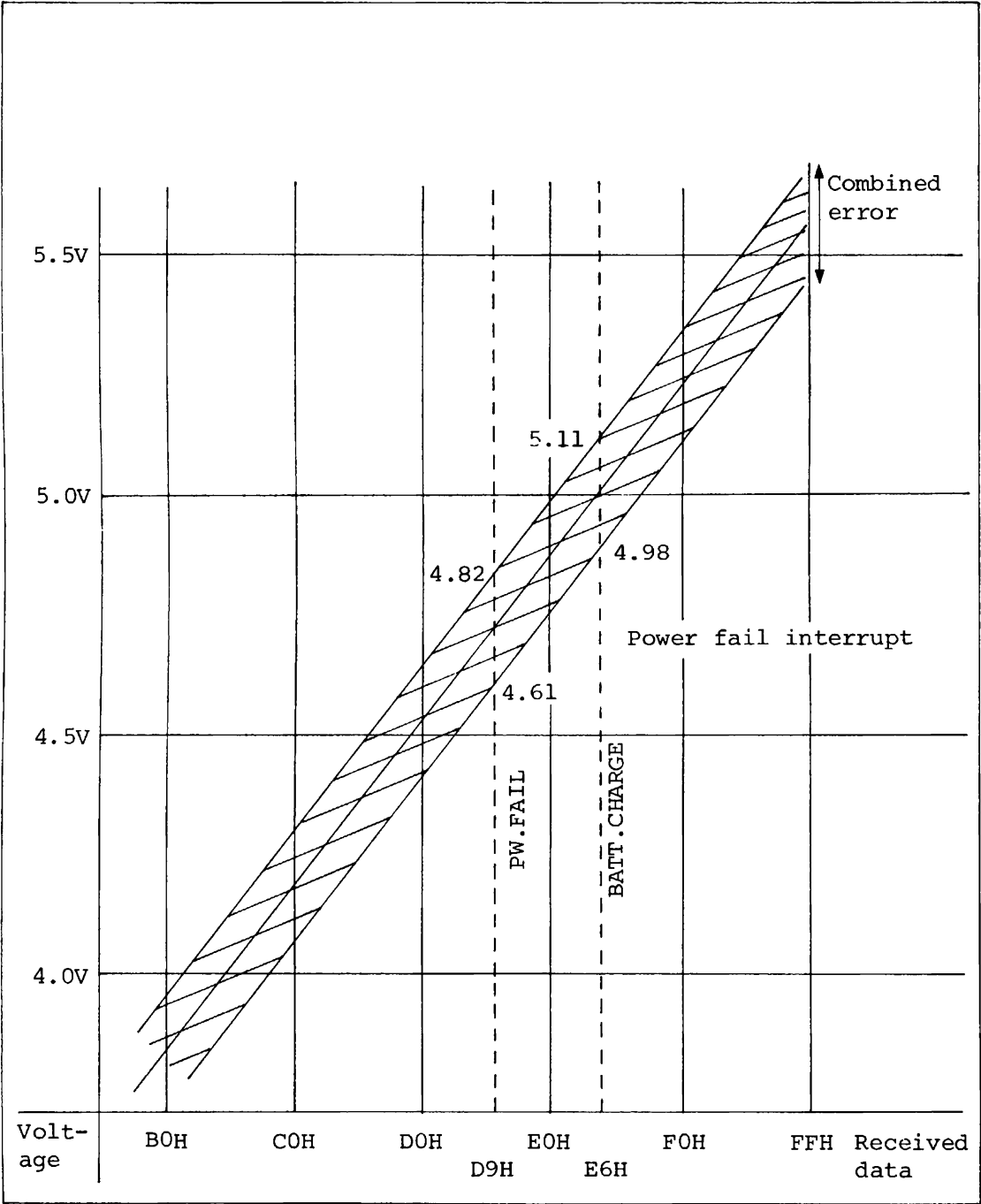
The graph below shows the correspondence between the temperature and the received data with combined errors.
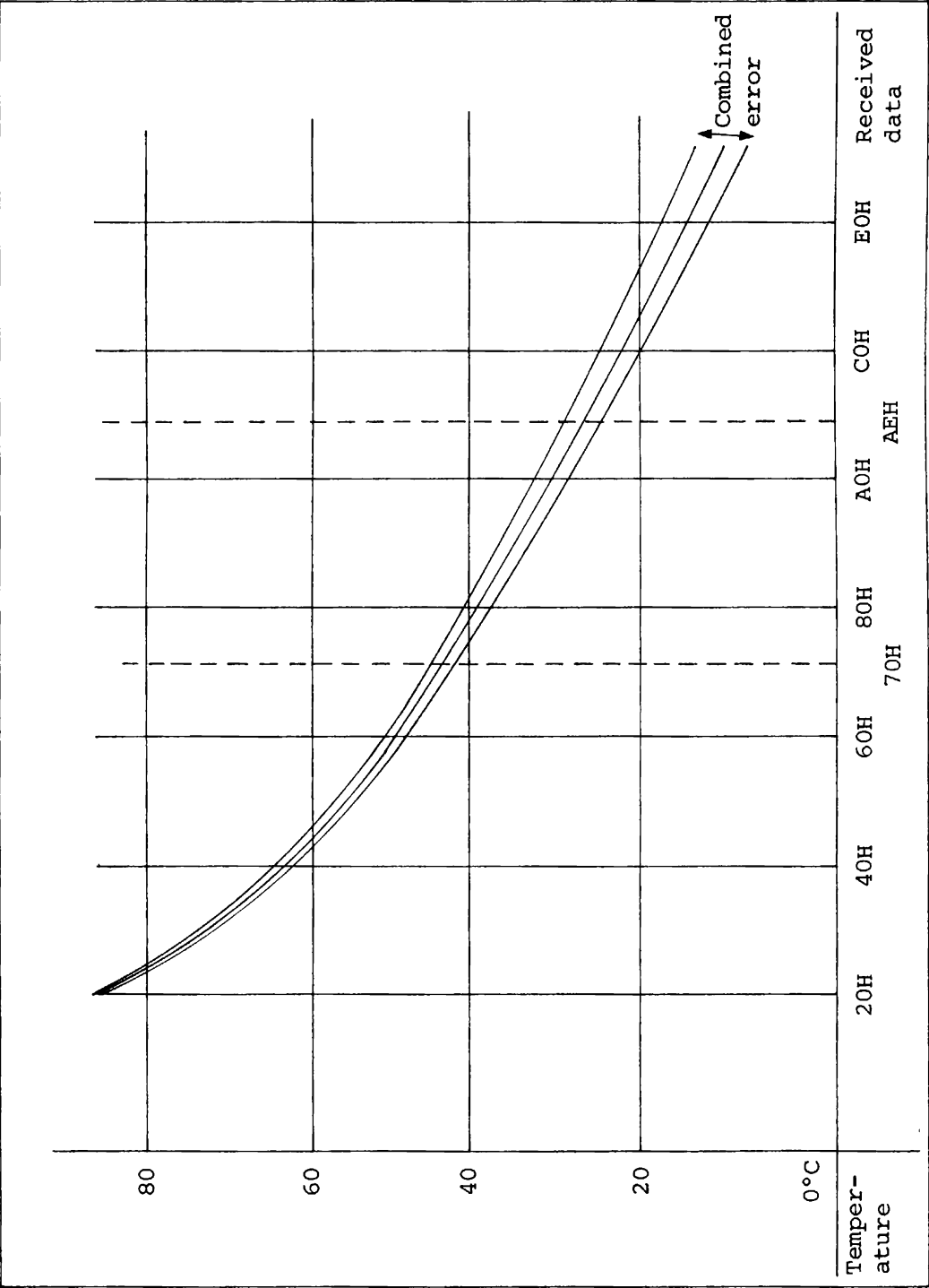
(22) Read analog jack 1

Code: 2CH

Send data: None.

Receive data: 1 byte

Function: Converts analog data from the analog

data input jack to digital data.

The received data has the following

format:

```
   bit                    bit
    7                      Ø
  ┌─────────────┬─────────┐
  │ │ │ │ │ │ │ │ │ │ │ │ │
  └─────────────┴─────────┘
   _____/ _____/
         │        Undefined
         │
         ▼
```
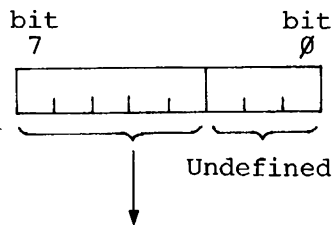
The highest order 6 bits represent analog data
voltages Ø to +2 V. Each bit represent an
increment of $2V \div 2^6 \cong 32$ mV (resolution).
These bits are set to all 1s when a voltage
higher than +2V is input. They are set to Ø
when a negative voltage is input.

(23) Read analog jack 2

Code: 3CH

Send data: None.

Receive data: 1 byte

Function: Converts analog data from the bar code

reader jack to digital data. The format

of the received data is the same as that

for the Read Analog jack 1 command (22) .

(24) 7508 power-on reset

Code: 0FH

Send data: None.

Receive data: None.

Function: Resets (initializes) the 7508 sub-CPU.

Note: This command is not used by application

programs.

(25) Read DIP-SW

Code: 0AH

Send data: None.

Receive data: See Chapter 15.

Function: Reads the settings of the DIP switches on

the main unit rear panel. See Chapter 15 for the

functions of the individual DIP switches.

(26) Set power failure detect voltage

Code: 0BH

Send data: 1 byte

Receive data: None.

Function: Defines the voltage at which power

fail interrupts are to be generated to the Z-80

CPU. A power fail interrupt is generated when

battery voltage falls below this voltage. The

relationship between the send data and the

set voltage is the same as that shown in the

figure on page 11-24.

(27) Set full charge voltage

   Code: 1BH

   Send data: 1

   Receive data: None

   Function: Defines the voltage at which full
      charging for the back-up battery is to be started.
      The relationship between the send data and the
      set voltage is the same as that shown on
      page 11-24.

      There are tow ways to charge batteries: full
      charging (a battery is fully charged in eight
      hours) and trickle charging (a battery is fully
      charged in 30 hours). When the AC adapter is
      connected, full charging is performed for the
      first eight hours and then switched to trickle
      charging. Battery voltage drops gradually if the
      MAPLE is kept in operation during a tickle
      charge. This command is used to set the voltage
      at which full charging is to be started.

(28) Read POWER or TRIGGER switch

    Code: 08H

    Send data: None.

    Receive data: 1 byte

    Function: Reads the state of the POWER switch
       (slide switch) on the main unit right-side panel
       or the state of the analog data input connector
       trigger.

    The receive data has the following format:

```
 bit                bit
  7                  0
 ┌──┬──┬──┬──┬──┬──┬──┬──┐
 │0 │0 │0 │0 │0 │0 │  │  │
 └──┴──┴──┴──┴──┴──┴──┴──┘
                     │  └──────► Indicates the POWER        1: ON
                     │           switch state.
                     │                                      0: OFF
                     │
                     └─────────► Indicates the trigger      1: ON
                                 switch state.
                                                            0: OFF
```
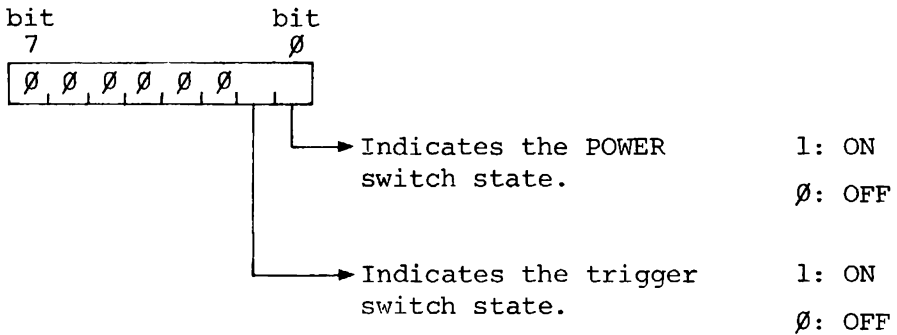
Chapter 12   Using 8251A Programmable Serial Controller

The MAPLE uses a CMOS type RS-232C controller
equivalent to the Intel 8251A Programmable Serial
Controller. Refer to an 8251A manual for details on the
functions and specifications for the 8251A. This
chapter explains how to interface the Z80 CPU with the
8251A and how to control the transmitter/receiver
clocks that determine the bit rate of the RS-232C
interface.

12.1 Interface between the Z80 and the 8251A

The Z80 CPU can exchange commands and data with the
8251A through the I/O port addresses 0CH and 0D.

Port	Read Mode	Write Mode
0CH	8251A receive data	8251A send data
0DH	8251A status	Command to 8251A

The Z80 CPU uses no special sequence when accessing I/O
port addresses 0CH and 0D. Refer to an 8251A manual for
the meanings of the 8251A status and commands.

12.2 Controlling the 8251A Transmitter/Receiver Clocks

The 8251A needs a clock with a frequency 1x, x16, or x64 times higher than the bit rate when it is to be used in asynchronous mode. The MAPLE controls the clock generator for that clock by outputting a command data into bits 4-7 of port 00H. The bit rate factor (X1, X16, or X64) can be specified by outputting a mode command into the 8251A.

To output data into port 0, use the following sequence:

```
      LD    A, (CTLR1)
      AND   0FH
      OR    [      ]        ;Bits 7-4: Select one of the
                             clocks listed in the table on the
                             next page.
                             Bits 3-0: Set to all zeros.
      LD    (CTLR1), A
      OUT   (0), A
```

CTLR1:   Overseas version = 0F0B0H

          Japanese-language version = 0ED90H

          CTLR1 is the data output to port 0.

Port Ø				Clock		RS-232C Baud Rate			
						X 16		X64	
bit 7	bit 6	bit 5	bit 4	Transmit (TxC)	Receive (RxC)	Transmit (Tx)	Receive (Rx)	Transmit (Tx)	Receive (Rx)
Ø	Ø	Ø	Ø	1.74545 KHz	←	110	←		
Ø	Ø	Ø	1	2.4K	←	150	←		
Ø	Ø	1	Ø	4.8K	←	300	←		
Ø	Ø	1	1	9.6K	←	600	←	150	←
Ø	1	Ø	Ø	19.2K	←	1200	←	300	←
Ø	1	Ø	1	38.4K	←	2400	←	600	←
Ø	1	1	Ø	76.8K	←	4800	←	1200	←
Ø	1	1	1	153.6K	←	9600	←	2400	←
1	Ø	Ø	Ø	19.2K	1.2K	1200	75		
1	Ø	Ø	1	1.2K	19.2K	75	1200		
1	Ø	1	Ø	307.2K	←	19200	←	4800	←
1	1	Ø	Ø	3.2K	←	200	←		

Note: Some MAPLE overseas versions do not support a bit rate of 2ØØ (32 KHz clock).

Chapter 13   6301 Slave CPU Operations

13.1 Functions

The 6301 slave CPU controls the following six types of devices:

- RAM

- LCD display

- Microcassette drive

- Serial communication interface

- ROM capsule

- Speaker

The slave CPU runs on its own control programs so that the Z80 CPU need only issue commands to the slave system to control the above devices. The Z80 commands to the slave CPU are detailed in Section 13.5.

```
┌──────┐   ┌───────┐        ┌──────────┐  ┌──────────┐  ┌──────────────┐
│      │ → │       │        │   RAM    │  │   LCD    │  │ Microcassette│
│      │   │       │        │          │  │ display  │  │    drive     │
│ Z8Ø  │   │ SLAVE │────────┴────↕─────┴──┴────↕─────┴──┴──────↕───────┴──
│      │ ← │       │        ┌────↕─────┐  ┌────↕─────┐  ┌──────↕───────┐
│      │   │       │        │ Serial   │  │   ROM    │  │              │
└──────┘   └───────┘        │communica-│  │ capsule  │  │   Speaker    │
                            │tion      │  │          │  │              │
                            │interface │  │          │  │              │
                            └──────────┘  └──────────┘  └──────────────┘
```
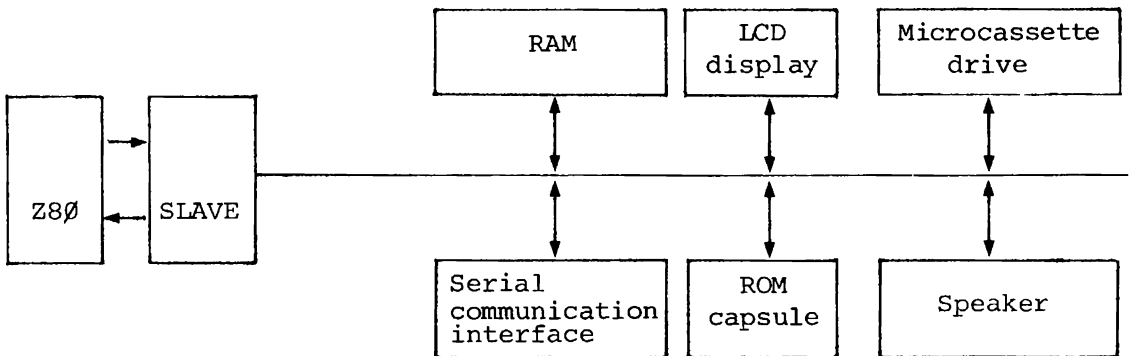
(1) RAM

The slave CPU has 6K bytes of RAM which is located at slave system memory locations 8000H and higher. This RAM contains the VRAM and external character areas; it is mainly used for screen-oriented operations. The RAM is also loaded with the control programs for the slave CPU. The memory configuration of this RAM is shown on page 13-3.

(2) LCD display

1) Theory of operation

The data to be displayed on the LCD is fetched by the LCD controller from RAM for display; the Z80 CPU need only place the display data into the specified RAM area.

There are two screen modes: the character and graphics modes. In the character mode, the LCD controller receives 1-byte character codes from the character screen area and searches the character generator in the controller for the corresponding (6 x 7 dots) font for display on the LCD. For user defined characters, it searches the user defined characters area at the beginning of the RAM for display.

Character screen mode   Graphics screen mode

```
8ØØØH  ┌────────────────────┐ ─ ─ ─ ─ ─ ─ ┌────────────────────┐  8ØØØH
       │ User defined       │             │ User defined       │
       │ characters area    │             │ characters area    │
       │ (256 bytes)        │             │ (256 bytes)        │
81ØØH  ├────────────────────┤ ─ ─ ─ ─ ─ ─ ├────────────────────┤  81ØØH
       │                    │             │ Character screen   │
       │ Character screen   │             │ (64Ø bytes)        │
       │ (512 bytes)        │             ├────────────────────┤  838ØH
       │                    │             │                    │
       │                    │             │ Graphics screen    │
       │                    │             │ (384Ø bytes)       │
       │                    │             │                    │
       │                    │             │                    │
       │                    │             ├────────────────────┤  928ØH
       │                    │             │ System buffer      │
       │                    │             │ (27Ø bytes)        │
       │                    │             ├────────────────────┤  938EH
       │                    │             │ Character flash    │
       │                    │             │ block difinition   │
       │                    │             │ area (2 bytes)     │
95ØØH  ├────────────────────┤             ├────────────────────┤  939ØH
       │ System buffer      │             │ Graphics flash     │
       │ (27Ø bytes)        │             │ block definition   │
96ØEH  ├────────────────────┤             │ area (433 bytes)   │
       │ Character flash    │             ├────────────────────┤  9541H
       │ block definition   │             │ Graphics user      │
       │ area (121 bytes)   │             │ defined char-      │
9687H  ├────────────────────┤             │ acters area        │
       │ User area          │             │ (558 bytes)        │
9765H  │ (222 bytes)        │             ├────────────────────┤  9765H
       ├────────────────────┤ ─ ─ ─ ─ ─ ─ │ System area (155   │
       │ System area (155   │             │ bytes)             │
       │ bytes)             │             │                    │
       └────────────────────┘             └────────────────────┘
```

Japanese-language kanji characters are displayed in the
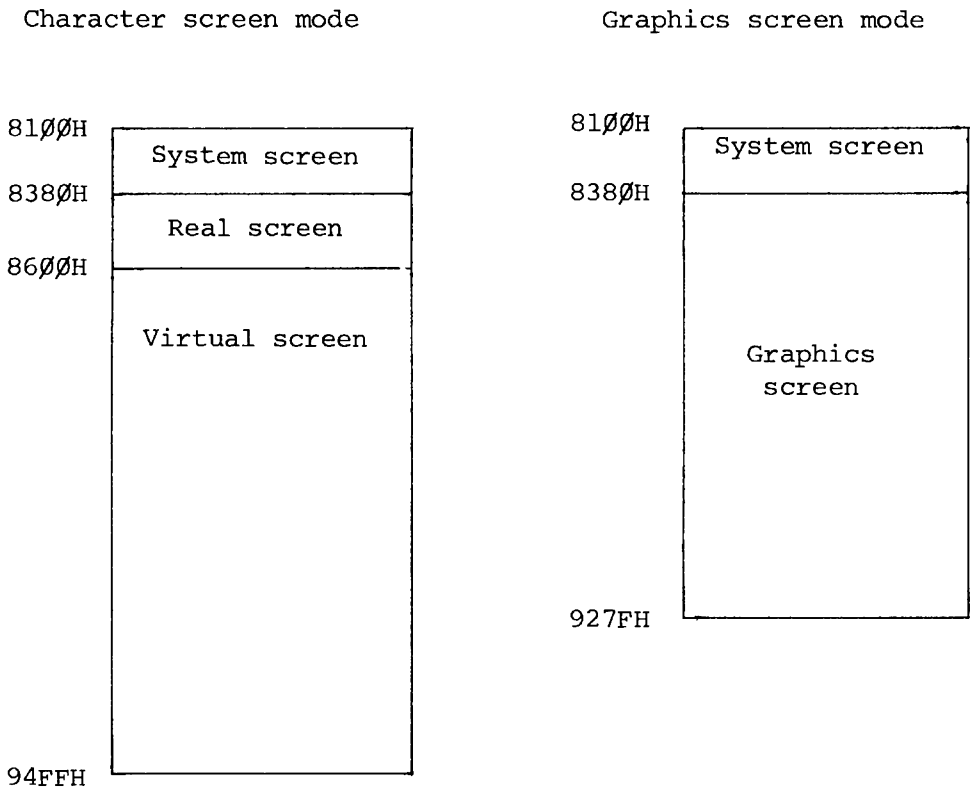graphics screen mode.

In the graphics mode, each single dot on the LCD is associated with a single bit in the graphics screen area in the RAM.

2) Screen configuration

The LCD screen configuration is shown below.

Character screen mode   Graphics screen mode

```
81ØØH  ┌─────────────────┐      81ØØH  ┌─────────────────┐
       │  System screen  │             │  System screen  │
838ØH  ├─────────────────┤      838ØH  ├─────────────────┤
       │   Real screen   │             │                 │
86ØØH  ├─────────────────┤             │                 │
       │                 │             │                 │
       │                 │             │    Graphics     │
       │ Virtual screen  │             │     screen      │
       │                 │             │                 │
       │                 │      927FH  └─────────────────┘
       │                 │
       │                 │
       │                 │
       │                 │
94FFH  └─────────────────┘
```

The system screen is used by the OS and not available to application programs. In the system screen, data is always displayed using character codes, independent of the screen mode. It has the same size as the LCD (80 x 8 = 640 bytes).

The real screen is as large as the 80 x 8 dot LCD and holds a portion of the virtual screen. Its image is displayed on the LCD by the LCD controller.

Since each dot on the LCD corresponds to a single data bit in the graphics screen and the LCD measures 480 dots by 64 dots, the LCD requires 480 dots x 64 dots ÷ 8 bits/byte = 3840 bytes of RAM. The graphics screen holds characters in bit image so that they may be displayed simultaneously with graphics data. See Chapter 6, "CONOUT" for further information.

3) Screen-related RAM areas

a) User defined characters area

The user can define characters in this area. It holds up to 32 user defined characters from 0E0H through 0FFH. User defined characters are defined using the ESC - 0E0H sequence via CONOUT.

b) Character screen area

The character screen area is used to store codes of characters to be displayed. In the screen configuration diagram in 2) above, all screen areas in the character screen mode and the system screen area in the graphics screen mode are used as the character screen area.

c) Graphics screen area

The graphics screen area is used to store data to be
displayed in dot image. Its size is zero in the
character screen mode.

The system buffer is used by the system when exchanging
data with the MCT, serial communication interface, and
Z80 CPU.

e) Character flash block definition area

The character flash block definition area contains
character data to be flashed on the character screen.
Three bytes are required to flash a single character (2-
byte address and the character to be flashed) so a
maximum of 40 characters can be flashed at a time in the
character mode. Flashing is not available in the
graphics screen mode (since the graphics screen holds
character data in bit image form, it can use the
graphics flash definition area that is described below).
A character flash definition area can be defined by
issuing the command code 31H to the slave CPU.

f) Graphics flash block definition area

The graphics flash block definition area contains
graphics data to be flashed on the graphics screen.
Three bytes are required to flash 8 consecutive dots

(2-byte address and 1-byte graphics data to be flashed)
so a maximum of 144 dots can be flashed at a time in
the graphics mode. This feature is available only in
the graphics screen mode. Since the graphics screen
holds graphics data in bit image form, it can use the
graphics flash definition area that is described
below. A graphics flash block definition area can be
defined by issuing the command code 21H to the slave
CPU.

g) Graphics user defined character definition area
The graphics user defined character definition area
contains user defined character data for display on
the graphics screen. This area can be used only in
the graphics mode. A graphics user defined character
definition area can be defined by issuing the command
code 20H to the slave CPU.

h) User area
The user area (the remainder of the graphics user
defined character definition area in the graphics
screen mode) is not used by the slave CPU programs.
The user may load programs into this area for
execution.

i) System area
The system area is used by the slave CPU programs.

(3) Microcassette drive

The application program can perform various operations
on the microcassette using commands described in Section
13.5. Since, however, files on the microcassette drive
is controlled all by MTOS, an error would occur if an
application program issued a microcassette command
directly to the slave CPU while it was performing an I/O
operation to the microcassette drove. No application
program is therefore allowed to control the
microcassette drive using the slave subsystem.

The application program may, however, exert direct
control over the microcassette drive when playing back
audio data tape (playback is not controlled by MTOS
because no file operation is involved). The following
precaution must be observed when driving a microcassette
drive directly from an application program:
Precaution: Be sure to execute motor stop (4AH) and head
            off (42H) commands in this sequence before
            issuing an MCT-related command to the slave CPU.
To playback a microcassette tape, execute the following
commands sequentially:

  - Head On   (command code 41H)

  - Play   (command code 48H)

(4) Serial communication interface

The MAPLE CP/M supports communication via a serial interface. Its slave CPU supports EPSP for controlling communication with external floppy disk drives.

(5) ROM capsule

The MAPLE can read data from the ROM capsules attached to the rear panel of the main unit. See Chapter 15 for the ROM memory map for the ROM capsules.

(6) Speaker

The MAPLE provides several functions to drive the speaker.

13.2 Data Backup

Since the MAPLE supports continue mode operations, the slave CPU must preserve the system status before power is turned off and restores it immediately after power is turned on again. To achieve this, the slave CPU provides the following functions:

-    Battery backup of 6K RAM. The RAM data is always maintained whether MAPLE power is turned on or off.
-    Saving and restoring the contents of CPU registers. These functions are used only by the OS POWER ON/OFF subroutines and not accessible to application programs (so they are not covered in this manual).

The Z80 and slave CPU communicate using the procedure
described below.
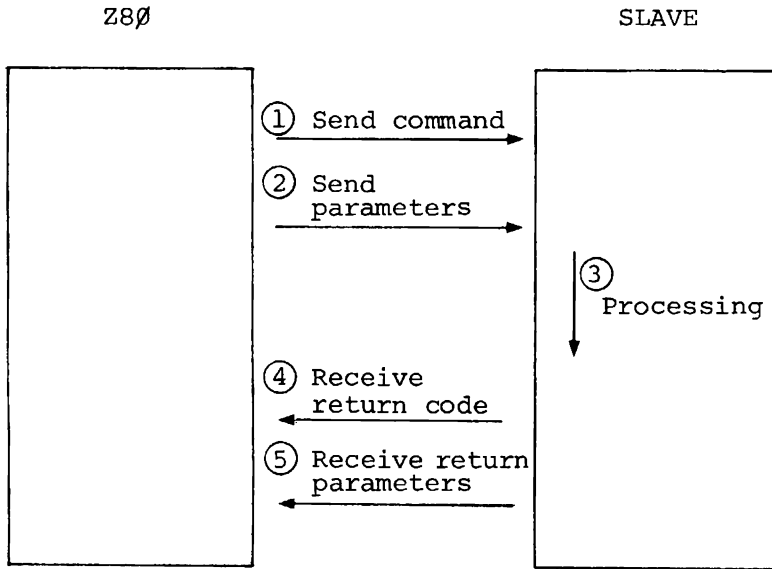
The command (1) and return code (4) are 1 byte long and
must always be issued. The presence and length of
parameters (2 and 4) may differ depending on the
command. The minimum parameter length is 0, that is, no
parameter. If the received return code contains a
nonzero value (abnormal termination), the calling
program will receive no return parameter.

Every sequence of steps 1 through 5 or 4 above must
always be concluded; that is, no subsequent command must
be issued before step 5 (or 4) is completed (the system
will hang up if attempted).

The application program communicate with the slave CPU through the BIOS SLAVE function (WBOOT + 72H) (see Chapter 4, "BIOS Call" for details).

A command packet for turning on the speaker is shown below.

SPON:		
	LD	A,0FFH
	LD	(SLVFLG),A
	LD	DE,072H
	LD	HL,(1)
	ADD	HL,DE
	LD	DE,PACK1 ◄─────── Communication packet address
	JP	(HL) ◄─────── BIOS call (Call SLAVE.)

Communication packet		
PACKE1:	DW	SENDSV ◄─────── Send packet address
	DW	SENDLN ◄─────── Send packet length
	DW	RCVSV ◄─────── Receive packet address
	DW	RCVLNG ◄─────── Receive packet length
SENDSV:	DE	72H,80H ◄─────── Send packet (command + send parameters)
SENDLN:	EQU	2 ◄─────── Send command + parameters
RCVSV:	DS	1 ◄─────── Receive packet
RCVLNG:	EQU	1 ◄─────── Receive packet length (return code only)

Note: Two-byte address parameters must be sent to the slave CPU in higher-order-byte-first-sequence. This is because the slave CPU (6301) handle the 2-byte data the higher-order-byte-first as opposed to the Z80 CPU.

13.4   Slave CPU Commands

The rest of this chapter describes the slave CPU commands arranged by device.

How to interpret the command table

- The send packet length is the length of the command (one byte) plus its send parameters (SP1, SP2, ... SPn in 1)).
- The receive packet length is the length of the return code (one byte) plus receive parameters (RP1, RP2, ... RPn in 2)).
- The return code in 3) identifies the type of the status code returned by the slave CPU. The return code table is given at the end of this chapter.
- Commands are represented in hexadecimal notation. All parameters are treated as binary data.
- "Physical screen" refers to the "real screen" used in previous sections.

(1) RAM

This subsection explains the monitor program which controls program execution and RAM access.

Table 3-1   RAM commands

Code	Function
00	Read data
01	Write data
02	Execute routine

(1-1) Read Data (ØØ)

    1) Send parameter

       SP1: Address (high)

       SP2: Address (low)

    2) Receive parameter

       RP1: Read data

    3) Return code

       RCDØØ

    4) Function

       Reads the data at the location designated by SP1
       and SP2.

    5) Note

       An attempt to read data at a location lower than
       8ØH will destroy the system.

(1-2) Write Data (Ø1)

    1) Send parameter

        SP1: Address (high)

        SP2: Address (low)

        SP3: Write data

        SP4: Operation {1: AND, 2: OR, 3: XOR

                    Others: Store only

    2) Receive parameter

        None.

    3) Return code

        RCDØØ

    4) Function

        Performs the specified operation on the data

        and places the results at the address

        designated by SP1 and SP2.

(1-3) Execute Routine (Ø2)

    1) Send parameter

       SP1: Address (high)

       SP2: Address (low)

    2) Receive parameter

       None.

    3) Return code

       ?

    4) Function

       Transfers control to the address designated by
       SP1 and SP2. The following registers are loaded
       with as follows:

       IX (97F4, 97F5)

       A (97F3)

       B (97F2)

       C (97F1)

(2) Screen

This subsection describes the screen-related commands used to control the graphics and character screens.

Table 3-2   Screen Commands

Code	Function
10	Defines screen mode.
11	Turns on/off LCD.
12	Selects screen.
13	Reads screen pointer.
14	Sets screen pointer.
15	Defines number of lines.
16	Defines cursor mode.
17	Reads cursor position.
18	Defines cursor position.
19	Starts/Stops control block flashing.
1A	Clears screen.
1B	Reads character font.
20	Defines user graphics character.
21	Defines graphics screen block flashing data.
22	Draws character font on graphics screen.

23	Draws user defined graphics character on graphics screen.
24	Reads graphics screen data.
25	Displays data on graphics screen.
26	Moves graphics screen block.
27	Sets point.
28	Reads point.
29	Draws line.
30	Defines user character.
31	Defines character screen block flashing data.
32	Reads window pointer.
33	Sets window pointer.
34	Reads character screen data.
35	Displays data on character screen.
36	Moves character screen block.

(2-1) Define screen mode (10)

    1) Send parameter

        SP1 SP2: Character screen starting address

        SP3 SP4: Graphics screen starting address

        SP5 SP6: Character flash block starting address

        SP7 SP8: Graphics flash block starting address

      SP9 SP10: Graphics user defined character buffer

               starting address

          SP11: Number of lines on character screen

          SP12: User defined character starting code

      SP13 SP14: Communication buffer address

          SP15: LCD status

          SP16: Screen status

    2) Receive parameter

       None.

    3) Return code

       RCD00

    4) Function

       Defines the configuration of the VRAM. The user

       cannot use this command since the VRAM

       configuration is defined by the OS as described

       on page 13-3. The OS display routine (CONOUT)

       will not function normally if the VRAM

       configuration is defined by the user.

(2-2) Turn On/Off LCD (11)

    1) Send parameter

       SP1: ON/OFF switch (00: OFF, Others: ON)

    2) Receive parameter

       None.

    3) Return code

       RCD00

    4) Function

       Turns on or off the LCD display. This command
       only specifies whether data is to be displayed
       on the LCD and do not affect the contents of
       VRAM at all.

(2-3) Select Screen (12)

    1) Send parameter

       SP1: Screen select code (ØØ: Graphics screen,

                                   Others: Character screen)

    2) Receive parameter

       None.

    3) Return code

       RCDØØ

    4) Function

       Selects the screen to be displayed on the
physical screen. The screen is set to the
window pointer position when the character
screen is selected and set to the top when
the graphics screen is selected.

(2-4) Read Screen Pointer (13)

    1) Send parameter

       None.

    2) Receive parameter

       RP1: Address (high)

       RP2: Address (low)

    3) Return code

       RCD00

    4) Function

       Reads the starting address of the VRAM area whose contents are currently displayed on the physical screen.

    5) Note

       The LCD controller accesses VRAM only between addresses 8000H and 97FFH. Therefore, the highest order three bits of the pointer is insignificant and always set to 100.

(2-5) Set Screen Pointer (14)

   1) Send parameter

      SP1: Address (high)

      SP2: Address (low)

   2) Receive parameter

      None.

   3) Return code

      RCD00

   4) Function

      Sets the physical screen starting address
      designated by SP1 and SP2.

      The valid addresses are 8000H to 97FFH.

      The LCD controller displays 640 bytes of data

      starting at the address pointed to by the

      pointer in the character screen mode. It

      displays 3840 bytes of data in the graphics

      mode.

(2-6) Define Number of Lines (15)

  1) Send parameter

     SP1: 00: 8 lines

          Others: 7 lines

  2) Receive parameter

     None.

  3) Return code

     RCD00

  4) Function

     Defines the number of lines to be displayed in
     the character screen mode. If this command is
     issued when the graphics screen mode is
     selected, the value defined by the command
     becomes valid when the screen mode is switched
     to character screen.
     8 or 7 lines refer to the number of lines to be
     displayed on the LCD. The OS always uses 8-
     line-per-screen mode. Characters are displayed
     with a wider spacing in 7-line mode than in 8-
     line mode.

(2-7) Define Cursor Mode (16)

    1) Send parameter

        SP1: Bit 7 - Bit 3: Ø

            Bit 2: Cursor font (Ø: Under line, 1: Block)

            Bit 1: Cursor blink (Ø: OFF, 1: ON)

            Bit Ø: Cursor ON/OFF (Ø: OFF, 1: ON)

    2) Receive parameter

    None.

    3) Return code

    RCDØØ

    4) Function

    Defines the cursor mode to be used in the
character screen mode according to the
settings of bits Ø to 2 of SP1.
This command is valid only in the character
screen mode. In the graphics mode, the cursor
is always displayed in the underlined blink mode.

(2-8) Read Cursor Position (17)

    1) Send parameter

       None.

    2) Receive parameter

       RP1: Cursor X (0 - 79)

       RP2: Cursor Y {7-line mode (0 - 6)

                    8-line mode (0 - 7)

    3) Return code

       RCD00

    4) Function

       Reads the cursor position on the physical screen

       (in the character screen mode).

       The upper left corner of the screen is taken as

       coordinates (0,0).

(2-9) Set Cursor Position (18)

    1) Send parameter

       SP1: Cursor X (0 - 79)

       SP2: Cursor Y {7-line mode (0 - 6)

                     8-line mode (0 - 7)

    2) Receive parameter

       None.

    3) Return code

       RCD00

    4) Function

       Sets the cursor position on the physical screen
       (in the character screen mode).

       The cursor position must be within the maximum
       line and column numbers of the physical screen.

(2-10) Start/Stop Control Block Flashing (19)

   1) Send parameter

   SP1: 00: Stop

       Others: Start

          (01, ... FF)

          The unit is 100 msec.

   2) Receive parameter

   None.

   3) Return code

   RCD00

   4) Function

   Turns on and off block flashing on the current

   screen.

   If no flash block is specified, this command

   does nothing even if block flashing is turned

   on.

   The block to be flashed changes as the screen

   mode is changed.

   Block flashing is specified by issuing a command

   21H (in the graphics screen mode) or 31H (in the

   character screen mode).

(2-11) Clear Screen (1A)

1) Send parameter

SP1: Screen to be cleared

00: Graphics screen

Others: Character screen

SP2: Clear code

SP3: Starting line

SP4: Number of lines to be cleared.

2) Receive parameter

None.

3) Return code

RCD00

4) Function

Fills the specified number of lines (specified by SP4) starting at the line designated by SP3 on the screen area specified by PS1 with the code specified by SP2.

This command is invalid when the specified screen does not exist.

SP4 must not be set to 0. (Specifying 0 as SP4 value will destroy the system.)

The screen specified in SP1 refers to the screen shown in the memory map on page 13-3. The relationship between the line number and the screen type is shown below.

Character screen mode   Graphics screen mode

Line   Line
number   0   8Ø columns   number   0
         ≶   System screen   Character ≶   System screen
          7   screen   7
          8   0
         ≶   Real screen
         15
         16   Graphics   Graphics screen
                 Virtual screen   screen
                                                   7
         63

13-31
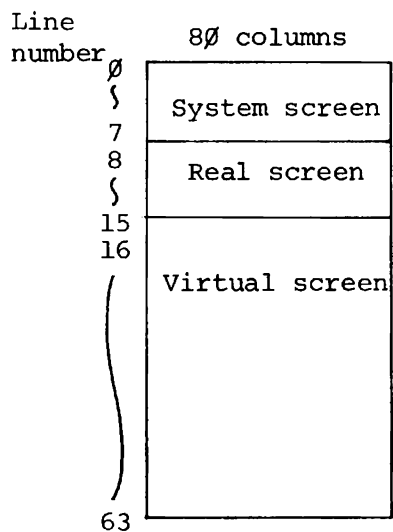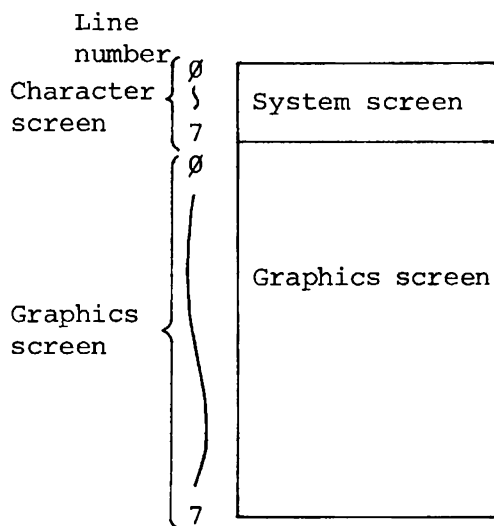
(2-12) Read Character Font (1B)

    1) Send parameter

       SP1: Character code

    2) Receive parameter

       RP1: Font data 1

       RP2: Font data 2

       RP3: Font data 3

       RP4: Font data 4

       RP5: Font data 5

       RP6: Font data 6

       RP7: Font data 7

       RP8: Font data 8

    3) Return code

       RCD00

    4) Function

       Reads the font of the specified character from
       the LCD controller or user defined character
       area.

    5) Font image

Font data 1

Font data 8

MSB   LSB
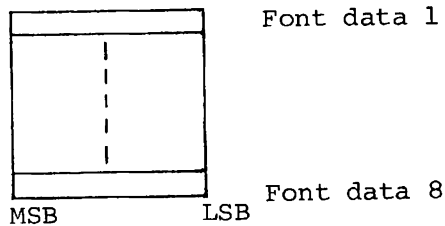
    Note: The correspondence between the character
      codes and fonts is shown in Chapter 20.

(2-13) Define User Defined Graphic Character (20)

   1) Send parameter

      SP1: Definition mode

            00: Clear user defined character area

            Others: User defined character code

      SP2: Hight (size along x-axis)

      SP3: Width (size along y-axis)

      SP4: Data 1

       .

       .

      SPn: Data n-3

   2) Receive parameter

      None.

   3) Return codes

      RCD00

      RCD04

   4) Function

      Defines a graphics user defined character or

      clears the user defined character area. The

      definition code must not be 00.

      Even if a user defined character is defined

      with a code which has already been assigned to

      another character, the character to be

      displayed on the graphics screen is the one

originally defined with that code (though the
later definition is accepted.)

The sizes along x- and y-axes must be within
the range x*y<=255. Specifying a size larger
than that will cause an error. The definition
will also be invalidated if too small x and y
values are specified.

The user defined character block must be
specified in byte units.

Data is placed in the area in row-first order.

Example:

        x:3, y:4, Byte count: 12

        Data 1   Data 2   Data 3

        Data 4   Data 5   Data 6

        Data 7   Data 8   Data 9

        Data 10   Data 11   Data 12

(2-14) Define Graphics Screen Block Flashing Data (21)

    1) Send parameter

        SP1: Number of blocks

        SP2: x-coordinate of the first block

        SP3: y-coordinate of the first block

        SP4: First block blink data

         •

         •

        SPn-2: x-coordinate of the kth block

        SPn-1: y-coordinate of the kth block

        SPn: kth block blink data

    2) Receive parameter

        None.

    3) Return codes

        RCD00

        RCD04

    4) Function

        Defines the coordinates of the blocks to flash
        and the blink data on the graphics screen. The
        block flashing data must not exceed the
        specified block. The maximum number of blocks
        that can be specified is 144.

(2-15) Draw Character Font on Graphics Screen (22)

1) Send parameter

SP1: x-coordinate (high)

SP2: x-coordinate (low)   (0 - 479)

SP3: y-coordinate   (0 - 63)

SP4: Character code   (0 - FF)

2) Receive parameter

None.

3) Return code

RCD00

4) Function

Reads the font corresponding to a character code
specified in SP4 from the character generator
and draws the font on the screen using the
coordinates designated by SP1, SP2, and SP3 as
the origin.

Graphics coordinates must be specified in bits.
The portion of the font extending beyond the
right edge of the screen is entered into the
screen from the left edge.

(2-16) Draw User Defined Character on Graphics Screen (23)

    1) Send parameter

        SP1: x-coordinate (0 - 59)

        SP2: y-coordinate (0 - 63)

        SP3: Graphics user defined character

    2) Receive parameter

        None.

    3) Return codes

        RCD00

        RCD05

    4) Function

        Draws the block of the user defined character
        code specified in SP3 starting at the graphics
        screen coordinates specified in SP1 and SP2.
        Graphics user defined character can be defined
        by issuing a command 20H.

(2-17) Read Graphics Screen Data (24)

1) Send parameter

   SP1: x-coordinate (0 - 59)

   SP2: y-coordinate (0 - 63)

   SP3: Byte count (0: 256, 1: 1, ... FF: 255)

2) Receive parameter

   SP1: Data 1

   .

   .

   SPn: Data n

3) Return codes

   RCD00

   RCD04

4) Function

   Reads the number of bytes specified in SP3

   in row-first order starting at the coordinates

   on the graphics screen designated by SP1 and

   SP2. Garbage data is returned for the portion

   of data which is specified in SP3 and which is

   out of the screen.

   The number of receive parameters (n) is the same

   as that of data count specified in the third

   send parameter.

8 dots × 6∅ bytes (48∅ dots)

8 dots

1
dot

64 dots
in total
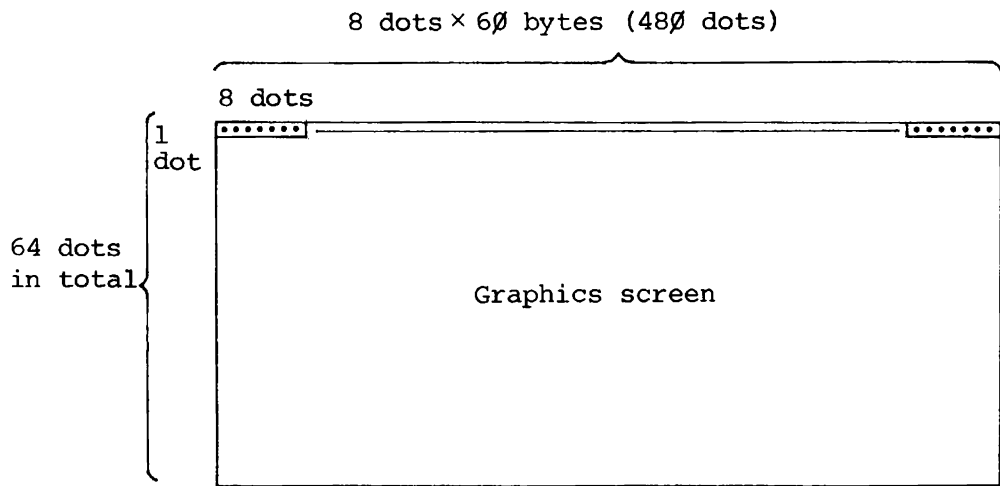
Graphics screen

(2-18) Display Data on Graphics Screen (25)

    1) Send parameter

        SP1: x-coordinate (0 - 59)

        SP2: y-coordinate (0 - 63)

        SP3: Hight (size along x-axis)

        SP4: Width (size along y-axis) }   x*y<=255

        SP5: Operation   00: Store   01: AND

                          02: OR   03: EOR

        SP6: Data 1

        •

        •

        SPn: Data n-5

    2) Receive parameter

        None.

    3) Return codes

        RCD00

        RCD04

    4) Function

        Stores the block of data specified in SP3 and
        SP4 at the specified coordinates on the graphics
        screen. Data is sent in row-first order.

          Data 1   Data 2

          Data 3   Data 4

          Data 5   Data 6   Size: x = 2, y = 3

This command performs the specified operation on
the write data and writes the result as the new
data.

When store operation is specified, the command
writes data as is, without masking with a mask
pattern. Drawing is stopped when the write data
overflows the screen.

Both height (x) and width (y) of the block must
be 1 or larger. The product of x and y must be
255 or less.

(2-19) Move Graphics Screen Block (26)

   1) Send parameter

      SP1: Source x-coordinate (0 - 59)

      SP2: Source y-coordinate (0 - 63)

      SP3: Size along X-axis

      SP4: Size along Y-axis

      SP5: Destination x-coordinate (0 - 59)

      SP6: Destination y-coordinate (0 - 63)

   2) Receive parameter

      None.

   3) Return codes

      RCD00

      RCD04

   4) Function

      Moves the block designated by SP1, SP2, SP3, and
      SP4 on the graphics screen starting at the
      coordinates specified by SP5 and SP6.

(2-20) Define Point (27)

    1) Send parameter

        SP1: x-coordinate (high) ⎫
        SP2: x-coordinate (low) ⎬ (0 - 479)

        SP3: y-coordinate   (0 -63)

        SP4: Operation

            01: OFF   02: ON   03: Complement

    2) Receive parameter

    None.

    3) Return codes

    RCD00

    RCD04

    4) Function

    Performs the specified operation on the dot at
    the specified graphic screen coordinates.

(2-21) Read Point (28)

1) Send parameter

SP1: x-coordinate (high)

SP2: x-coordinate (low) } (Ø - 479)

SP3: y-coordinate (Ø - 63)

2) Receive parameter

RP1:   ØØ: off   Others: ON

3) Return codes

RCDØØ

RCDØ4

4) Function

Reads the state of the dot at the specified graphics screen coordinates.

(2-22) Draw Line (29)

    1) Send parameter

        SP1: Starting x-coordinate (high) ⎫
                                          ⎬ (0 - 479)
        SP2: Starting x-coordinate (low) ⎭

        SP3: Starting y-coordinate (high) ⎫
                                          ⎬ (0 - 63)
        SP4: Starting y-coordinate (low) ⎭

        SP5: Ending x-coordinate (high) ⎫
                                        ⎬ (0 - 479)
        SP6: Ending x-coordinate (low) ⎭

        SP7: Ending y-coordinate (high) ⎫
                                        ⎬ (0 - 63)
        SP8: Ending y-coordinate (low) ⎭

        SP9: Operation vector (high)   ⎫ 0: No operation
                                          ⎬
        SP10: Operation vector (low)   ⎭ 1: Operation

        SP1: Point mode   01: OFF   02: ON   03: Complement

    2) Receive parameter

        None.

    3) Return code

        RCD00

    4) Function

        Draws a line on the graphics screen between the
        starting and ending coordinates. The points are
        displayed in the point mode only when the MSB of
        the operation vector is set to 1. The operation
        vector is rotated to the left at setting every
        single dots.

The top left corner of the screen is set to
(0, 0). The horizontal and vertical axes are
defined as x-axis and y-axis, respectively.
Data at the points outside the screen is not
displayed.

(2-23) User Defined Character (30)

    1) Send parameter

       SP1: Character code (E0H-FFH)

       SP2: Data 1

         ·

         ·

       SP9: Data 8

    2) Receive parameter

       None.

    3) Return codes

       RCD00

       RCD06

    4) Function

       Defines an user defined character in the user
       defined character area. Invalid codes not in
       the range 0E0H to 0FFH are ignored.

       "A", for example, is defined with the following
       data:

	bit 7	6	5	4	3	2	1	bit 0
Data 1	0	0	0	0	1	1	0	0
Data 2	0	0	0	1	0	0	1	0
Data 3	0	0	1	0	0	0	0	1
Data 4	0	0	1	0	0	0	0	1
Data 5	0	0	1	1	1	1	1	1
Data 6	0	0	1	0	0	0	0	1
Data 7	0	0	1	0	0	0	0	1
Data 8	0	0	0	0	0	0	0	0

(2-24) Define Character Screen Block Flashing Data (31)

    1) Send parameter

        SP1: Number of blocks

        SP2: x-coordinate of the first block (0 - 79)

        SP3: y-coordinate of the first block (0 - 63)

        SP4: First block blinking data

        .

        .

        SPn-2: x-coordinate of the kth block

        SPn-1: y-coordinate of the kth block

        SPn: kth block blinking data

    2) Receive parameter

        None.

    3) Return codes

        RCD00

        RCD04

    4) Function

        Specifies the coordinates of the block to flash
        and the blink data on the character screen.
        Specifying coordinates outside the screen will
        make all definitions in this command invalid.
        Up to 40 blocks can be defined. All blocks will
        be cleared when the screen mode is altered.

(2-25) Read window pointer (32)

    1) Send parameter

       None.

    2) Receive parameter

       RP1: x-coordinate (0 - 79)

       RP2: y-coordinate (0 - 63)

    3) Return code

       RCD00

    4) Function

       Reads the character screen coordinates at which
       display is to begin.
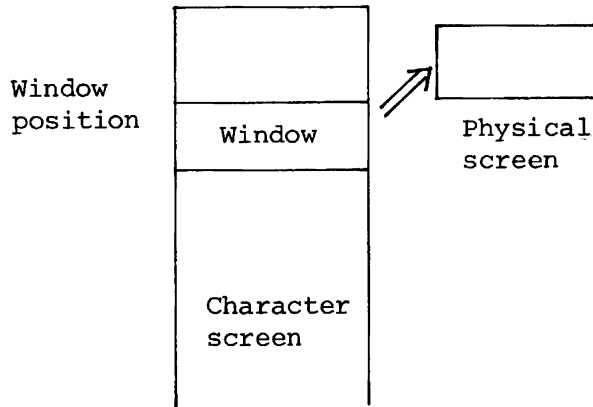
(2-26) Set Window Pointer (33)

    1) Send parameter

       SP1: x-coordinate (∅ - 79)

       SP2: y-coordinate (∅ - 63)

    2) Receive parameter

       None.

    3) Return code

       RCD∅∅

    4) Function

       Defines the window position on the character

       screen. This command is valid if issued

       in the graphics screen mode (it will be executed

       when the screen is switched to character mode).

(2-27) Read Character Screen Data (34)

    1) Send parameter

        SP1: x-coordinate (0 - 79)

        SP2: y-coordinate (0 - 63)

        SP3: Byte count (0:256, 1:1, ... FF:255)

    2) Receive parameter

        RP1: Data 1

          •

          •

        RPn: Data n

    3) Return code

        RCD00

    4) Function

        Reads the specified number of data bytes from
        the character screen in row-first order. If
        the parameters are specified to read data
        beyond the character screen area, data other
        than display data is read.
        The number of receive parameters (n) is the same
        as the number specified in the third send
        parameter.

(2-28) Display Data on Character Screen (35)

    1) Send parameter

        SP1: Starting x-coordinate (0 - 79)

        SP2: Starting y-coordinate (0 - 63)

        SP3: Byte count (1: 1, ... FF: 255)

        SP4: Data 1

         •

         •

        SPn: Data n-3

    2) Receive parameter

        None.

    3) Return code

        RCD00

    4) Function

        Displays the specified data in row-first order

        on the character screen starting at the

        specified coordinates. The portion of data

        not fitting in the screen is ignored.

        Byte count must not be set to 0.

        The relationship between the y-coordinate values and

        screen types is as follows:

          0 - 7: System screen

          8 - 15: Real screen

          16 - 63: Virtual screen

(2-29) Move Character Screen Block (36)

    1) Send parameter

        SP1: Source x-coordinate (0 - 79)

        SP2: Source y-coordinate (0 - 63)

        SP3: Hight (size along x-axis)

        SP4: Width (size along y-axis)

        SP5: Destination x-coordinate (0 - 79)

        SP6: Destination y-coordinate (0 - 63)

    2) Receive parameter

        None.

    3) Return codes

        RCD00

        RCD04

    4) Function

        Moves the block specified by SP1, SP2, SP3, and
        SP4 on the character screen from the coordinates
        designated by SP5 and SP6.

(3) Microcassette

This subsection describes the commands for driving the microcassette drive.

Table 3-3   Microcassette commands

Code	Command
40	Read microcassette status
41	Head on
42	Head off
43	Rewind n counts
44	Fast forward n counts
45	Rewind
46	Fast forward
47	Slow rewind
48	Play
49	Record
4A	Stop
4B	Read write protect pin
4C	Read counter
4D	Set counter
51	Write data and non-stop
52	Write data and stop
53	Read data and non-stop

54	Read data and stop
55	Set write protect area pointer.
56	Reset write protect area pointer

(3-1) Read Microcassette Status (40)

    1) Send parameter

       None.

    2) Receive parameter

       RP1: Microcassette status

         Bit 7: Head position (0: OFF 1: ON)

         Bit 6: Motor   (0: Stop 1: Move)

         Bit 5: Rewind   (0: No 1: Wind)

         Bit 4: FF   (0: No 1: FF)

         Bit 3: Play   (0: No 1: Play)

         Bit 2: Record   (0: No 1: Record)

         Bit 1:

         Bit 0: Write protect area set flag

                      (0: Reset 1: Set)

    3) Return code

       RCD00

    4) Function

       Reads the status of a microcassette drive.
       The write protect area setting flag (bit 0) is
       set to 1 when a command 55H is executed and set
       to 0 (reset) when a command 56H is executed.

(3-2) Head On (41)

    1) Send parameter

        None.

    2) Receive parameter

        None.

    3) Return codes

        RCD00

        RCD07

    4) Function

        Turns the read/write head on.

        This command must be issued while the

        microcassette drive is not in motion.

(3-3)   Head Off (42)

    1) Send parameter

       None.

    2) Receive parameter

       None.

    3) Return codes

       RCD00

       RCD07

    4) Function

       Turns the read/write head off.

       This command must be issued while the

       microcassette drive is not in motion.

(3-4)   Rewind n counts (43)

1) Send parameter

SP1: Counter value (high-order)

SP2: Counter value (low-order)

2) Receive parameter

None.

3) Return codes

RCD00

RCD07

RCD08

4) Function

Moves tape backward the number of tape count

specified in SP1 and SP2.

5) Note

If tape is rewound 100 tape counts from count

1000, for example, the tape at position 900 will

come under the read/write head.

The head is automatically turned off in the

rewind mode.

(3-5) Fast Foward n Counts (44)

1) Send parameter

SP1: Counter value (high-order)

SP2: Counter value (low-order)

2) Receive parameter

None.

3) Return codes

RCD00

RCD07

RCD08

4) Function

Winds the tape forward by the counts specified
by SP1 and SP2.

(3-6) Rewind (45)

    1) Send parameter

       None.

    2) Receive parameter

       None.

    3) Return codes

       RCD00

       RCD07

    4) Function

       Rotates the motor in the reverse direction (rewind).

(3-7)　Fast Foward (46)

    1) Send parameter

       None.

    2) Receive parameter

       None.

    3) Return code

       RCD00

    4) Function

       Rotates the motor in the forward direction.

(3-8) Slow Rewind (47)

    1) Send parameter

       None.

    2) Receive parameter

       None.

    3) Return codes

       RCD00

       RCD07

    4) Function

       Rotates the motor in the reverse direction at a

       low speed.

(3-9) Play (48)

    1) Send parameter

       None.

    2) Receive parameter

       None.

    3) Return code

       RCD00

    4) Function

       Rotates the motor in the play mode.

       The read signal is placed on the read line if

       the microcassette drive is in the head on

       state.

(3-1Ø) Record (49)

    1) Send parameter

       None.

    2) Receive parameter

       None.

    3) Return code

       RCDØØ

    4) Function

       Rotates the motor in the record mode.

       Data on the tape will be erased if the

       read/write head is on.

(3-11)   Stop (4A)

    1)   Send parameter

       None.

    2)   Receive parameter

       None.

    3)   Return code

       RCD00

    4)   Function

       Stops the motor to rotate.   Head position does
       not move.

(3-12)   Read Write Protect Pin (4B)

    1) Send parameter

       None.

    2) Receive parameter

       RP1:   ØØ: Protected

             Others: Not protected

    3) Return code

       RCDØØ

    4) Function

       Used to check whether the current microcassette

       tape is write protected.

(3-13)   Read Counter (4C)

    1) Send parameter

       None.

    2) Receive parameter

       RP1: Counter value (high-order)

       RP2: Counter value (low-order)

    3) Return code

       RCD00

    4) Function

       Reads the current value of the 16-bit counter.

(3-14)   Set Counter (4D)

    1) Send parameter

       SP1: Counter value (high-order)

       SP2: Counter value (low-order)

    2) Receive parameter

       None.

    3) Return code

       RCD00

    4) Function

       Sets the counter.

(3-16) Write Data and Non-stop (51)

    1) Send parameter

        SP1: Data length (high-order) ⎫ 0000:65536,

        SP2: Data length (low-order) ⎭ 0001:1, ...

        SP3: Pointer to block ID number

        SP4: Data 1
         ·
         ·

        SPn: Data n-3

    2) Receive parameter

        RP1: Block end counter (high-order)

        RP2: Block end counter (low-order)

    3) Return codes

        RCD00

        RCD07

        RCD08

        RCD09

    4) Function

        Writes the specified number of data bytes onto
        tape. A 00H in SP3 identifies the first block
        that is written for the current block and a 01H
        the second write for the block. This byte is
        automatically incremented by the slave CPU
        during processing. The issuing program must
        set this byte in the send data to 0FFH.

(3-17) Write Data and Stop (52)

    1) Send parameter

        SP1: Data length (high-order)

        SP2: Data length (low-order)

        SP3: Pointer to block ID number

        SP4: Data 1

         •

         •

        SPn: Data n-3

    2) Receive parameter

        RP1: Block end counter (high-order)

        RP2: Block end counter (low-order)

    3) Return codes

        RCD00

        RCD07

        RCD08

        RCD09

    4) Function

        This command performs the same function as Write
        Data and Non-stop except that it stops the tape
        after writing data.

(3-18)   Read Data and Non-stop (53)

    1) Send parameter

       SP1: Data length (high-order)

       SP2: Data length (low-order)

       SP3: Block ID code

    2) Receive parameter

       RP1: Block start counter (high-order)*

       RP2: Block start counter (low-order)

       RP3: Data 1

         •

         •

       RPn: Data n-2

    3) Return codes

       RCD00

       RCD07

       RCD08

       RCD10

       RCD11

       RCD11-1

    4) Function

       Reads the specified bytes of data from tape.
       If an error is found in a block, this command
       returns a return codes RP1 and RP2, and starts
       reading the next block. If the specified block

is read normally, the command terminates

reading of the block without stopping the

motor.

The block ID code is a control code which is

used to identify the beginning of a block.

No check is made on the block ID if 3FH ('?') is

specified in SP3.

*: Counter value after preamble is read.

(3-19) Read Data and Stop (54)

    1) Send parameter

       SP1: Data length (high-order)

       SP2: Data length (low-order)

       SP3: Block ID code

    2) Receive parameter

       RP1: Block start counter (high-order)*

       RP2: Block start counter (low-order)

       RP3: Data 1

         •

         •

       RPn: Data n

    3) Return codes

       RCD00

       RCD07

       RCD08

       RCD10

       RCD11

       RCD11-1

    4) Function

       Reads the specified bytes of data from tape.
       This command has the same functions as Read Data
       and Non-stop except that it stops the motor
       after reading the specified block.

(3-20)   Set Write Protect Area Pointer (55)

   1) Send parameter

      SP1: Counter value (high-order) ⎫ Protect area

      SP2: Counter value (low-order) ⎬   pointer

   2) Receive parameter

      None.

   3) Return code

      RCD00

   4) Function

      Decrements the write protect pointer by 1 every

      time the tape counter increments by 1 at each

      read or write operation and terminates either

      read or write when the pointer reaches 0.

      This command is used to stop processing on a

      cassette at a desired count. MTOS uses this

      function to erase a length of tape specified in

      the tape count parameter.

      Whether processing has been stopped or not can

      be identified by examining bit 6 of the status

      information returned by command 40H. If this

      bit is 0, the motor has been stopped (that means

      processing has been terminated).

      Whether the pointer is set or not can also be

identified by checking bit 0 of the status

information returned by command 40H.

(3-21)   Reset Write Protect Area Pointer (56)

    1) Send parameter

       None.

    2) Receive parameter

       None.

    3) Return code

       RCD00

    4) Function

       Makes the value of the write protect area
       pointer invalid. After this command, the user
       can write on the tape without considering the
       write protect area pointer value.

(4) Serial I/O and serial communication

This subsection describes the commands for serial I/O and serial communication. Refer to Chapter 15 for the serial communications protocol.

Table 3-4 Serial I/O, serial communications commands

Code	Command
60	Read serial I/O port status
61	Set serial port bit rate
62	Serial input
63	Serial output
64	Send with header
65	Receive data

(4-1) Read Serial I/O Status (60)

    1) Send parameter

    None.

    2) Receive parameter

    RP1: Serial I/O status

       Bit 7: Receive register status

                       (0: Empty 1: Full)

       Bit 6: Overrun framing error

                       (0: No 1: Error)

       Bit 5: Transmit register status

                       (0: Full 1: Empty)

       Bit 4: Control out   (0: Low 1: High)

       Bit 3: Control in   (0: Low 1: High)

    3) Return code

    RCD00

    4) Function

    Reads the serial I/O status value.

(4-2) Set Serial Port Bit Rate (61)

    1) Send parameter

        SP1: Specifies the bit rate, general purpose
input port check bit, and general
purpose output port level.

        Bits 0 and 1: Sets bit rate.

	Bit 1	Bit 0
38400	0	0
4800	0	1
600	1	0
150	1	1

        Bit 5: General purpose output level.

    2) Receive parameter

    None.

    3) Return code

    RCD00

    4) Function

    Sets the serial port bit rate and defines the
level of the general purpose output port.

(4-3)　Serial Input (62)

    1) Send parameter

       None.

    2) Receive parameter

       RP1: Data

    3) Return codes

       RCD00

       RCD13

    4) Function

       Reads one character from the serial port.

(4-4) Serial Output (63)

1) Send parameter

SP1: Send data

2) Receive parameter

None.

3) Return code

RCD00

4) Function

Sends one character to the serial port.

(4-5) Send Data with Header (64)

    1) Send parameter

      SP1: Receive data flag

            (00: Receive no data after sending.

             01: Receive data after sending.)

      SP2: FMT

      SP3: DID

      SP4: SID = 22H

      SP5: FNC

      SP6: SI2

      SP7: Data 1

         •

         •

      SPn+7 : Data n

    2) Receive parameter

    (Valid only when SP1 $\geq$ 1.)

    RP1: Header information $\begin{pmatrix} 00: \text{Header} \\ 01: \text{No header} \end{pmatrix}$

        When SP1=00

            FMT

            DID

            SID

            FNC

            SIZ

RP2: Data 1

.

.

RPk: Data n

3) Return codes

RCD00

RCD12

RCD13

RCD14

4) Function

Sends data with a header to the serial port according to the EPSP protocol. The command also receives data with a header if the receive data flag is set to 1. It terminates processing immediately when an error is detected.

(4-6) Receive Data with Header (65)

1) Send parameter

None.

2) Receive parameter

RP1: Header information $\left(\begin{array}{l} \text{00: Header} \\ \text{01: No header} \end{array}\right)$

RP2: FMT

RP3: DID

RP4: SID $\qquad$ Valid only when RP1 = 00

RP5: FNC

RP6: SIZ

RP7: Data 1

.

.

RPn+5: Data n

3) Return codes

RCD00

RCD12

RCD13

RCD14

4) Function

Receives data with a header from the serial

port. FMT through SIZ are omitted if the header

information is 01.

(5) ROM and speaker

This subsection deals with the commands for PROM
capsules and speakers.

Table 3-5 PROM and speaker commands

Code	Command
70	Turn on/off PROM capsule power
71	Read PROM data
72	Turn on/off speaker power
73	Beep
74	Melody

(5-1) Turn On/Off PROM capsule power (70)

   1) Send parameter

      SP1: 00: OFF

            Others: ON

   2) Receive parameter

      None.

   3) Return code

      RCD00

   4) Function

      Turns on or off ROM capsule power according to the

      parameter specification.

   5) Note:

         It will take one second after a power-on

         before the ROM capsule is ready.

(5-2) Read Data (71)

    1) Send parameter

        SP1: Power on flag

            (00: OFF   Others: ON)

        SP2: Data address (high-order)

        SP3: Data address (low-order)

        SP4: Data count (0: 256, 1: 1, ... FF: 255)

    2) Receive parameter

        (Valid only when Return code = 00.)

        RP1: Data 1

         •

         •

        RPn: Data n

    3) Return code

        RCD00

    4) Function

        Reads the specified bytes of data starting at
        the specified data address. Cartridge #0 is
        selected when the MSB of the data address is 0.
        Cartridge #1 is selected when the MSB is 1.
        When the power off flag is set to 00, this
        command turns off ROM capsule power after
        reading the data. ROM capsule power must be
        turned on before this command is executed.

(5-3) Turn On or Off Speaker Power (72)

    1) Send parameter

       SP1: Power on/off switch

          Bit 7 0: OFF 1: ON

    2) Receive parameter

       None.

    3) Return code

       RCD00

    4) Function

       Turns on or off the power to the speaker

       amplifier.

(5-4)   Beep (73)

1) Send parameter

SP1: Period (high-order)

SP2: Period (low-order)

SP3: Duration (in 100 msec units)

2) Receive parameter

None

3) Return code

RCD00

4) Function

Sounds the speaker at the specified frequency
(reciprocal of period) for the specified length
of time.   The unit of period is 3.2 μsec.

(5-5) Melody (74)

    1) Send parameter

        SP1: Repeating count

            (0: 256, 1: 1, ... FF: 255)

        SP2: Data address (high-order)

        SP3: Data address (low-order)

    2) Receive parameter

    None.

    3) Return code

    RCD00

    4) Function

    Sounds the speaker according to the data read

    from the specified address.

      Data format

        Duration 1 (1)

        Period 1 (2)

            •

            •

        Duration n (1)

        Period n (2)

          00

Return Codes (in decimal)

1) RCD00 (SYS)

    Code: 00

    Explanation: Normal termination.

2) RCD01 (SYS)

    Code: 01

    Explanation: Break acknowledged.

3) RCD02 (SYS)

    Code: 02

    Explanation: Command error.

        A command code (00H - 7FH) not defined by

        the system was issued.

4) RCD03 (SYS)

    Code: 03

    Explanation: Communications error.

        A command was issued while sending or

        receiving data or sending another command.

5) RCD04 (LCD)

    Code: 11

    Explanation: Invalid size specification.

        The specified data did not fit in the

        screen.

6) RCD05 (LCD)

 Code: 12

 Explanation: Undefined graphics user defined

       character.

8) RCD06 (LCD)

 Code: 13

 Explanation: Invalid user defined character

     An attempt was made to specify a code

     other than the codes under which user

     defined characters are defined.

9) RCD07 (MCT)

 Code: 41

 Explanation: Head error.

     The head did not function normally.

10) RCD08 (MCT)

 Code: 42

 Explanation: Tape stopped during processing.

11) RCD09 (MCT)

   Code: 43

   Explanation: Write protect error.

   An attempt was made to write on the tape

   having no write protect tab.

12) RCD10 (MCT)

   Code: 44

   Explanation: Data error.

   A pulse of an invalid width was received and

   the logical state of the pulse (1 or 0) could

   not be determined.

13) RCD11 (MCT)

   Code: 45

   Explanation: CRC error.

14) RCD12 (ESPS)

   Code: 61

   Explanation: Linking unsuccessful.

15) RCD13 (ESPS)

   Code: 62

   Explanation: Communication error.

   An overrun framing error occurred.

16) RCD14 (ESPS)

Code: 63

Explanation: Time over.

17) RCD15 (BEEP)

Code: 71

Explanation:

A BEEP or MELODY command was issued
before the execution of the preceding
BEEP or MELODY command was completed.

18) RCD11-1 (MCT)

Code: 46

Explanation: Block mode error.

A block with an invalid block identifier
was read.

Chapter 14   MTOS/MIOS Operations

The MAPLE is provided as standard with a microcassette
drive which is placed under control of MTOS/MIOS.   This
chapter describes the function and operation of MAPLE
MTOS/MIOS.

14.1   MTOS/MIOS

14.1.1   MTOS/MIOS Outline
The MAPLE supports I/O operations on microcassette tape
(MCT) at two levels of control programs (MTOS and
MIOS).   MTOS is an operating system which corresponds
to CP/M BDOS and manages files on MCT automatically.
MTOS communicates with CP/M through unique interface
modules and built-in CP/M.   MTOS, on the other hand,
corresponds to CP/M BIOS and controls the microcassette
drive motor and MCT on a record basis.

In MAPLE CP/M, MCT is assigned to drive H:.
Application programs can handle MCT as an ordinary
external drive, being unaware of the difference between
ordinary disk drives and MCT. More specifically, the
user need only select drive H with the BDOS drive
select function or place 08H at the beginning of the
FCB to control the MCT drive.   Subsequently, all drive
commands are directed to the MCT drive automatically

14-1

and the application program can control the MCT drive
using ordinary BDOS drive functions.

MIOS calls are functionally equivalent to BIOS calls.
Application programs can execute a MIOS function by
loading the function number in a specific register and
calling a specific BIOS entry (see 14.4 for details).

Each MCT has a directory file at its beginning. It 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 a memory location called
the directory work area (RAM directory). The results
of operations on files on MCT are managed also in the
RAM directory. Accordingly, when an MCT file is
updated, the contents of the RAM directory must also be
rewritten onto MCT. Reading in this directory file
into memory is called a mount and writing it onto MCT
is called a remove. File positioning is accomplished
using the tape counter to enable accessing to the MCT.

The figure below shows the relationship of MTOS/MIOS to CP/M.
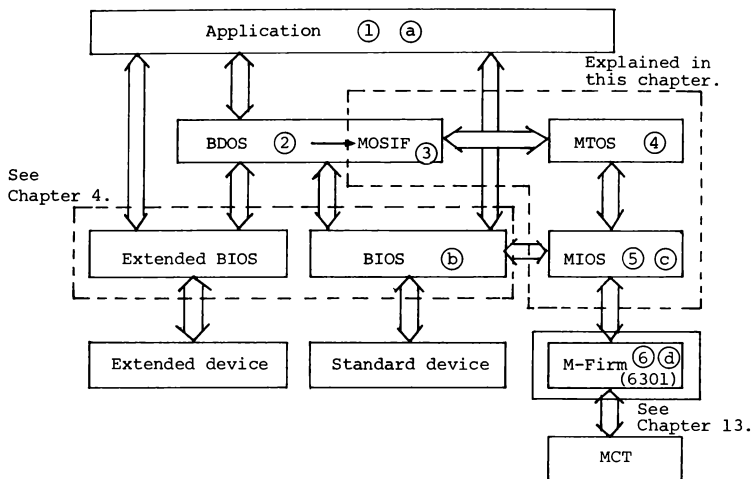
MTOS/MIOS control flow

(1) MTOS

1)   Calls BDOS (application program).

2)   Identifies the destination of call (standard BDOS or MCT).

3)   Calls MTOS.

4)   Calls MIOS.

5)   Calls slave CPU firmware control routine.

6)   Writes or reads to or from MCT.

(2) MIOS

a)   Calls BIOS (application program).

b)   Recognizes an MIOS call.

c)   Calls slave CPU firmware control routine.

d)   Writes or reads to or from MCT.

This chapter describes the steps 3), 4), and 5) (c))
above.

14.1.2 File Control

The structure of an MCT file is shown below.

i) Tape format

Directory file	File	File	- - - - - - - - -	File	

ii)
File
format

Header section	Date section	EOF section

Header block	Data block 1	Data block 2	- - - - - - -	Data block n	EOF block

iii) Block format

SYNC field	Pre-amble	Block ID field	Data field	Block check character field	Post-amble
10 bytes	2 bytes	4 bytes	256 bytes	2 bytes	2 bytes

Differ depending on block type.
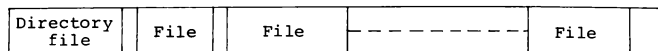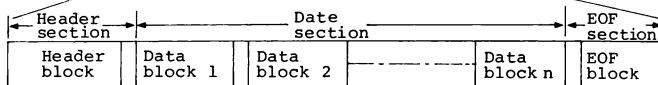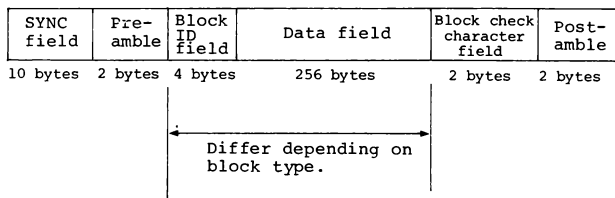
The MAPLE directory file can contain a maximum of 12
entries (this number may be modified by recording at
assembler level). Its contents are loaded into MAPLE
RAM for control of actual file accesses.

As shown in the above figures, each file consists of
header, data, and EOF sections.

Header section: Contains the information pertaining to
the organization of the file.

Data section: Contains actual file data. The data
section is normally made up of more than
one block. The number of blocks
corresponds to the file size.

EOF section: Is the last block identifying the end of
the file.

The MAPLE can write each block several times to
increase data reliability (normally, a block is written
twice. See Chapter 13 "6301 Slave CPU Operations" for
how to specify the number of writes).

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 read operations.

[1] Block format

Field	Description
SYNC field	80 bits of 0
Preamble	FF, AA (2 bytes)
Block ID field	4 bytes field.
	Byte 0: Block identifier "H": Header block "D": Data block "E": End of file block
	Bytes 1 and 2: 2-byte binary number (0000 ∿ FFFF) indicating the block number
	Byte 3: Block ID number indicating the ordinal number of writes.
Data field	Contains (256) data bytes.
BCC (Block Check Character) field	2-byte BCC characters computed using CRC (Cyclic Redundancy Check) technic. CCITT CRC method is used. Calculation covers from the block number to the BCC field itself.
Postamble	FF, AA (2 bytes)

The blocks have the format shown above. The block tables given in the subsequent subsections indicate the contents of the data field shown above. The length of the data field is fixed at 256 bytes.

Column From	Column To	Size	Item	Description
0	3	4	ID field	Identifies an "HDR1" header (ASCII).
4	11	8	File name	Contains the file name.
12	19	8	File type	Contains the file type.
20		1	Record type	"F": Fixed length "V": Variable "n": Fixed length length The same block is written n times (ASCII).
21		1	Block mode	" ": An intergap enough to stop the tape successfully is created. "s": Short gap (not long enough to stop the tape successfully).
22	26	5	Block length	Indicates the length of the block (ASCII number from "ØØØØØØ" to "65535").
* 27	28	2	Counter value	Indicates the tape count at which this file starts.
* 29		1	Free	
30		1	File attribute	File attribute
31		1		Free
32	37	6	Date of creation	Date on which this file is created. 2-byte ASCII code in month/day/year format.
38	43	6	Creation time	Time at which this file is created. 2-byte ASCII code in hour ("ØØ" – "23")/minute/second format.
44	49	6	Free	
50	51	2	Volume number	Tape volume number ("Ø1" –)
* 52	59	8	System name	Name of system in which the file is created.
* 60	61	2	System file number	File number by which the file is controlled in the directory. ØØØØ – FFFE
* 62	67	8	Password	System password
* 68	255		Free	

Note: System name is the name of the OS under which M-TOS runs.
*:   Items identified by an asterisk differ from those for the EPSON HC(X)-20.

2) Data block data field format

The block is the area which contains actual data bytes. There is no restriction on the data format except that the data field size is 256 bytes.

3) EOF block data field

| Column | | Size | Item | Description |
from	to			
0	3	4	ID field	Contains "EOF."
4	11	8	File name	Contains the file name.
12	19	8	File type	Contains the file type.
20*	21	2	File number	Contains the file number ("0000" to "FFFE").
22*	23	2	Counter	Contains the last counter value for the preceding block.
24*	255			Free

*: Differs from those for the HC(X)-20.

[3] Directory File Structure

The directory file comprises the following three blocks.

1) Directory ID (first block)
The directory block contains the information obtained
when the tape directory is created for the first time,
the information set up when the MCT is used last, and
the information pertaining to the MCT itself such the
current total block count and the total record count.

2) Directory (second and third blocks)
The directory contains the information pertaining to the
files on the MCT. One file entry is 32 bytes long and
one block of directory contains the information for 8
files. Two blocks must be reserved for the directory
because each MCT contains a maximum of 12 files.

3) RAM directory
The directory file is loaded into memory by the MOUNT
function for controlling subsequent accesses to MCT.
The directory in memory is called the RAM directory.
Its structure is identical to that on the MCT, which is
explained in 2).

i) Directory ID (first block)

Column	Size	Item	Description
0 - 7	8	Tape neme	Loaded with the tape name and volume number specified when the directory is created. This field is not used by MTOS. The field is set to EPSON SD by the system display function.
8 - 9	2	Volume number	
10 - 17	8	Password	Loaded with the system password created when the directory is created. This field is not used by MTOS.
18 - 23	6	Creation date	Loaded with the date at which the directory is created in the month/day/year format (ASCII).
24 - 29	6	Creation time	Loaded with the time at which the directory is created in the hour ($\emptyset\emptyset$-23)/minute/second format (ASCII).
30 - 35	6	Date last removed	Loaded with the time at which the tape is removed in the month/day/year format (ASCII).
36 - 41	6	Time last removed	Loaded with the time at which the tape is removed in the hour ($\emptyset\emptyset$-23)/minute/second format (ASCII).
42 - 43	2	Total mount count	Loaded with the total number of mounts. This field is initialized to $\emptyset$ and incremented for each mount operation.
44 - 45	2	Total block count	Loaded with the total number of blocks on the MCT except those for the directory file.
46 - 47	2	Total record count	Loaded with the total number of records on the MCT except those for the directory file.
48	1	Total file count	Loaded with the total number of files on the MCT.
49	1	System flag 1	Tape ID flag (used by the automatic MTOS identification feature. See 14.1.4).
50	1	System flag 2	Tape TOS controller (used by the automatic TOS identification feature. See 14.1.4).
51 - 52	2	Last file number	The file number of the last file on the MCT.
53 - 255	203	Free	

ii) Directory (second and third blocks)

Column	Size	Item	Description
0 - 1	2	File number	Is the number of the file by which MTOS manages files. MTOS manages files not by their file name but a file number assigned to the file because of the usage of Rename & erase CP/M and commands.
2	1	Presence flag	Indicates the file cataloged in this directory entry is valid or invalid. ØØH: Valid (file is present). ØFFH: Invalid (file is nonexistent).
3	1	File attribute 1	Indicates the logical characteristics of the file and the mode in which it is accessed during read operations.

<table>
<tr><th>Bit</th><th>Description</th></tr>
<tr><td>7</td><td>Ø: Nonstop mode access (Note)
1: Stop mode access</td></tr>
<tr><td>6
5
4
3</td><td>Not used.</td></tr>
<tr><td>2</td><td>Number of retries during read (Ø - 15).</td></tr>
<tr><td>1
Ø</td><td></td></tr>
</table>

The contents of bits 7 and 3-Ø are determined by the access attribute set up when the file is created.

Column	Size	Item	Description
4	1	File attribute 2	Not used.
5 - 6	2	Number of blocks	Total number of blocks in the file. A file containing only a header block is counted as ØØØØ. (Stored with higher order byte first.)
7 - 8	2	Number of records	Total number of records in the file. The first 128-byte record in the first block is counted as record ØØØØ. No record exists in the header and EOF blocks. (Stored with higher order byte first.)
9 - 10	2	Starting count	Tape counter value at which the header of this file starts. (Stored with higher order byte first.)
11 - 12	2	Ending count	Tape counter value at which an EOF block is placed. (Stored with higher order byte first.)

Column	Size	Item	Description
13 - 14	2	Number of opens	Number of open operations performed. (Stored with higher order byte first.)
15	1	User number	User number assigned by the CP/M system.
16 - 23	8	File name	Loaded with the name of the current or renamed file.
24 - 31	8	File type	Loaded with the file type of the current or renamed file. The highest order three bytes of this field are used by the system.

Note: See 14.1.5 for the stop and nonstop modes.

14.1.3 Tape File Control Block (T-FCB)

When using MTOS services, the application program must use a packet, called the T-FCB, which corresponds to the CP/M FCB. The T-FCB is placed in the same memory location as FCB but their formats are different.

T-FCB Format

00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15
dr	f1	f2	f3	f4	f5	f6	f7	f8	t1	t2	t3	ex		WA	rc

Disk drive code

File name

File type (high order 3 bytes)

A 1 in bit 7 identifies an SYS file.

A 1 in bit 7 indicates a read-only file.

Logical extent number assigned to this T-FCB (0-31).
The user must set this field to 00 when creating this T-FCB.

Total number of records in this logical extent. (0-128)

10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
t4	t5	t6	t7	t8	TA1	TA2	TBH	TBL	TRH	TRL	TCH	TCL	BCH	BCL	NCr

File type (lower order 5 bytes)

Tape attribute

Total number of blocks

Total number of records

Starting count

Ending count

Not used

20	21	22	23
32	33	34	35
cr	r0	r1	r2

Random record overflow field

Random record number (0-65535)

Record number (within current logical extent)

T-FCB field description

Offset	Field name	Description
* 0	dr	Loaded with the disk drive code. Must be set to 00 when drive H is selected or to 08 when another drive is selected.
* 1 - 8	f1 - f8	Loaded with the file name.
* 9 - 11	t1 - t3	File type (higher order 3 bytes). A 1 in t1, bit 7 identifies an SYS file. A 1 in t2, bit 7 indicates a read-only file.
* 12	ex	Logical extent number assigned to this T-FCB (0-31). The user must set this field to 0 when creating this T-FCB.
13 - 14	-	Used by the system.
15	rc	Total number of records in this logical extent (0-128)
16 - 20	t4 - t8	Loaded by the system with the lower five bytes of the file type.
21 - 22	TA1, TA2	Loaded by the system with a tape attribute.
23 - 24	TBH, TBL	Loaded by the system with the total number of blocks reserved for the file. This field is incremented by one each time a block is written onto the file.
25 - 26	TRH, TRL	Loaded by the system with the total number of records stored in the file. This field is incremented by one each time a record is written onto the file.
27 - 28	TCH, TCL	Loaded by the system with the starting tape count of the file.
29 - 30	BCH, BCL	Loaded by the system with the ending tape count of the file. This field is updated each time a record is written onto the file.
31	NCr	Not used.
* 32	Cr	Loaded with the record number in the current logical extent. The user must set this field to 0 when creating the T-FCB for the first time.
33 - 34	r0 - r1	Loaded with the random record number (0-65535).
35	r2	A nonzero value in this field indicates that a random record overflow condition has occurred.

Note: Fields identified by an asterisk must be filled by the user.

The contents of the T-FCB are updated whenever an access is made to the associated file on the MCT.

Reference: Format of the capacity of the H drive in the

STAT command

A>stat drive:filename

Incorrect file size values will be returned when a STAT command is executed for the H drive. This is because MTOS uses the file allocation vector field (FCB+16 through FCB+31) for purposes different from the original ones. In CPM, the file allocation vector field indicates the utilization status of the file space on the disk and the CP/M STAT command examines this field to calculate the size of the file (see the preceding two pages).

14.1.4 MTOS Programming Considerations

(1) Mount and remove

As mentioned in 14.1.2, MTOS loads the directory file
at the beginning of the MCT into the RAM directory area
in memory when controlling a tape file. Accordingly,
when a new MCT is inserted into the MAPLE microcassette
slot, it is necessary to execute a mount command. In
actual practice, however, the user need perform nothing
for this purpose because the automatic mount function
is executed when he accesses the H drive via BDOS (not
calling MIOS directly).

When a write or rename operation is performed on the
MCT, however, the associated file information is updated
only in the RAM directory. It is therefore necessary to
restore the updated file information from the RAM
directory onto the MCT at the end of the file
processing. This is called a remove operation. The
user must perform a remove operation immediately before
removing an MCT from the MAPLE whenever a file is
written on the MCT (execute a remove command from the
system screen or call BDOS 252).

(2) Automatic MTOS identification features

MTOS locates the position on MCT of a file by examining
the directory at the beginning of the MCT to make the
file accessible. Each file is blocked and accessed on
a block basis. Blocks are identified by sequentially
assigned numbers. Files are given a file name but
controlled by the system by a file number. The file
number counter is set to 0000 when the directory is
created for the MCT and incremented each time a file is
created. Therefore, the system can distinguish the
file from the other files. The maximum file number is
0FFFEH.

The following restrictions on file accessing must be
noted:

1)   No more than one file can be opened simultaneously.
2)   When a file is opened in the write mode, no other
     files can be opened until that file is closed.
3)   A new file may be opened when another file has been
     opened in the read mode. The currently open file,
     however, is closed.

MTOS accesses an MCT one block (256 bytes) at a time but
the interfaces of the operating system can access one

record (128 bytes) at a time. This means that MTOS
performs blocking/deblocking when constituting the
blocks.

The user must observe the following restrictions on
sequential file access:

In the write mode:

1)   Once a record is written in a block, no record can
      be placed in the next block until that block is
      filled.

2)   No data can be written in any locations other than
      the record areas in the current and next blocks.

In the read mode:

1)   No record area can be accessed other than those in
      the current and next blocks.

Read mode in the write mode:

1)   Any record in the current block may be read.

(3) Note on the creation of an MTOS file

A file created on the H drive (MCT) is placed after the last file on the MCT. Consequently, a disk full condition may occur even when there is only one file on the MCT if that file is stored as shown in the figure below.

Biginning of
new file
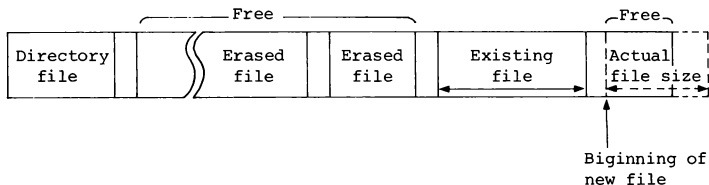
(1) MTOS and MIOS termination and power-off processing

MCT processing may be terminated by stopping the MCT
(CTRL STOP) or turning off MCT power. MTOS and MIOS
react as summarized below when one of the above
conditions occurs.

	MTOS	MIOS
Stop	If the current operation is a seek, write, or read, MTOS stops the operation, stops the motor, unload the read/ write head, and returns control to the calling program with the return code ØFFH in the A register.	MIOS terminates processing signaling a bad sector condition if the current operation is a read or write. The file is closed. MIOS does not restores the directory file when the current operation is a write.
Power off	If a power off condition occurs during a read or write, MTOS stops the motor and unloads the read/write head after completing the access to the current block.	Power is turned off after MIOS processing is completed. Subsequent MIOS processing when power is restored differs depending on in which mode power was turned off. When power was turned off in the continue mode, MCT processing continues at the point when power was turned off. In the other cases, the system is booted. Consequently, the file that was open when power was turned off is closed. The file that was opened in the write mode is not cataloged in the directory.

(2) Automatic MTOS identification features

1) Auto mount

MTOS carries out an automatic mount when an MTOS command
is executed to an MCT which has not been mounted (auto
mount feature). This allows the user to operate the MCT
without being aware of the need for mount processing.

2) Auto remove check

MTOS does not perform remove processing unconditionally;
when performing remove processing, MTOS reads in the
directory ID and compares the creation date and time
with those in the RAM directory. It carries out the
actual remove processing only when a match occurs.

(3) MTOS flags for the automatic MTOS identification
feature

MTOS uses a flag to control automatic MTOS operations
such as auto mount and remove. This flag is referenced
by the automatic MTOS identification feature to
identify the operation to be performed. The flag is
labelled TOSCTL. The value of TOSCTL is initialized
when an MCT is mounted. The value of TOSCTL are
determined as summarized below based on the system flag
2 in the directory ID.

TOSCTL is set to 50H when the system flag 2 contains a
0.
TOSCTL is set to the value of the system flag 1 when the
system flag 2 contains a 1.

TOSCTL   Overseas version = 0F078H
           Japanese-language version = 0ED84H

Bit	Name	Description
7	Reserved	Always set to 0.
6	TCCTCR	1: Enables counter adjustment.
		0: Disables counter adjustment.
5	TCSLAT	1: Determines access mode based on system status.
		0: Determines access mode based on directory data.
4	TCAMFG	1: Enables auto mount.
		0: Disables auto mount.
3	TCRCFG	1: Enables auto remove check.
		0: Disables auto remove check.
2	Reserved	
1	Reserved	
0	TCIGTS	1: Disconnects MTOS from CP/M.
		0: Includes MTOS into CP/M.

Normally, system flag 2 is set to 0 and TOSCTL defaults to 50H. The TOSCTL value may be altered using the following procedure:

1. Mount a work MCT and change system flag 1 field in the RAM directory to the desired value and system

flag 2 to 1.

System flag 1 on the RAM directory (IDFLG)

      Overseas version = 0F908H

      Japanese-language version = 0F88CH

System flag 2 on the RAM directory (TCNTTP)

      Overseas version = 0F909H

      Japanese-language version = 0F88DH

2.   Remove the MCT.

3.   Mount the target MCT.

    This causes the TOSCTL to be set to the value of
system flag 1.

(4) Tape access mode

MTOS and MIOS allow accesses to MCTs in two modes, i.e.,
stop and nonstop modes. These modes are selected by a
flag. Accesses are retried several times if the
requested block is not accessed successfully. The rest
of this subsection describes the stop and nonstop modes.

1) Stop and nonstop mode operations

i) Stop mode

-    Read

    In the stop mode, the motor is stopped each time one

block of data has been read. The motor is started
again when the next block is read. Each block may
be read any number (n) of times (n defaults to 2),
so that there can be n stops and starts in a single
block read. Since the motor is in the stopped
state after a block has been read, the next block
may be read at any given time.

- Write

  In the stop mode, the motor is stopped each time one
  block of data has been written. The motor is
  started again when the next block is written.
  Because the times required for turning the motor on
  and off are added to the write time, write
  operations in the stop mode take longer processing
  time than in the nonstop mode. Since the motor is
  in the stopped state after a block has been
  written, the next block may be written at any given
  time.

ii) Nonstop mode

- Read

  During a read in the nonstop mode, the motor is not
  stopped after one block of data has been read, so
  that no time is required for turning the motor on
  and off as in the stop mode. The subsequent read

operation must be started before the next block
passes through the read/write head of the MCT
drive; otherwise, a read error would occur.

- Write

    The motor is not stopped one block of data has been
    written in the nonstop mode. Accordingly, no time
    is required for turning the motor on and off as in
    the stop mode. If this mode is used for single-
    block write, the inter-block gaps will be
    increased. This results in a waste of tape space
    and an increase in the access time in the read
    mode.

2) Retrial of block read

MTOS writes a block of data onto an MCT n times (n
defaults to 2) to increase tape reliability (see Chapter
13 for the procedure for changing the default value).
Because of this redundancy, MTOS needs to read only one
block out of n blocks successfully in the read mode. If
errors occur in all (n) blocks written as a single
block, MTOS makes several times of retries.

(3) Controlling the tape access modes

The tape access mode is determined by the state of the
TAPEMOD. The meanings of the related flags are
described below, followed by the procedure for changing
the flag state.

i)

- TAPMOD   Overseas version = ØF2DDH

               Japanese-language version = ØFØ1AH

    Bit 7   Ø: Nonstop   1: Stop
    Bits 3-Ø: Number of retries

    The above bits are set when a file is created or
    opened.

    *:   When a file is created
          Set to the TACATR value.

    *:   When a file is opened
          Set to dirtam(i) if the TOSCTL TCSLAT bit is 1.
          Set to the TACATR value if the TOSCTL TCLSAT
          bit is Ø.

- TACATR   Overseas version = ØF78FH

               Japanese-language version = ØF7ØCH

    The bit assignments are identical to those of
    TAPMOD. TACATR is set to the value of DFTATR at a
    warm boot.

- DFTATR   Overseas version = 0F2E0H

                     Japanese-language version = 0F01DH

     DFTATR contains the default value of TACATR. It is
     set when a system reset occurs.

- TOSCTL   Overseas version = 0F708H

                     Japanese-language version = 0ED84H

     Bit 5 (TCSLAT): A 0 in this bit specifies that the
                  access mode is to be determined based on
                  the mode established when the file was
                  created.

                  A 1 in this bit specifies that the access
                  mode is to be determined based on the
                  value of the TACATR (current system
                  state).

     The above bit is set to 0 at a warm boot.

- dirtam(i)

     Corresponds to TAPMOD established when file i is
     created. dirtam(i) is equal to the value specified
     in the file attribute 1 field in the RAM directory.

ii) Changing the tape access mode

The value of TAPMOD is established when a new file is created or an existing file is opened for read. The user can change the value of TAPMOD using the following procedures:

a) Write

    1. Mount the MCT.

    2. Set TACATR (initialized to DFTATR value during warm boot) and DFTATR (initialized to default value on reset) to the desired values. (See previous pages).

    3. Create or open a file.

    4. Restores the flag values set up in step 2, as required.

    5. Remove the MCT.

b) Read

    1. Mount the MCT.

    2. Set TOSCTL bit 5 (TCSLAT) to 1. Set TACATR (initialized to DFTATR value during warm boot) and DFTATR (initialized to default value on reset) to the desired values. (See previous pages).

    3. Open and read a file.

    4. Restores the flag values set up in step 2, as required.

Note that if the values that are set up in step 2. above are not restored, the subsequent read/write operations will be affected accordingly.

The stop/nonstop tape access modes may also be specified from the system screen. The currently selected tape access mode determined by bit 7 of TACATR is displayed on the system screen.

MTOS/MIOS is included into CP/M through an interface.
MTOS calls are compatible with BDOS calls. Which of
MTOS or BDOS is specified is determined according to
the following rules:

1.   Functions not supported by MTOS are carried out by
     BDOS.

2.   Functions requiring an FCB (T-FCB) as a parameter are
     identified by the disk drive code at the beginning
     of the FCB (T-FCB).

3.   Other MTOS functions are serviced only when the MCT
     drive is logged in.

The application program must call BDOS when using MTOS
functions. MTOS calls differ from BDOS calls in that
MTOS calls using an FCB (T-FCB) must specify drive H as
the drive code at the beginning of the FCB. See 14.1.4
for programming notes on MCT processing.

Although FCB and T-FCB are located in the same memory
location as mentioned in 14.1.3, their format or use is
different. In the subsequent discussion, only T-FCB is
used to represent a file control block.

This section presents descriptions of the MTOS functions
with the entry and return information. The return
information include return codes which are returned when
CP/M error reporting is not suppressed (referenced to as
return code 1) and error information (obtained by
calling SETERROR; see 3.3.1) that is returned when CP/M
error reporting is suppressed. When the latter
information is returned by MIOS, the application program
can obtain the error details by referring to (BIOSERROR)
and (ISORCD). See the section about MIOS for the
location of these areas.

The MAPLE assigns drive H (physical device number is 8)
to MCTs, so the application program must set up the
drive code field of the T-FCB for a file as follows:

When the current drive is H: Ø
When the current drive is other than H: 8

In the description of the MTOS functions that follow,
descriptions of the use (and error information) of MTOS
functions which is identical to that of BDOS functions
(except the setting of the drive code at the beginning

of the T-FCB) are omitted.

TOSRCD address: Overseas version = 0F78CH

Japanese-language version = 0E709H

14.3.1 BDOS calls

The BDOS calls described in this subsection are as
follows:

	BDOS call number	Function
1.	No. 14	Select Disk Drive
2.	No. 15	Open File
3.	No. 16	Close File
4.	No. 17	Search for First File
5.	No. 19	Delete File
6.	No. 20	Read Sequential
7.	No. 21	Write Sequential
8.	No. 22	Create File
9.	No. 23	Rename File
10.	No. 30	Set File Attributes
11.	No. 33	Read Random
12.	No. 34	Write Random
13.	No. 251	Verify MCT
14.	No. 252	Close MCT (remove)
15.	No. 253	Open MCT (mount)
16.	No. 254	Read MCT ID number
17.	No. 255	Create Directory File

| Name | Select Disk Drive | BDOS No. 14 |

Function   Selects drive H:.

Entry   E reg = Drive No.

parameter   Load the E reg. with 7 when selecting

              drive H:.

Return   Return code 1

parameter   None.

              Return code 2

(TOSRCD)	A reg	H reg	Result
Ø	Ø	Ø	Normal termination
4	FFH	4	Mount unsuccessful
-	FFH	1	Error in MIOS

Explanation

This function selects the drive specified in the E reg.   If

no MCT is mounted when drive H is specified, the function

automatically performs the MCT mount function.

Name	Open File	BDOS No. 15

Function   Opens an MCT file.

Entry   DE reg = T-FCB address

parameter   Set the first byte of the T-FCB to drive H:
(or to 8 when the current drive is other than
H:).
Load the name of the file to be opened in
the T-FCB area.
Load each of the T-FCB+12, T-FCB+15, and
T-FCB+32 with Ø.

Return   Return code 1

parameter   A reg = $\begin{cases} \text{0: Normal termination} \\ \text{FFH: File not found} \end{cases}$

Return code 2

(TOSRCD)	A reg	H reg	Result
Ø	Ø	Ø	Normal termination
Ø	FFH	Ø	No file found
4	FFH	4	Mount unsuccessful
8	FFH	Ø	File not found
9	FFH	5	File is already open.

11H	FFH	5	File number associated with the specified file was not found in the MCT directory.
-	FFH	1	Error in MIOS

Explanation

- The Open File function searches the RAM directory for the specified file and, if it is found, reads the header block of that file into the T-FCB for subsequent read/write operations on the file. The function performs an automatic MCT mount function if no MCT is mounted on the MCT.

- An error is generated if the file is already open in the write mode. If the file is already open in the read mode, the function closes that file and opens a new file.

- Tape access mode (stop or non-stop) is determined by the file attribute in the directory file when the file is opened in the read mode. When opening a file for write, the user can overwrite the system-defined mode. See 14.1.5 for tape access modes.

Name	Close File	BDOS No. 16

Function   Closes an MCT file.

Entry   DE reg = T-FCB address

parameter   Set the first byte of the T-FCB to drive H:

(or to 8 when the current drive is other than

H:).

Load the name of the file to be closed in

the T-FCB area.

Then, call BDOS No. 16.

Return   Return code 1

parameter   A reg = $\begin{cases} 0: \text{Normal termination.} \\ \text{FFH: File not found (close error)} \end{cases}$

Return code 2

(TOSRCD)	A reg	H reg	Result
0	0	0	Normal termination
0	FFH	0	No file found
2	FFH	5	No MCT mounted
8	FFH	0	File not found
10H	FFH	0	Verify error
11H	FFH	5	The file number
			associated with the

found in the MCT directory.

	FFH	1	Error in MIOS

Explanation

The directory of a file which on which write operations are
performed during processing must be updated at the end of
the processing. This update is carried out by the Close
File function. When this function is called during write
file processing, an EOF block is written on the MCT and the
directory in memory is updated. (When the verify flag is on,
the function automatically verifies the validity of the file
contents before closing the file. The file will not be
cataloged into the directory file on tape if an error is
found.)

Once the RAM directory is updated, the directory file on the
MCT must also be updated with the Remove function before the
MCT is demounted from the MAPLE main unit.

Name	Search for First	BDOS No. 17

Function   Searches the file directory on MCT for the
            first time.

Entry   DE reg = T-FCB address
parameter   Set the first byte of the T-FCB to drive H:
            (or to 8 when the current drive is other than
            H:).
            Load the T-FCB area with the name of the file
            to be searched for.
            Then, call BDOS No.17.

Return   Return code 1
parameter
            A reg = $\begin{cases} \text{0: File found} \\ \text{FFH: File not found} \end{cases}$

            Upon return, the lowest 32 bytes of the
            DMA buffer is loaded with file directory
            information and the remaining bytes are
            padded with 0E5Hs.

            Return code 2

            o (TOSRCD)   A reg   H reg   Result

               0   0   0   File found

               0   FFH   0   No file found

               4   FFH   4   Mount unsuccessful

| | FFH | 1 | Error in MIOS |

Same as above for DAM.

Explanation

The Search for First function searches the RAM directory for the file specified in the T-FCB address in the DE registers. If the file is found, the function loads the starting 32 bytes of the DMA buffer with the file directory information and pads the remaining bytes with 0E5Hs.

This function loads the A register with 0FFH if the file is not found.

Name	Delete File	BDOS No. 19

Function   Deletes the specified T-FCB file from
the MCT directory.

Entry   DE reg = T-FCB address

parameter   Set the first byte of the T-FCB to drive H:
(or to 8 when the current drive is other than
H:).
Load the T-FCB area with the name of the file
to be searched for.
Then, call BDOS No.19.

Return   Return code 1

parameter   A reg = $\begin{cases} \text{0: Delete successful} \\ \text{FFH: File not found} \end{cases}$

Return code 2

(TOSRCD)	A reg	H reg	Result
0	0	0	Normal termination.
0	FFH	0	No file found
4	FFH	4	Mount unsuccessful
6	FFH	3	R/O file
FH	FFH	2	The file number associated with the specified file was not

found in the directory.

FFH   1   Error in MIOS

Explanation

The Delete File function deletes the file specified in the T-FCB address in the DE registers from the directory on drive H: (MCT).

If no MCT is mounted, this function performs the mount function automatically. A file is actually deleted when the corresponding presence flag in the RAM directory is reset. Therefore, the user must execute a Remove function before demounting an MCT from the MAPLE.

Name	Read Sequential	BDOS No. 20

Function   Reads an MCT file sequentially.

Entry   DE reg = T-FCB address

parameter   Set the first byte of the T-FCB to drive H:
            (or to 8 when the current drive is other than
            H:).
            Load the T-FCB area with the name of the file
            to be read sequentially.
            Then, call BDOS No.20.

Return   Return code 1

Parameter   A reg = $\begin{cases} 0: \text{Normal termination} \\ \text{Others: An EOF was encountered.} \end{cases}$

            DMA = 128-byte record

            Return code 2

            o (TOSRCD) A reg   H reg   Result

                0   0   0   Normal termination

                4   FFH   4   Mount unsuccessful

                9   FFH   5   An attempt was
                                        made to read a
                                        file other than
                                        the currently
                                        open files.

AH	1	0	File not opened
BH	FFH	5	A record number violating record access conventions was specified.
CH	1	0	EOF was reached.
-	FFH	1	Error in MIOS

o Same as above for DMA data.

Explanation

The Read Sequential function reads a record specified in the cr field (T-FCB+32) from the file specified in the T-FCB and returns that record to the DMA.

The A register is loaded with 0 on normal return. When an EOF is encountered, the register is loaded with a nonzero value.

The file to be read must be opened before this function is called. Although the function attempts the automatic MCT mount function if no MCT is mounted when it is executed, an error will be signaled by MTOS because the specified file is not open.

| Name | Write Sequential | BDOS No. 21 |

Function   Writes an MCT file sequentially.

Entry   DE = T-FCB address

parameter   Set the first byte of the T-FCB to drive H:

             (or to 8 when the current drive is other than

             H:).

             Load the T-FCB area with the name of the file

             to be written sequentially.

             DMA buffer = 128-byte record

Return   Return code 1

parameter   A reg = $\begin{cases} 0: \text{ Normal termination} \\ \text{Others: An error occurred.} \end{cases}$

             Return code 2

(TOSRCD)	A reg	H reg	Result
0	0	0	Normal termination
4	FFH	4	Mount unsuccessful
6	FFH	3	R/O file
9	FFH	5	An attempt was made to write on a file other than the current open file.

AH	1H	0	File not opened
BH	FFH	5	A record number
			violating record
			access conventions was
			specifi.d.
FH	FFH	2	R/O drive
-	FFH	1	Error in MIOS

Explanation

The Write Sequential function writes the record specified in
the cr field (T-FCB+32) to the file specified in the T-FCB.
Blocking/deblocking is performed during a sequential write
since the DMA buffer size is 128 bytes while the data block
size is 256 bytes/block. A record in the data block can be
read immediately.

The requested file must be opened and closed before and after
write processing is performed, respectively. A Remove
function must be executed before any MCT is to be removed
from the MAPLE's MCT drive unit.

Although this function performs the automatic MCT mount
function if no MCT is mounted when it is executed, an error
is signaled by MTOS because the specified file is not opened.

Note: How an EOT is detected during a sequential write

When an EOT is reached during a sequential write, control is passed from MIOS, a low level facility of the MTOS, to the MCT error processing routine. The routine reports CP/M of the error and forces the calling program to an abnormal termination.

A program can detect an EOT using the following procedure:
(1) Call SET ERROR (see 3.3.1).
(2) Perform a sequential write.
(3) A reg. ≠ Ø and (IOSRCD) = 2 (the motor is stopped) indicate that an EOT is reached.

    IOSRCD: Is a work area for storing an error return code from MIOS

Name	Create File	BDOS No. 22

Function   Makes a directory entry for an MCT file.

Entry   DE reg = T-FCB address

parameter   Set the first byte of the T-FCB to drive H:
                (or to 8 when the current drive is other than
                H:).
                Load the T-FCB area with the name of the file
                to be created.
                Then, execute BDOS No.22.

Return   Return code 1

parameter   A reg = $\begin{cases} \text{0: File creation successful} \\ \text{FFH: Directory full} \end{cases}$

                Return code 2

(TOSRCD)	A reg	H reg	Result
0	0	0	Normal termination
4	FFH	4	Mount unsuccessful
5	FFH	0	MCT directory full
7	FFH	5	The specified name is already assigned to an existing file.
9	FFH	5	Another file is already open.

| FH | FFH | 2 | R/O drive |
| - | FFH | 1 | Error in MIOS |

Explanation

The Create File function catalogs the file specified by the
T-FCB address in the DE registers into the MCT directory in
RAM.

If no MCT is mounted, this function performs the automatic
move function. Up to 12 files can be managed under the MTOS.
A directory full error occurs if an attempt is made to
create the 13th file.

Name   Rename File   BDOS No. 23

Function   Renames an MCT file.

Entry   DE reg = T-FCB address

parameter   Set the first byte of the T-FCB to drive H:

             (or to 8 when the current drive is other than

             H:).

             Load the T-FCB area with the old file name (in

             FCB Fields dr - t3) and a new file name (in

             FCB fields do - dl5).

Return   Return code 1

parameter   A reg = $\begin{cases} \text{0: Normal termination} \\ \text{FFH: File not found} \end{cases}$

             Return code 2

(TOSRCD)	A reg	H reg	Result
0	0	0	Normal termination.
0	FFH	0	File does not exist.
4	FFH	4	MCT need be mounted
6	FFH	3	R/O file
8	FFH	0	File not found
FH	FFH	2	R/O drive
-	FFH	1	Error in MIOS

Explanation

The Rename File function renames a file on drive H: (MCT).
The change is actually made in the corresponding entry of the
MCT directory in RAM. A Ø in the A register indicates normal
termination and an ØFFH indicates that the specified file was
not found.

A Remove function must be executed before an MCT is removed
from the MAPLE's MCT drive unit.

| Name | Set File Attributes | BDOS No. 30 |

Function   Sets MCT file attributes.

Entry   DE reg = T-FCB address

parameter   Set the first byte of the T-FCB to drive H:

(or to 8 when the current drive is other than

H:).

Load the T-FCB area with the requested file

name.

File attributes are specified as follows:

Extension {First byte   MSB=0: R/W file

MSB=1: R/O file

Second byte MSB=0: SYS file

MSB=1: DIR file

Return   Return code 1

parameter   A reg = {0: Normal termination

{FFH: File not found

Return code 2

(TOSRCD)	A reg	H reg	Result
0	0	0	Normal termination
0	FFH	0	No file found
4	FFH	4	Mount unsuccessful
8	FFH	0	File not found

	FFH	1	Error in MIOS
—			

Explanation

The Set File Attributes function changes attributes of an MCT
file cataloged in the RAM directory. Since the change is
actually made in the RAM directory, a Remove function must be
executed before the MCT is removed form the MAPLE.

Name	Read Random	BDOS No. 33

Function   Performs a random read of an MCT file.

Entry   DE reg = T-FCB address

parameter   Set the first byte of the T-FCB to drive H:
               (or to 8 when the current drive is other than
               H:).
               Load the T-FCB area with the target file name.

Return   Return code 1

parameter   A reg = 0: Normal termination
                        01: Reading a block and logical
                            extent not cataloged in
                            directory
                        02: Not returned in random mode

                        03: Cannot close current logical
                            extent
                        04: Seek to logical extent not
                            cataloged
                        05: Not returned in read mode
                        06: Seek past physical end of disk
                       DMA: Current 128-byte record

Return code 2

(TOSRCD)	A reg	H reg	Result
0	0	0	Normal termination
4	FFH	4	Mount unsuccessful
9	FFH	5	Another file already open
AH	1	0	File not open
BH	FFH	5	Illegal record access
CH	1	0	File not open
DH	4	0	Random access error
-	FFH	1	Error in MIOS

DMA: Current 128-byte record

Explanation

The Read Random function reads a record specified in the cr
field (T-FCB+32) from the file addressed by the T-FCB and
places the record in the DMA.
This function performs the same as the Read sequential
function.

Name	Write Random	BDOS No. 34

Function   Performs a random write to an MCT file.

Entry   DE reg = T-FCB address

parameter   Set the first byte of the T-FCB to drive H:
               (or to 8 when the current drive is other than
               H:).
               Load the T-FCB area with the target file name.
               DMA: 128 byte record

Return   Return code 1

parameter   A reg = 0: Normal termination

                      01: Writing a block and logical
                          extent not cataloged in
                          directory

                      02: Not returned in random mode

                      03: Cannot close current logical
                          extent

                      04: Seek to logical extent not
                          cataloged

                      05: Directory full

                      06: Seek past physical end of disk
                          (This error occurs when the
                          contents of the T-FCB + 35
                          are other than 0.)

Return code 2

(TOSRCD)	A reg	H reg	Result
0	0	0	Normal termination
4	FFH	4	Mount unsuccessful
6	FFH	3	R/O file
9	FFH	5	Another file already open
AH	1	0	File not open
BH	FFH	5	An illegal record number violating record access conventions
FH	FFH	2	R/O drive
-	FFH	1	Error in MIOS

Explanation

The Write Random function writes a record to the file addressed by the T-FCB with a record number specified in the cr field (T-FCB+32). This function performs the same as the Write sequential function except that it starts a write at a particular record.

Name	Verify MCT	BDOS No. 251

Function   Verifies an MCT file.

Entry   DE reg = T-FCB address

parameter   Set the first byte of the T-FCB to drive H:
            (or to 8 when the current drive is other than
            H:).
            Load the T-FCB area with the target file name.
            Then, call BDOS No.251.

Return   Return code 1

parameter   None.

            Return code 2

(TOSRCD)	A reg	H reg	Result
Ø	Ø	Ø	Normal termination
4	FFH	4	Mount unsuccessful
8	FFH	Ø	File not found
9	FFH	5	File already open
10H	FFH	Ø	Verify error
11H	FFH	5	File number corresponding to the specified file was not found in directory
-	FFH	1	Error in MIOS

Explanation

The Verify MCT File function reads a file specified in the T-
FCB address in the DE registers and checks the file on a
block basis to confirm that the values of the blocks match
the values calculated when the file was stored.

When called, this function performs an automatic MCT mount
function and opens it before verifying the specified file.

An error is signaled if the function is specified for a file
which is already open.

Name   Close MCT (remove) BDOS No. 252

Function   Writes RAM directory to MCT directory file.

Entry   None.

parameter   Call BDOS No.252.

Return   Return code 1

parameter   None.

                Return code 2

(TOSRCD)	A reg	H reg	Result
0	0	0	Normal termination
2	FFH	5	MCT not mounted
3	FFH	5	Cannot remove
12	FFH	5	Auto remove check erroɪ
—	FFH	1	Error in MIOS

                Note: See 14.1.5 for auto remove check.

Explanation

The Close MCT function writes the directory data from the RAM
directory onto the MCT directory file. Subsequently, no
directory function other than MMKDIR and MMOUNT can be
executed. No actual write takes place if no change has been
made to the RAM directory (i.e., neither write, delete, nor
rename operations have been performed).

Name   Open MCT (mount)   BDOS No. 253

Function   Reads the MCT directory file to the RAM
                directory.

Entry   None.
parameter   Call BDOS No.253.

Return   Return code 1
parameter   None.
                Return code 2

(TOSRCD)	A reg	H reg	Result
Ø	Ø	Ø	Normal termination
1	FFH	5	Not removed (mounted already)
4	FFH	4	Cannot mount
-	FFH	1	Error in MIOS

Explanation

The Open MCT function reads the MCT directory file and loads
file control information into the system RAM directory.
Subsequently, the calling program can execute MTOS functions.

Name   Read MCT ID Number   BDOS No. 254

Function   Reads the ID number of the current MCT.

Entry   None

parameter   Call BDOS No.254.

Return   Return code 1

parameter   A reg = ⎰0: Not mounted
                         ⎱Others: Mounted

                                    (Upon return, the DMA

                                    buffer is loaded with

                                    columns 0 - 53 of the

                                    directory file's first

                                    block.)

                  Return code 2

                  Same as above.

Explanation

The Read MCT ID Number function places the ID information

pertaining to the currently mounted tape (columns 0 - 53 of

the directory file's first block) into the DMA buffer. The

function returns nothing when no MCT is mounted.

Name	Create Directory File	BDOS No. 255

Function   Creates a directory file on MCT.

Entry   DE reg = T-FCB address

parameter   T-FCB contents

             0 - 7: Tape name (ASCII code)

             8 - 9: Volume number (ASCII code)

             Call BDOS No.255.

Return   Return code 1

parameter   None

             Return code 2

(TOSRCD)	A reg	H reg	Result
0	0	0	Normal termination
1	FFH	5	The function was called with the MCT mounted
-	FFH	1	Error in MIOS

Explanation

If executed when no tape is mounted, the Create Directory
File function performs an automatic MCT mount function and
rewinds the tape to the beginning, then creates a directory
file.

The first 10 bytes of the directory file are copied from the
the first 10 bytes of the T-FCB and the number of directory
entries is set to 0.

A RAM directory is also created by this function.

Each MTOS or MIOS function returns several return codes. An
MTOS function returns control to the calling program with
error information loaded in the A and H registers and TOSRCD.
The values placed in the A and H registers depends on the
error code in the TOSRCD. Furthermore, an MTOS function
reports four types of CP/M error conditions, or returns
control to the caller with the corresponding error codes
loaded in A and H registers. (As described in 3.3.1, the
MTOS error report mode can be changed using the Set Error and
Reset Error functions.)

a) BDOS error reports
The MTOS reports four CP/M error conditions with one of
the following messages:
When control is returned from the MIOS on error:

    BAD SECTOR

When an error occurred during a mount:

    BAD SELECT

When an illegal access was made to an R/O drive:

    R/O

When an illegal access was made to an R/O file:

    R/O FILE

If error reporting is suppressed, MTOS returns control to the
calling program with corresponding error codes in the
A and H registers. Error codes assigned to the four error
conditions are listed below.

	A	H
BAD SECTOR	FF	1
R/O	FF	2
R/O FILE	FF	3
BAD SELECT	FF	4

b) TOSRCD return codes

There are 18 TOSRCD return codes in total. Table 1
lists the TOSRCD return codes and the meanings of the
corresponding return codes that are placed in the A and H
registers.

Return code 5 in the H register indicates that the
indicated error is unique to MTOS. When the H register
contains a 0, the indicated error corresponds to a CP/M
error. If CP/M error reporting is suppressed, the value in
the H register may be changed to the value shown in (a).

Table 2 lists the error codes reported by the MTOS functions.

TOSRCD address: Overseas version   ØF78CH

Japanese-language version   ØF7Ø9H

Table 1   MTOS Error Codes

TOSRCD return value	A register	H register	Meaning
0	0	0	Normal termination
1	FF	5	Not removed
2	FF	5	Not mounted
3	FF	5	Cannot remove
* 4	FF	5 (4)	Cannot mount
5	FF	0	Directory full
* 6	FF	0 (3)	R/O file
7	FF	5	File already exists
8	FF	0	File not found
9	FF	5	File already open
A	1	0	File not open
B	FF	5	Illegal record access
C	1	0	EOF reached
D	4	0	Random access error
* F	FF	0 (2)	R/O drive
10	FF	0	Verify error
11	FF	5	File number corresponding to the specified file not found in the directory
12	FF	5	Auto remove check error

*   Numbers enclosed in parentheses identify the values returned
    when CP/M error reporting is suppressed.

Note: Auto remove check is a function performed before a file is
        removed.   This function reads the directory ID assigned to
        that file, compares the date and time at which the file was
        created with the information in the RAM directory, and removes
        the file if they match.   Auto remove check can be specified by
        setting a particular bit.

Table 2 Errors Reported by MTOS Functions

MTOS call No.	Function name	Return code / Parameter	1	2	3	4	5	6	7	8	9	A	B	C	D		F	10	11	12
12	Get version No.			·																
13	Reset disk system																			
14	Select disk drive					O														
15	Open file	DE ← FCBad				O				O	O								O	
16	Close file	DE ← FCBad		O						O								O *1	O *1	
17	First search	DE ← FCBad				O														
18	Next search																			
19	Delete file	DE ← FCBad				O		O									O			
20	Sequential read	DE ← FCBad				O					O	O	O	O						
21	Sequential write	DE ← FCBad				O		O			O	O	O				O			
22	Create file	DE ← FCBad				O	O		O		O						O			
23	Rename file	DE ← FCBad				O		O		O							O			
24	Get login vector																			

MTOS call No.	Function name	Return code / Parameter	1	2	3	4	5	6	7	8	9	A	B	C	D	╳	F	10	11	12
25	Get login disk No.																			
26	Set DMA address																			
27	Get allocation address																			
28	Set write protect																			
29	Get R/O vector																			
30	Set file attributes					○				○										
31	Disk parameter block address																			
32	Get user code																			
33	Random read	DE ← FCB				○					○	○	○	○	○					
34	Random write	DE ← FCB				○		○			○	○	○				○			
35	Compute file size	DE ← FCB																		
36	Set random record	DE ← FCB																		

MTOS call No.	Function name	Return code / Parameter	1	2	3	4	5	6	7	8	9	A	B	C	D	✕	F	10	11	12
37	Reset disk drive	DE → Drive vector																		
38	Un-defined																			
39	Un-defined																			
40	Random write with zero fill																			

MTOS call No.	Function name	Return code / Parameter	1	2	3	4	5	6	7	8	9	A	B	C	D	✕	F	10	11	12
251	Verify	DE ← TFCB				○				○	○							○	○	
252	Remove			○	○															○
253	Mount		○			○														
254	Read tape ID																			
255	Create tape directory	DE ← FCB																		

Notes 1: MTOS functions identified by an o marks are not
supported by the MTOS.

2: For details about the functions that return
parameters in registers, see the explanations of
those functions in 14.3.1.

3: *1 denotes that the code is returned when verify
if specified.

MIOS is located between the MTOS and MCT firmware and
consists of 19 functions. One MIOS function is equivalent to
a sequence of n BIOS calls. To use an MIOS function, call
the MCMTX after loading the B register with its entry number
and some other registers and work area with parameter data,
if necessary.

MCMTX Entry address: WBOOT + 78H

B register	Function
0	MIRDST
1	MIRDCT
2	MISTCT
3	MISTOP
4	MIPLAY
5	MIREC
6	MIFF
7	MISREW
8	MIREW
9	MIFFTE
10	MIRWTT
11	MIHDON
12	MIHDOF

13	MISKTP
14	MISTMP
15	MIRSMP
16	MIEDBL
17	MIWTBL
20	MIGTWP

14.5 MIOS Functions

This section lists describes the entry and return
parameters and error codes for each MIOS function. A
table of error codes is located at the end of this
section.

* Return information work area address

    BIOSERROR   Overseas version: ØF536H

                Japanese-language version: ØF2B3H

    IOSRCD   Overseas version: ØF78DH

                Japanese-language version: ØF70AH

*: Programming notes on the use of MIOS functions
Reserve the buffer area for the MIOS (pointed to by the
TOSDMA) in a location between 8000H and ØFFFFH.
The number of data bytes specified for reading or
writing MCT blocks must be in the range 4 to 270.

Function   Reads MCT Status.

Entry   None

parameter   Load the B reg. with 0 and call MCMTX.

Return   Hreg. = Microcassette status

parameter   Bit 7: Head position (0: OFF, 1: ON)

                 Bit 6: Motor   (0: Stop, 1: Move)

                 Bit 5: Wind   (0: No, 1: Wind)

                 Bit 4: Fast Feed   (0: No, 1: Fast Feed)

                 Bit 3: Play   (0: No, 1: Play)

                 Bit 2: Record   (0: No, 1: Record)

                 Bit 1: Not used

                 Bit 0: Reserved (used by another device)

              No error information is returned.

Explanation

The MIRDST function reads the microcassette status.

Details of the status is as shown above.

Name	MIRDCT	Entry No. MIOS 1

Function   Reads the MCT counter.

Entry   None

parameter   Load the B reg. with 1 and call MCMTX.

Return   Hreg. = Counter value

parameter   No error information is returned.

Explanation

The MIRDCT function reads the value of the 16-bit MCT counter. The counter increments when the tape is moved forward and decrements when it is moved backward.

Name   MISTCT   Entry No.   MIOS 2

Function   Sets the MCT counter.

Entry   DE reg = Counter value

parameter   Load the B reg. with 2 and call MCMTX.

Return   None

parameter

Explanation

The MISTCT function sets the MCT counter to the

specified value.

Name   MISTOP   Entry No.   MIOS 3

Function   Stops MCT.

Entry   None

parameter   Load the B reg. with 3 and call MCMTX.

Return   None

parameter

Explanation

The MISTOP function stops the microcassette drive motor.

Name   MIPLAY   Entry No. MIOS 4

Function   Turns on the MCT motor in the play mode.

Entry   None

parameter   Load the B reg. with 4 and call MCMTX.

Return   None

parameter

Explanation

The MIPLAY function rotates the MCT motor in the play

mode.   The read data signal is sent if the read/write

head is on.

Name	MIREC	Entry No. MIOS 5

Function   Turns on the MCT motor in the record mode.

Entry   None

parameter   Load the B reg. with 5 and call MCMTX.

Return   None

parameter

Explanation

The MIREC function rotates the MCT motor in the record
mode. Tape data is erased if the read/write head is on.

Name	MIFF	Entry No. MIOS 6

Function   Starts the MCT motor in the fast foward
                mode.

Entry   None

parameter   Load the B reg. with 6 and call MCMTX.

Return   None

parameter

Explanation

The MIFF function rotates the MCT motor in the fast foward
mode.

Name	MIREW	Entry No. MIOS 7

Function   Moves the MCT motor at a slow speed in
                the backward direction.

Entry   None
parameter   Load the B reg. with 7 and call MCMTX.

Return   A reg. (BIOSERROR)
parameter   0: Normal termination
                 0FEH: An unrecognizable error occurred
                C reg. (IOSRCD)
                    0: Normal termination
                    1: An error occurred while the motor was
                       moving (The head is automatically
                       turned off in the rewind mode.)

Explanation

The MIREW function rotates the MCT motor in the slow
rewind mode. The head is automatically turned off in
this mode.

Function   Moves the MCT motor in the reverse
              direction.

Entry   None

parameter   Load the B reg. with 8 and call MCMTX.

Return   A reg. (BIOSERROR)

parameter   0: Normal termination

              0FEH: An unrecognizable error occurred

              C reg. (IOSRCD)

                 0: Normal termination

                 1: An error occurred while the head is
                    moved (The head is automatically
                    turned off in the rewind mode.)

Explanation

The MIREW function rotates the MCT motor in the rewind
mode. The head is automatically turned off in this mode.

Name	MIFFTE	Entry No. MIOS 9

Function   Moves MCT rapidly in the forward
                direction to the end.

Entry   None
parameter   Load the B reg. with 9 and call MCMTX.

Return   A reg.   (BIOSERROR)
parameter   0: Normal termination
                    4: Time out error
                    5: Seek error
                    6: BREAK key pressed
                    7: Power off error
                C reg.   (IOSRCD)
                    0: Normal termination
                    1: Head move error
                    7: Tape not found

Explanation

The MIFFTE function rotates the MCT motor until the MCT
end is reached.

Function   Rewinds MCT to the start.

Entry   None

parameter   Load the B reg. with 10 and call MCMTX.

Return   A reg.   (BIOSERROR)

parameter   0: Normal termination

                   4: Time out error

                   5: Seek error

                   6: BREAK key pressed

                   7: Power off error

              C reg.   (IOSRCD)

                   0: Normal termination

                   1: Head move error

                   7: Tape not found

Explanation

The MIRWTT function moves MCT backward up to the start of the tape.

Name	MIHDON	Entry No. MIOS 11

Function   Turns on the MCT head.

Entry   None

parameter   Load the B reg. with 11 and call MCMTX.

Return   A reg.   (BIOSERROR)

parameter   0: Normal termination

                OFEH: An unrecognizable error occurred.

               C reg.   (IOSRCD)

                   0: Normal termination

                   1: An error occurred while the head is

                       moved

Explanation

The MIHDON function turns on the MCT head (i.e., loads
the head).

Function   Turns off the MCT head.

Entry   None

parameter   Load the B reg. with 12 and call MCMTX.

Return   A reg.   (BIOSERROR)

parameter   0: Normal termination

                OFEH: An unrecognizable error occurred.

              C reg.   (IOSRCD)

                   0: Normal termination

                   1: Head move error

Explanation

The MIHDOF function turns off the MCT head off (i.e.,
unloads the head.).

Name   MISKTP   Entry No. MIOS 13

Function   Seeks for the specified MCT position.

Entry   DE reg = Desired count
parameter   Load the B reg. with 13 and call MCMTX.

Return   A reg. (BIOSERROR)
parameter   0: Normal termination
                  5: Seek error
                  6: BREAK key pressed
                  7: Power off error
              C reg. (IOSRCD)
                  0: Normal termination
                  1: Head move error
                  3: Write protect error
                  7: Tape not found

Explanation

The MISKTP function moves tape to the count specified by
the DE register pair as follows:

    Current count < desired count: Moves tape in the
                                   forward direction.
    Current count = desired count: Does nothing.
    Current count > desired count: Moves tape in the
                                   reverse direction.

Name   MISTMP   Entry No. MIOS 14

Function   Sets the MCT move protect count.

Entry   DE reg = Move protect count
parameter   Load the B reg. with 14 and call MCMTX.

Return   None
parameter

Explanation

The MISTMP function sets the MCT move protect count (
total count to which MCT can run). After a move protect
count is set, the tape automatically stops when an
attempt to seek past this count is made.
Example:

            Current count   500

            Move protect count   300

Assume that the tape has been rewound to count 400 and
an attempt is made to seek the tape up to count 700.
Because the move protect count is decremented by 100
when the tape is rewound to 400, the tape can now feed
only 200 counts and, therefore, stops at count 600.

Name   MIRSMP   Entry No. MIOS 15

Function   Resets the MCT protect count.

Entry   None

parameter   Load the B reg. with 15 and call MCMTX.

Return   None

parameter

Explanation

The MIRSMP function resets the MCT move protect count

(total count to which MCT can run).

Function   Reads one block of data from an MCT file.

Entry   A = Block number position offset

parameter   C = Read mode

              DE = Bytes of data to be read

              HL = Block number

              TOSDMA = Pointer to the buffer for storing

                  read data (specify a value in the

                  range 8000H to 0FFFFH).

              Specify a block ID character or ? as the

              first buffer data.

              Load the B reg. with 16 and call MCMTX.

Return   TOSDMA = Read data

parameter   A reg. (BIOSERROR)

                  0: Normal termination

                  1: Read error

                  6: BREAK key pressed

              C reg. (IOSRCD)

                  0: Normal termination

                  1: Head move error

                  4: Data error (Data bits 0 and 1

4: Data error (Data bits 0 and 1

   unrecognizable)

5: CRC error

6: Invalid block mode

Explanation

See the page after the next.

Function   Writes one block of data to an MCT file.

Entry   DE reg = Bytes of data to be written

parameter   C = Write mode

              TOSDMA = Pointer to the buffer storing

                 write data (specify a value in the

                 range 8000H to 0FFFFH).

              Load the B reg. with 17 and call MCMTX.

Return   A reg. (BIOSERROR)

parameter   0: Normal termination

                 2: Write error

                 3: Write protect error

                 6: BREAK key pressed

              C reg. (IOSRCD)

                 0: Normal termination

                 1: Head move error

                 2: Motor stopped

                 3: Write protect error

Explanation

See the next page.

1. Parameter description

 a. Specify either read or write mode by setting the C
register as follows:

    Bit

    7   Ø: Non-stop mode, 1: Stop mode

    6

    5

    4

    3   ⎱ Number of retries (Ø -15)

    2   ⎱

    1   ⎱

    Ø   ⎱

  b. Load the DE register pair with the number of bytes
     to be read or written. The number must be in the
     range from 4 to 27Ø (the block identification field
     requires four bytes at least).

  c. Load TOSDMA with a buffer address in the range from
     8ØØØH to ØFFFFH.

d. Read/write data format

Buffer top (pointed to by TOSDMA)
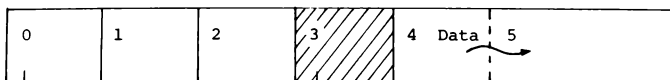
A field identifying the nth
write for the current block.
This area is used by the
system.

Block ID field:
Filled by the user with arbitrary data during write.
During read, this field is compared with the
user-specified block ID to identify the requested
block.

2. MIOS 16 (Read One Block)

Explanation:

This function reads the number of bytes specified in the
DE register from a block in the mode specified in the
C register into the buffer addressed by TOSDMA.   The
block must satisfy the following conditions:

(Condition 1)

     The read data ID field value (specified at the
     beginning of the buffer) matches the ID code loaded
     at the beginning of the buffer during execution of
     this function.   The ID code "?" matches any ID
     code.

(Condition 2)

     The block number of the read data placed in the two
     bytes starting at the address identified by (buffer
     address + value in A) matches the block number in
     the HL register.

An error is generated if the following condition occurs:
     (Block number specified in HL) - 2 <= (Block number
     of read block) < (Block number specified in HL)

As stated in 14.1.5, MTOS writes a block of data several times to increase data reliability. If read errors occur in all duplicated blocks while verifying the write, it attempts to retry the read.

3. MIOS 17 (Write One Block)

Explanation:

This function writes on MCT the number of bytes
specified in DE from the buffer addressed by TOSDMA in
the mode specified in C.

Chapter 15   I/O and Peripheral Devices

This chapter discusses the following tops:

1. I/O address space

2. Physical file structure

3. EPSP protocol

4. DIP switches

15.1   I/O Address Space

The MAPLE I/O address space listed below.

I/O address	Read	Write
00H	ICRL (Input Capture Register Low)	CTLR1 (Control Register 1)
01H	ICRH (Input Capture Register High)	CMDR (Command Register)
02H	ICRL.B (ICRL Bar code Trigger)	CTLR2 (Control Register 2)
03H	ICRH.B (ICRH Bar code Trigger)	
04H	ISR (Interrupt Status Register)	IER (Interrupt Enable Register)
05H	STR (Status Register)	
06H	SIOR (Serial I/O register)	SIOR (Serial I/O register)
0CH	8251 Data Read	8251 Data Write
0DH	8251 Status Read	8251 Command Write
0EH	SED 1320 PSR	SED 1320 PDIR
0FH	SED 1320 PDOR	SED 1320 PDIR

I/O addresses between 00H and 7FH excluding the above addresses are not used.

I/O addresses 80H through 0FFH are used to access optional units over the system bus. Since addresses 80H through 0DFH are assigned to EPSON optional units addresses 0E0H through 0FFH must be used for user-supplied options.

Currently used I/O addresses

I/O address	Optional unit	I/O address	Optional unit
8ØH	Intelligent RAM disk	9ØH	Nonintelligent RAM disk
81H		91H	
82H		92H	
83H		93H	
84H	Direct modem	94H	
85H		95H	
86H		96H	
87H		97H	
88H	Japanese-language processor	98H	
89H		99H	
8AH		9AH	
8BH		9BH	
8CH		9CH	
8DH		9DH	
8EH		9EH	
8FH		9FH	

I/O address	Optional unit
AØH	Synchronous communication unit
A1H	
A2H	
A3H	
A4H	
A5H	
A6H	
A7H	

See Chapter 16, "Extension Units" for use of I/O addresses.

(1) I/O address 00H

[Read] ICRL

The CPU reads the lower 8 bits from the current FRC (16-bit counter running at 614.4 KHz clock) through this I/O port address. Since the contents of the FRC are loaded into port addresses 00H and 01H immediately once this port is read, the higher 8 bits from the FRC can also read from address 01H immediately.

Addresses 00H and 01H must be read in that order.

[Write] CTLR1

CTLR1 bits are assigned as follows:

Bit	Name	Function
7	BRG3	Sets the clock rate for the 8251 (see section 12.2).
6	BRG2	
5	BRG1	
4	BRG0	
3	SWBCD	Indicates the state of the bar code bar code connector power switch (5V). 1: ON, 0: OFF
2	BCD1	Sets the bar code reader interrupt trigger (see section 10.7)
1	BCD0	
0	BANK	Specifies the memory bank. 0: BANK0 0000H - 07FFFH = ROM 8000H - 0FFFFH = RAM 1: BANK1 0000H - 0FFFFH = RAM

Any data to be written into this I/O address must also be saved into work area labeled CTLR1.

```
LD   A, (CTLR1)
```

```
Set necessary bits of A reg. to 1.
```

```
LD   (CTLR1), A
OUT  (00H), A
```

```
CTLR1 --- Overseas version = 0F0B0H
              Japanese-language version =0ED90H
```

(2) I/O address 01H

[Read] ICRH

The CPU reads the higher 8 bits from the current FRC
through this port address. The contents of the FRC is
latched immediately when address 00H is read.
Consequently, the contents of 00H and 01H must be read
in that order.

[Write] CMDR

CMDR bit assignments are as follows:

Bit	Name	Function
7 6 5 4 3		Unused Always set to 0.
2	RESOVF	1: Resets OVF interrupt INTR signal generated by FRC overflow. 0: Does nothing. The interrupt INTR signal must be reset by the OVF interrupt processing routine before OVF interrupts are to be enabled.

1	RESRDYSIO	1: Resets RDYSIO signal used for communicating with the 7508 (the signal indicates whether the 7508 is ready). 0: Does nothing. See Chapter 11, "7508 CPU" for the use of this bit.
0	SETRDYSIO	1: Sets RDYSIO signal used for communicating with the 7508. 0: Does nothing. This bit is not used by applications.

Set only the necessary bit (bit 1 or 2) to 1 before sending data to this I/O address.

(3) I/O address 02H

[Read] ICRL.B

This address contains the lower 8 bits from the FRC
latched on a transition in the state of the signal from
the bar code reader (positive or negative trigger). A
transition in the signal state can be recognized through
the ICF interrupt processing routine or by checking I/O
address 04H, bit 3 (INT3 signal).

The higher 8 bits can be read from I/O address 03H.

I/O addresses 02H and 03H must be read in that order.

[Write] CTLR2

CTLR2 bit assignments are as follows:

Bit	Name	Function
7 6		Unused
5	AUX	1: Specifies that the 8251 is to be connected to the RS-232C connector. 0: Specifies that system bus lines TxDE and *RxDE are to be used to control 8251 handshaking. This bit is set to 0 immediately after the RESET switch is pressed.

Bit	Name	Function
4	INHRS	Used to prevent generation of garbage data when power to the RS-232C drivers is turned on or off. Set this bit to 1 when turning on or off the driver power.
3	SWRS	1: Indicates that RS-232C power (+8 V) is on. 0: Indicates that RS-232C power (+8 V) if off.
2	LED2	1: Indicates that keyboard LED2 is set to on. 0: Indicates that keyboard LED2 is set to off.
1	LED1	1: Indicates that keyboard LED1 is set on. 0: Indicates that keyboard LED1 is set to off.
0	LED0	1: Indicates that keyboard LED0 is set to on. 0: Indicates that keyboard LED0 is set to off.

Write data into this I/O address using the following procedure:

```
    LD    A, (CTLR2)
```

```
    Set necessary bits of A reg. to 1.
```

```
    LD    (CTLR2), A
    OUT   (02H), A

    CTLR2 --- Overseas version = 0F0B2H
              Japanese-language version =0ED92H
```

(4) I/O address Ø3H

[Read] ICRH.B

This address contains the higher 8 bits from the FRC
latched by a transition in the state of the signal from
the bar code reader (positive or negative trigger).
Transition in the signal state can be recognized through
the ICF interrupt processing routine or by checking I/O
address Ø4H, bit 3 (INT3 signal).
The INT3 signal (interrupt signal from the bar code
reader) is reset when this I/O address is read.
Addresses Ø2H and Ø3H must be read in that order.

[Write]

None.

(5) I/O address 04H

[Read] ISR

The bits in I/O address 04H indicate the associated

interrupt status as shown below:

Bit	Name	Function
7		Unused
6		
5	INT5 (EXT)	Indicates the (EXT) external interrupt (system bus external interrupt) status.
4	INT4 (OVF)	Indicates the status of the OVF interrupt caused by FRC overflow. This bit is reset by setting I/O address 01H, bit2.
3	INT3 (ICF)	Indicates the bar code reader interrupt status. This bit is reset by when I/O address 03H is read.
2	INT2 (CD)	Complement of RS-232C CD signal. (When CD is set low, INT2 is set high, generating a CD interrupt.)

1	INT1 (8251)	Indicates the status of the 8251 interrupt generated when RxRDY is set. This bit is reset when receive data is read from the 8251.
0	INT0 (7508)	Indicates the 7508 interrupt status. This bit is reset when the 7508 status is read.

Each of the above statuses can be read if the corresponding interrupt is masked.

[Write] IER

The IER bits enable or disable the corresponding interrupt. All interrupts are disabled when the RESET switch is pressed.

Bit	Name	Function	
7		Unused	
6			
5	IER5	EXT interrupts	
4	IER4	OVF interrupts	
3	IER3	ICF interrupts	1: Enabled
2	IER2	CD interrupts	0: Disabled
1	IER1	8251 interrupts	
0	IER0	7508 interrupts	

Write data into this I/O address using the following
procedure:

```
LD    A, (IER)
```

```
Set necessary bits of A reg. to 1.
```

```
LD   (IER), A
OUT  (04H), A

IER --- Overseas version = 0F0B3H
         Japanese-language version =0ED93H
```

(6) I/O address 05H

[Read] STR

The bits in I/O address 05H indicate the I/O status as
follows:

Bit	Name	Function
7		Unused
6		
5		
4		
3	RDYSIO	Indicates the state of the control signal for the serial bus that serves as an interface to the 7508. 1: 7508 accessible 0: 7508 inaccessible See Chapter 11, "7508 CPU" for to to access the 7508.
2	RDY	Indicates the state of the RDY input line from the 7508. This line is not used.
1	BRDT	Indicates the state of the data input signal from the bar code reader.

Bit	Name	Function
0	BANK	Indicates the current BANK status. 0: BANK0 0000H – 7FFFH = ROM 8000H – 0FFFFH = RAM 1: BANK1 0000H – 0FFFFH = RAM

(7) I/O address 06H

[Read]

The Z80 CPU reads this I/O address when receiving data from the 7508.

[Write]

The Z80 CPU reads this I/O address when sending a command or data to the 7508.

See Chapter 11, "7508 CPU" for how to access the 7508.

(8) I/O address 0CH

[Read]

The Z80 CPU reads this I/O address when receiving RS-232C receive data from the 8251.

[Write]

The Z80 CPU reads this I/O address when sending RS-232C send data to the 7508.

(9) I/O address 0DH

[Read]

The Z80 CPU reads this I/O address when reading the 8251 status.

[Write]

The Z8Ø CPU reads this I/O address when sending a
command to the 8251.

See Chapter 12, "Using 8251" or consult technical
reference manuals on 8251 for I/O addresses ØCH and ØDH.

(10) I/O address 0EH

[Read]

The Z80 CPU reads this I/O address when reading the

6301 status.

[Write]

The Z80 CPU reads this I/O address when sending data to

the 6301.

(11) I/O address 0FH

[Read]

The Z80 CPU reads this I/O address when receiving data

from the 6301.

[Write]

The Z80 CPU reads this I/O address when sending a

command to the 6301.

The user cannot read I/O addresses 0EH and 0FH directly.

Use the slave BIOS call (WBOOT + 72H) to access the

6301.

This section describes the structure of the MAPLE files
stored on the MAPLE drives. The MAPLE drives use
various types of storage media.   The storage drives and
media are summarized below.

1. Drive A: Internal RAM disk

   See Chapter 16, "Extension Units" for the

   extension unit RAM disk which is also assigned

   to drive A:.

2. Drive B: and C: ROM capsule

3. Drive D:, E:, F: and G: Floppy disk

4. Drive I: Extension unit ROM capsule

See Chapter 14, "MTOS and MIOS" for MCT files in drive
H:.

(1) Internal RAM disk

The internal RAM disk format in main memory is shown below.

0000H

File area

(9 sectors)

Check sum area

(512 bytes, sectors 5-8)

Directory area

(512 bytes, sectors 1-4)

0FFFFH

(SIZRAM) contains the RAM disk size in 1K byte units. (Overseas version = 0F009H, Japanese-language version = 0ED09H)

(TOPRAM) - 1
↓
Contains the RAM disk top address (Overseas version = 0F076H, Japanese-language version = 0ED82H)

The internal RAM disk area is divided into sectors from address (TOPRAM)-1 to higher address. The sectors are numbered in ascending order as shown in the figure at left.

The structure of the directory is the same as that of the standard CP/M. Up to 16 entries can be accomodated in the directory.

The check sum area stores 1-byte check sums calculated with data in each sector.

The 9th and following sectors are used as the file area. The size of the file area is determined according to the size of the RAM disk area.

Files are managed in 1K-byte units using the allocation map in the directory.

The actual size of the file area is (SIZERAM)-1K bytes (directory area + check sum area).

(2) ROM capsule

(2-1) Types of ROM

2764, 27128, and 27256 can be used as MAPLE ROM devices.

(2-2) Addresses

Addresses as viewed from the OS (logical addresses) have a one-to-one correspondence with actual ROM addresses (physical addresses) on 2764 and 27128. On 27256, the relationship between the logical and physical addresses is reversed at address 4000H. This is because the meaning of pin A14 is different for 2564, 27128 and 27256. On 2764 and 27128, this pin must always be set high and, therefore, the signal at pin A14 is inverted by hardware. On 27256, however, pin A14 is used for addressing. This means that address 0 is mapped into address 4000H because accessing address 0 sets A14 pin high.

Example:

○ When accessing address ∅∅∅∅:

```
                        A14 A13 ～～～ A∅
Logical address          ∅   ∅  ～～～  ∅      (Address ∅∅∅∅)
                         │   │        │
              Inverted   │   │        │
                         ▼   ▼        ▼
Physical address         1   ∅  ～～～  ∅      (Address 4∅∅∅)
```

○ When accessing address 5∅∅∅H

```
                        A14 A13 A12 A11 ～～～ A∅
Logical address          1   ∅   1   ∅  ～～～  ∅
                         │   │   │   │        │
              Inverted   │   │   │   │        │
                         ▼   ▼   ▼   ▼        ▼
Physical address         ∅   ∅   1   ∅  ～～～  ∅
```
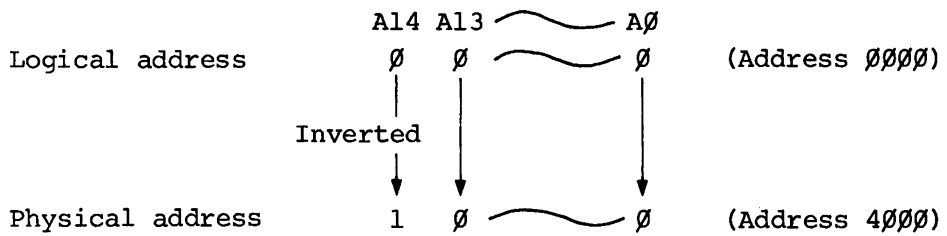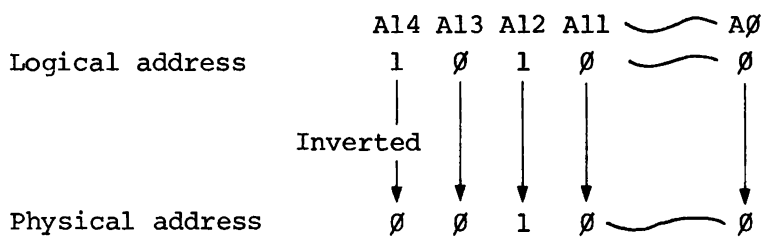
Physical address viewed from the OS	Actual ROM address		
	2764	27128	27256
∅ ∅ ∅ ∅ H	∅ ∅ ∅ ∅ H	∅ ∅ ∅ ∅ H	4 ∅ ∅ ∅ H
1 F F F H	1 F F F H		
2 ∅ ∅ ∅ H			
3 F F F H		3 F F F H	7 F F F H
4 ∅ ∅ ∅ H			∅ ∅ ∅ ∅ H
7 F F F H			3 F F F H

15-23

(2-3) ROM capsule memory map

The addresses referred to in the following description
are all logical ones. Care must be taken when using
27256 ROM devices.

(For example, address 1000H corresponds to address 5000H
in 27256 ROM.)

i) General

ROM is divided into directory and file areas.

```
ØØØØH ┌──────────────┐ ⎫
      │              │ ⎬  Directory area
      │              │ ⎪  (128-byte units,
      ├──────────────┤ ⎭  1K bytes max.)
      │              │ ⎫
      │              │ │
      │              │ │
      │              │ │
      │              │ │
      │              │ ⎬  File area
      │              │ │
      │              │ │
      │              │ │
      │              │ │
1FFFH └──────────────┘ ⎭
 ⎛or 3FFFH⎞
 ⎝  7FFFH⎠
```
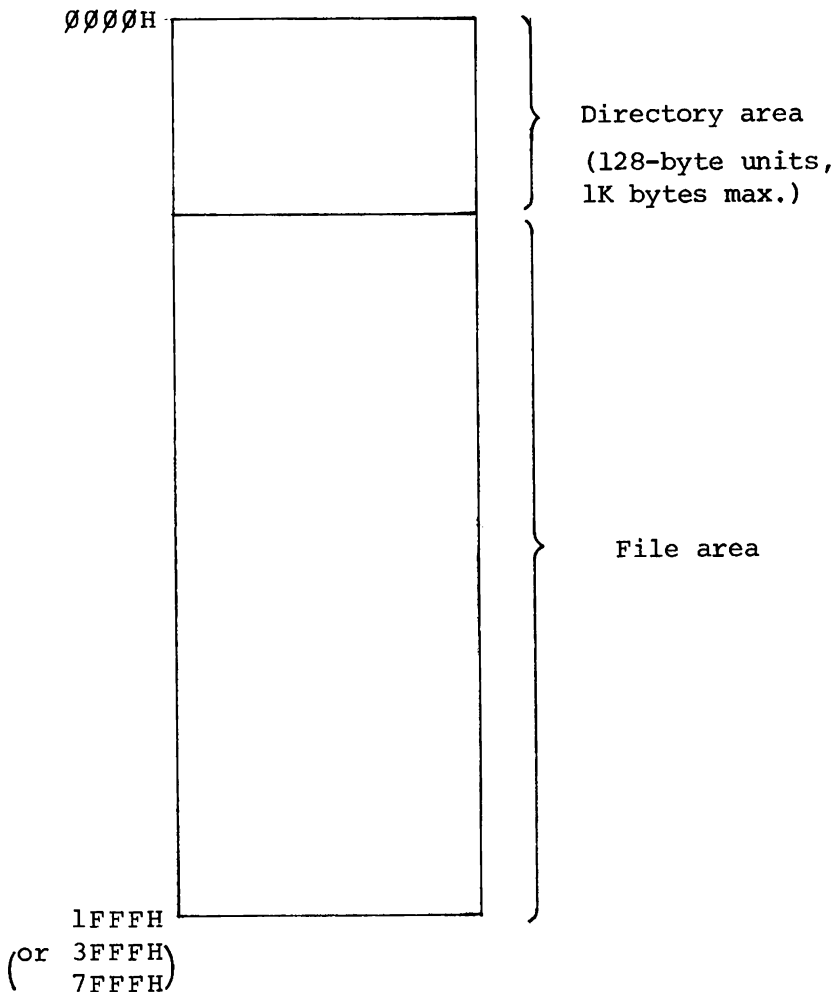
ii) Directory area

1) The directory area is divided into two sections: a header (first 32 bytes) and a directory entry area.

2) Each directory entry is 32 bytes wide and the directory can hold a maximum of 31 directory entries.

3) The directory area is allocated in 128-byte increments up to 1K bytes.

ØØØØH

Header
Directory entry 1
Directory entry 2
Directory entry 3
Directory entry 4

iii) Header format

	Ø	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
ØØØØH	E5H	37H	Capa-city	Check sum L	H	System name			ROM name							
ØØ1ØH	ROM name							"V"	Version No.		M	M	D	D	Y	Y
														Date		

Number of entries
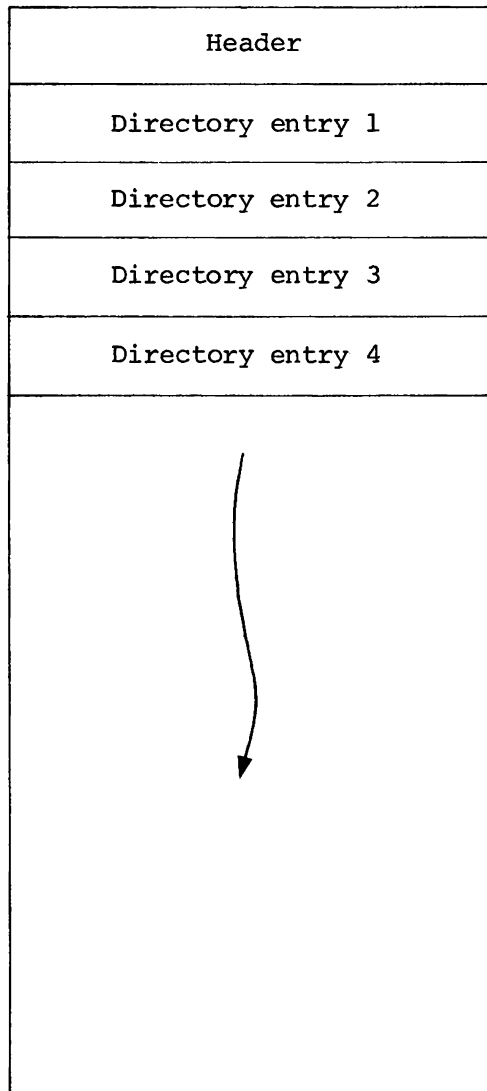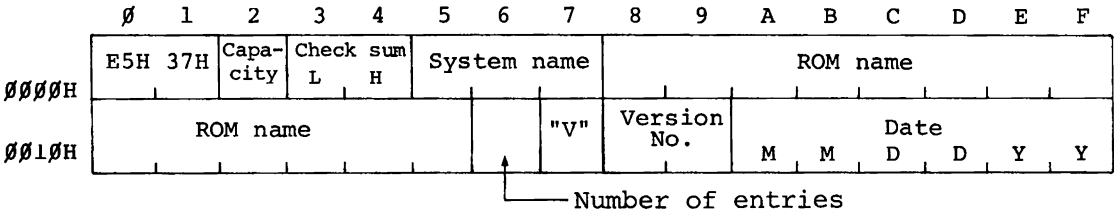
No.	Address	Description
1	0000H – 0001H	Is the ROM identifier. Always set to 0E5H and 37H.
2	0002H – 0002H	Contains the ROM capacity in 1K bytes in binary form. 2764 --- 08H 27128 -- 10H 27256 -- 20H
3	0003H – 0004H	Contains the lowest two bytes of the size of the ROM file area from the beginning of the file area to the end of ROM. 0003H contains the low-order byte and 0004H contains the high-order byte.
4	0005H – 0007H	Contains the user-specified system name.
5	0008H – 0015H	Contains a user-specified ROM name.

No.	Address	Description
6	0016H - 0016H	Contains the number of 32-byte directory entries. The number is either 04H, 08H, 0CH, 14H, 18H, 1CH, or 20H since the directory area is allocated in 128-byte units up to 1K bytes.
7	0017H - 0017H	Set to "V".
8	0018H - 0019H	Contains the ROM version number.
9	001AH - 001FH	Contains the date on which ROM is implemented (latest version).

Fields 1, 2, and 6 must be supplied by the user. The other fields are supplied by the system. (The third field (CHECK SUM) should be filled with correct data though the OS makes no check on that field.)

iv) Directory entry format

The format of the directory entries in memory is the same
as that of the directory entries on the disk.

1	2 File name						3 File type	4	5 ØØH	6 ØØH	7

8 Disk allocation map												

No.	Address	Size (Byte)	Description
1	0H - 0H	1	Contains 00H for a valid directory entry and 0E5H for an invalid directory entry. Invalid entries refer to free entries in a 128-byte unit directory area. In the example below, the 128-byte directory contains five valid entries and two invalid entries (64 bytes).

```
┌─────────────────────────┐ ⎫
│        Header           │ ⎪
├─────────────────────────┤ ⎪
│   Directory entry 1     │ ⎪
├─────────────────────────┤ ⎬ 128 bytes
│   Directory entry 2     │ ⎪
├─────────────────────────┤ ⎪
│   Directory entry 3     │ ⎪
├─────────────────────────┤ ⎭
│   Directory entry 4     │ ⎫
├─────────────────────────┤ ⎪
│   Directory entry 5     │ ⎪
├─────────────────────────┤ ⎬ 128 bytes
│   Invalid directory     │ ⎪
│   entry                 │ ⎪
├─────────────────────────┤ ⎪
│   Invalid directory     │ ⎪
│   entry                 │ ⎭
└─────────────────────────┘
```

No.	Address	Size (Byte)	Description
2	1H - 8H	8	Contains a 1- to 8-character name.
3	9H - BH	3	Contains a 1- to 3-character file type.
4	0CH - 0CH	1	The logical extent number of the current directory entry (00H - 1FH). As described later, one directory entry can manage a file extent of up to 16K bytes. Therefore, two or more directory entries are required for a file larger than 16K bytes. The logical extent number identifies a 16K-byte extent. It starts at 00H.
5	0DH - 0DH	1	Set to 00H.
6	0EH - 0EH	1	Set to 00H.
7	0FH - 0FH	1	Number of records controlled by the directory entry. (0 - 128, in binary). A record is a unit of data accessed by CP/M at a time

No.	Address	Size (Byte)	Description
			and 128 byte long. Since one directory entry can manage up to 16K bytes of data, it can manage a maximum of 128 records.
8	10H - 1FH	16	Disk allocation map. A file is actually controlled in 1K-byte block units in the file area. The block number of the block currently used by the file is indicated here. (Block numbers begin at 1 and are assigned to 1K-byte blocks sequentially from the first block. The file top location differs depending on the directory area size. The file top is indicated in the header.

(3) FD

The structure of the floppy disk is shown below:

```
Ø TRK ┌──────────────────────┐
      │                      │      40 tracks/media
      │    System tracks     │
3 TRK │                      │
4 TRK ├──────────────────────┤      64 sectors/track
      │    Directory         │
      │    ─────────         │
      │                      │
      │      File area       │      128 bytes/sector
      │                      │
      │                      │
      │                      │
38 TRK│                      │
39 TRK├──────────────────────┤
      │        Unused        │
      └──────────────────────┘
```
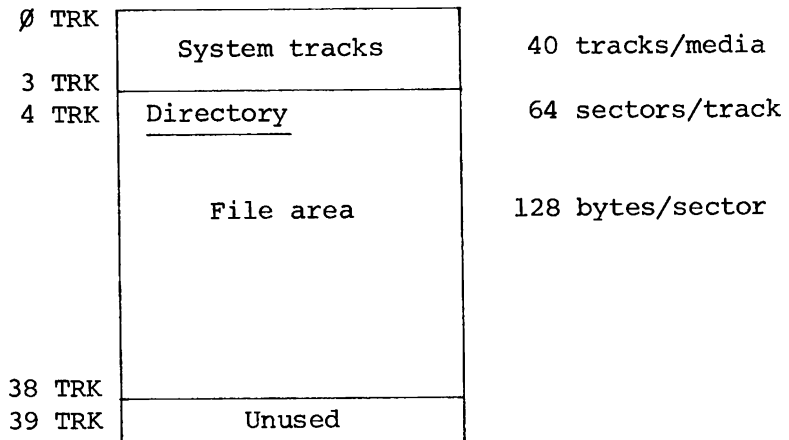
Tracks Ø through 4 contain the boot program for the TF-2Ø floppy disk drive. Sectors 1 through 16 on track 4 are reserved for the directory. The directory can contain up to 64 entries.

The file area starts at sector 17 on track 4 and ends at sector 64 on track 38.

Track 39 is not used so that the MAPLE is compatible with the QC-1Ø/QX-1Ø which does not use track 39. The actual file area size can be calculated as follows:

1 track = 8K bytes

File area = ((40 - 4 - 1) x 8) - 2 = 278K bytes

```
              ↑     ↑   ↑        ↑       ↑
            Total System│      Track  Directory
            number tracks│     size   size
            of          │
            tracks    Unused
                      track
```

(4) Extension unit ROM capsule

Overseas MULTI UNIT 64 and MULTI UNIT II can install a
ROM capsule of up to 1M bit (128K bytes). Overseas OS
version B and up support a ROM capsule as drive I:.

Applicable ROM types are as follows:

    i) 64K bits (8K bytes)

    ii) 128K bits (16K bytes)

    iii) 256K bits (32K bytes)

    iv) 512K bits (64K bytes)

    v) 1M bits (128K bytes)

The the format of the extension unit ROM capsule made up
of 64K- to 256K-bit ROM devices is identical to that of
the ROM capsule described in (2). The format of the
512K- and 1M-bit ROM capsules is also the same except
that provide larger address space.

```
           512K bit                  1M bit

      ┌───────────────┐         ┌───────────────┐
ØØØØH │    Header     │   ØØØØH │    Header     │
      ├───────────────┤         ├───────────────┤
      │   Directory   │         │   Directory   │
      ├───────────────┤         ├───────────────┤
      │               │         │               │
      │     File      │         │     File      │
      │       │       │         │       │       │
      │       │       │         │       │       │
      │       ▼       │         │       ▼       │
FFFFH └───────────────┘  1FFFFH └───────────────┘
```
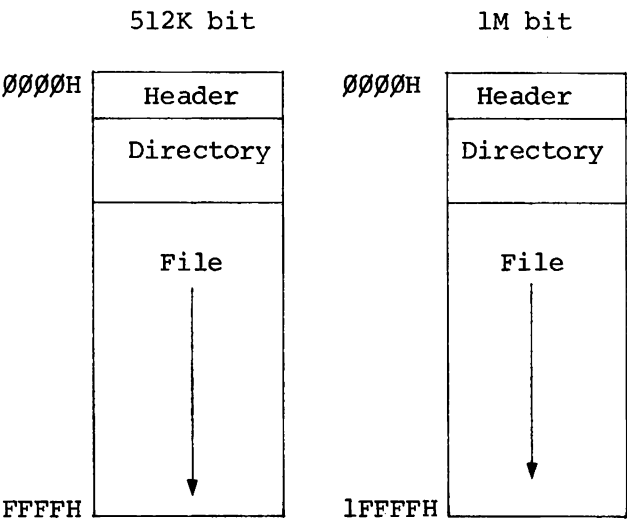
The capacity field at the second byte of the header contains different values for different ROM types.

512K-bit ROM = 40H (64K bytes)

1M-bit ROM = 80H (128K bytes)

15.3   EPSP Protocol

The MAPLE can connect to external disks of the type listed below via a serial interface. Up to two external units (four drives maximum) can be connected in daisy cain configuration.

    TF-20 (5.25 inches, 2 drives)

    TF-15 (5.25 inches, 1 or 2 drives)

    PF-10 (3.5 inches, 1 drive)

CP/M can access these disks as drive D:, E:, F:, and G:. The physical characteristics of the serial interface to external disks are identical to those of the RS-232C interface as shown below.

    Level: +8V

    Baud rate: 38,400 bps

    Data length: 8 bits/word

    Start bit: 1

    Stop bit: 1

    Parity: None.

Logically, CP/M accesses external disks by

communicating with the external disks using the EPSON
Serial Communication Protocol (EPSP).   There are six
OS commands which are used to access external disks
(described on the following pages).

Application programs can access external disks directly
by calling the slave BIOS call (WBOOT + 72H) with
necessary parameters specified.

4) Commands for drives are summarized below.

FMT	DID	STD	FNC	SIZ	Text data No.	Function and text contents
						Reset terminal floppy.
ØØ	SS	MM	ØD	ØØ	ØØ	XX
Ø1	MM	SS	ØD	ØØ	ØØ	Return code ØØ
						Format disk.
ØØ	SS	MM	7C	ØØ	ØØ	Drive code (1 or 2)
Ø1	MM	SS	7C	Ø2	ØØ	High-order byte of the track number of the currently formatting track
					Ø1	Low-order byte of the track number of the currently formatting track $\begin{bmatrix} \text{Ø} - 39 \\ \text{FFFF: end} \end{bmatrix}$
					Ø2	Return code (BDOS error or Ø)
						Read disk direct.
ØØ	SS	MM	77	Ø2	ØØ	Drive code (1 or 2)
					Ø1	Track No. (Ø – 39)
					Ø2	Sector No. (1 – 64)
Ø1	MM	SS	77	8Ø	ØØ 〉 7F	Read in data (128 bytes)
					8Ø	Return code (BDOS error or Ø)
						Write disk direct.
ØØ	SS	MM	78	83	ØØ	Drive code (1 or 2)
					Ø1	Track No. (Ø – 39)
					Ø2	Sector No. (1 – 64)
					Ø3	Contents of C reg. (write type) $(\text{Ø} - 2)^{*1}$
					Ø4 〉 83	Write data (128 bytes)
Ø1	MM	SS	78	ØØ	ØØ	Return code (BDOS error or Ø)

FMT	DID	STD	FNC	SIZ	Text data No.	Function and text contents
						Flush buffer.
ØØ	SS	MM	79	ØØ	ØØ	XX
Ø1	MM	SS	79	ØØ	ØØ	Return code (BDOS error or Ø)

FMT	DID	STD	FNC	SIZ	Text data No.	Function and text contents
						Disk volume.
ØØ	SS	MM	7A	ØØ	ØØ	Drive code (1 or 2)
Ø1	MM	SS	7A	Ø2	ØØ	High-order byte of the track number of the currently copying track
					Ø1	Low-order byte of the track number of the currently copying track $\begin{bmatrix} Ø - 39 \\ FFFF: \ end \end{bmatrix}$
					Ø2	Return code (BDOS error or Ø)

The command 7AH (Copy All disk) is used not by the OS but used by the disk utility program COPYDISK. The function is not supported for one-drive disk systems (PF-1G, for example).

Command Descriptions

FMT: Identifies the header block type.

ØØH: Indicates message transmission from the main unit (MAPLE).

·01H: Indicates message transmission from the FDD.

(All values in FMT through SIZ is in hexadecimal.)

DID: Destination device ID. This identifies the drive
to which the current message (command) is to be
sent when two FDDs are connected in daisy chain
configuration.

31H: First drive (Drive D: or E:)

32H: Second drive (Drive F: or G:)

The device of address of the FDDs (TF-20, for
example) is determined by DIP switches.

SID: Source device ID

Identifying the source of the current message
(command). This field contains 22H if the message
(command) is from the MAPLE.

FNC: Command for FDD.

SIZ: Indicates the text block length (00H - 0FFH). The
value in this field is the length of the actual text
block minus 1.

Text block: A block of data necessary for executing the
command. This block can contain 1 to 256 data
bytes.

1) Reset Terminal Floppy (RESET)

Causes the FDD to initialize itself and wait for an
ENQ block. The FDD returns return code 00 to the
system.

2) Format Disk (FORMAT)

Causes the FDD to format two tracks and return the
corresponding track number (logical numbers) and a
return code to the system. The FDD continues
formatting in two track units and sets the logical
track number in the return message to 0FFFFH when
it completes formatting.

3) Read Disk Direct (READ)

Causes the FDD to transfer the data (128 bytes)
to the system from the disk sector on the specified
logical track at the specified sector number and a
return code to the system. Deblocking technique
(physical to logical conversion of tracks and
sectors) is adopted to speed up this processing.

4) Write Disk Direct (WRITE)

Causes the FDD to write the specified data (128
bytes) to the location on the disk addressed by the
specified logical track and sector numbers.
Actually, this command only places the specified
data into the 1K-byte host buffer because of the

blocking technique (logical to physical conversion

of tracks and sectors).

5) Flush Buffer (WRITEHST)

Causes the FDD to flush the contents of the 1K-byte

buffer filled by the WRITE command onto the disk.

6) Copy Volume

    Causes the FDD to copy the entire diskette on the
    specified drive onto another diskette. This
    command is not available if the system has only one
    drive.

7) Return codes

Return code (hex)	Meaning	
00	Normal termination	
FA	BDOS error	Read error
FB		Write error
FC		Drive select error
FD		Write protect
FE		

*1: The third byte of the data block for FNC=78H

    indicates the write mode:

    00H: Standard write (The FDD blocks data before
         write.)

    01H: Flush buffer (The FDD immediately writes data
         on the FD without blocking.)

    02H: Sequential write

00H is used when writing ordinary files. 01H is used
only when writing directories.

Other commands

The FDD supports some other commands in addition to the
six commands used by the MAPLE. Refer to FDD manuals
for further information on these commands. They can
also be activated easily by calling the slave BIOS
function (WBOOT + 72H).

15.4 DIP Switches

The table below lists the uses of the DIP switches on
the main unit back panel.

Uses of DIP switches

SW	Overseas version	Kana and Japanese-language version
1	Identifies the keyboard type.	Identifies the keyboard type. ∅ = Kana keyboard 1 = Japanese-language keyboard or touch type keyboard
2		Not used.
3		Not used.
4		Not used.
5	Specifies whether the check sum is to make a check at power-on time when the RAM disk unit 6∅ or 12∅ is connected.' ∅ = No check made 1 = Check made	⟵
6	Specifies the range of code conversion to be used during screen dump. ∅ = Converts ∅∅H – 1FH, 7FH, or ∅FFH to a space. 1 = Converts ∅∅H – 1FH or 7FH – ∅FFH to a space.	Not used.
7	Not used.	Not used.
8	Not used.	Not used.

Chapter 16   Extension Units

The capability of the MAPLE can be extended easily by
connecting various extension units via the system bus.
These extension units are accessible through I/O
addresses 80H through 0FFH. (80H through 0DFH are
reserved for EPSON-supplied extension units. I/O
addresses 0E0H through 0FFH are available for user-
supplied extension units.)

This chapter describes the following extension units
with focus mainly given on their specifications and
functions as viewed from the software standpoint:

    1. Nonintelligent RAM disk unit

    2. Intelligent RAM disk unit

    3. Direct modem unit

    4. Multi Unit 64

    5. Multi Unit II

The Japanese-language processing unit may also be
attached addition to the above units. Details on the
Japanese language processing unit are discussed in a
separate manual.

16.1 Nonintelligent RAM Disk Unit

The nonintelligent RAM disk unit is classified as two types according to their RAM capacity: 64K- and 128K- byte RAM versions. The nonintelligent RAM disk unit is not available as a stand-alone extension unit but is located on the following extension units:

- Japanese-language processing unit 64 (Model H105A)

    64K RAM disk + Japanese-language processing unit

- Japanese-language processing unit 128 (Model H106A)

    128K RAM disk + Japanese-language processing unit

- Japanese-language processing unit 64 (Model H110A)

    64K RAM disk + Touch-type Japanese-language

    processing unit

- Japanese-language processing unit 128 (Model H111A)

    128K RAM disk + Touch-type Japanese-language

    processing unit

- Multi Unit 64 (Model H108A)

    64K RAM disk

    ROM capsule

    Direct modem

- Multi Unit II (Model H115A)

    64K RAM disk

ROM capsule

Synchronous communication unit

RAM file hardware

(1) Memory access method

This unit is allocated in some I/O addresses of the
MAPLE main unit. All operations on this unit can be
controlled by issuing I/O instructions.

Data in memory can be read from or written into this
unit by reading or writing the access port after loading
the correct address in the address register.

The address register is automatically incremented as
an access is made to memory. However, the highest eight
bits of the address register are not incremented because
only the lowest eight bits work as a counter. This
limits the number of data bytes to 256 that can be
transferred to or from RAM in a single read or write
operation.

Since the RAM file is closed in the normal state for
saving power, an open command (described later) must be
executed before an attempt is made to access the file.

(2) Address map

RAM is allocated in memory as illustrated below.

This starting address must be loaded in the address register when accessing the RAM file.

(2) Address map
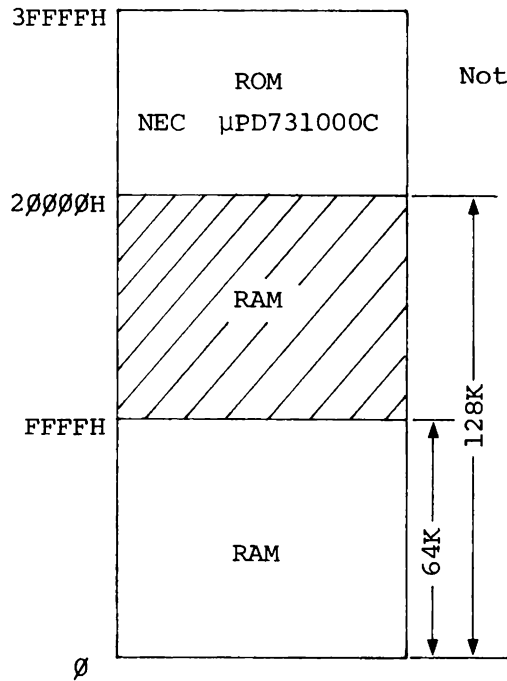
```
3FFFFH ┌─────────────────────┐
       │        ROM          │      Note:  MPD731000 Chip Select
       │  NEC   µPD731000C   │             pin must be programmed
       │                     │             to low active.
 2ØØØØH ├─////////////////////┤  ▲               ▲
       │ //////////////////// │  │               │
       │ ////////RAM///////// │  │               │
       │ //////////////////// │  │               │
 FFFFH ├─////////////////////┤  │ 64K           │ 128K
       │                     │  │               │
       │        RAM          │  │               │
       │                     │  ▼               │
    Ø  └─────────────────────┘  ▼               ▼
```

Memory structure

```
ØØØØØH   ┌─────────────────┐   ØTRK 1SEC
         │                 │                        Directory entries = 32
         │ Directory area  │   ØTRK 4SEC
ØØ4ØØH   ├─────────────────┤   ØTRK 5SEC
         │                 │
         │                 │
         │   File area     │
         │                 │
         │                 │
         │                 │
         │                 │                    (  ):  For 128K RAM
         │                 │
ØFFFFH   └─────────────────┘   7TRK 64SEC
(or 1FFFFH)                    (or 15TRK 64SEC)
```
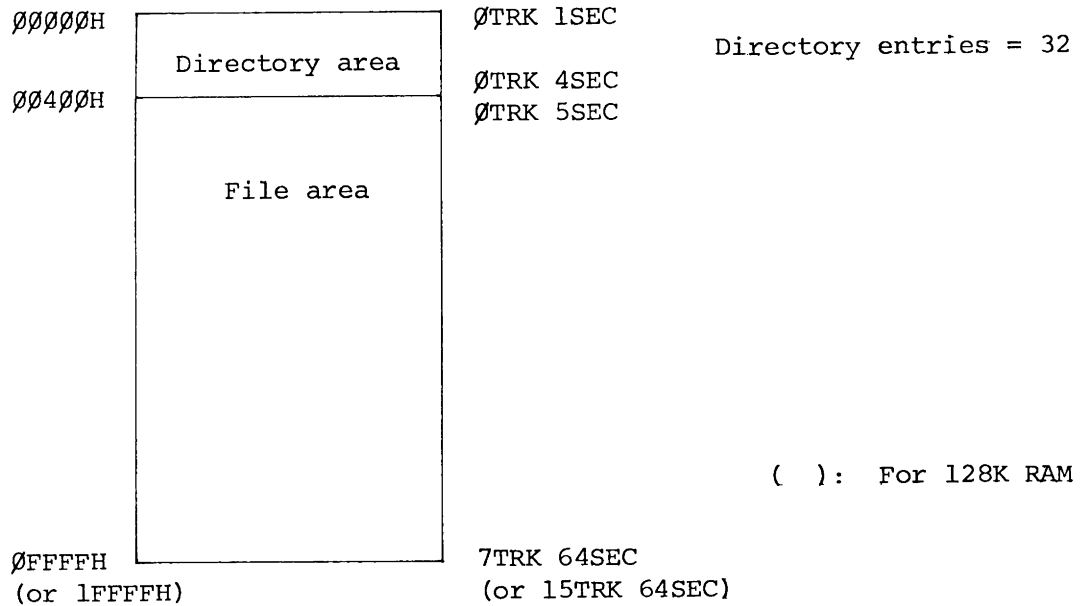
The first 1K bytes of RAM makes up the directory area and the rest is used as the file area. Data (directory and file) is stored in RAM in the same format as it is on FD.

(3) RAM/ROM file register format

① Address register 1   WRITE ONLY

	D7							Ø
90H	A7	A6	A5	A4	A3	A2	A1	AØ
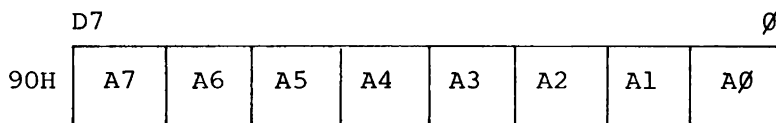
② Address register 2   WRITE ONLY

	D7							Ø
91H	A15	A14	A13	A12	A11	A10	A9	A8
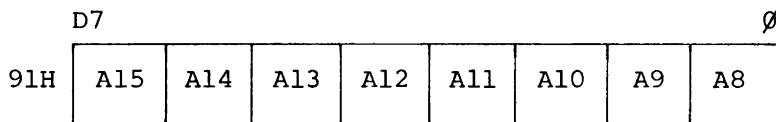
③ Address register 3   WRITE ONLY

	D7							Ø
92H	X	X	X	X	X	A18	A17	A16

X: No care

④ Access port   WRITE/READ

	D7							Ø
93H	Corresponds to the data in memory							
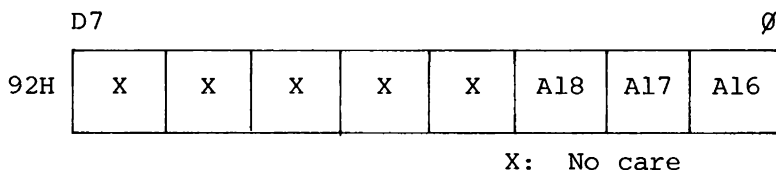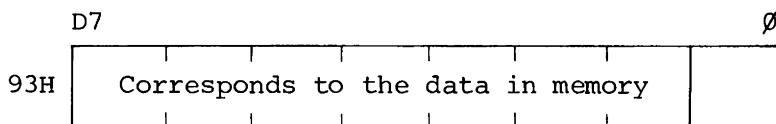
⑤ Command register   WRITE

	D7							Ø
94H	X	X	X	X	X	X		

← 1: Write protect
   Ø: Write enabled

1: RAM file open
Ø: RAM file close

⑥ Status register   READ

	D7							Ø
94H		X	X	X	X		Same as the command register	

1: Not installed on this unit
Ø: Installed on this unit

1: RAM 128KB
Ø: RAM 64KB

16-7
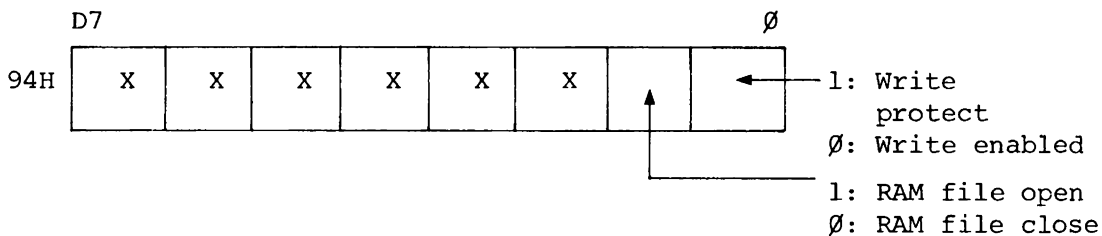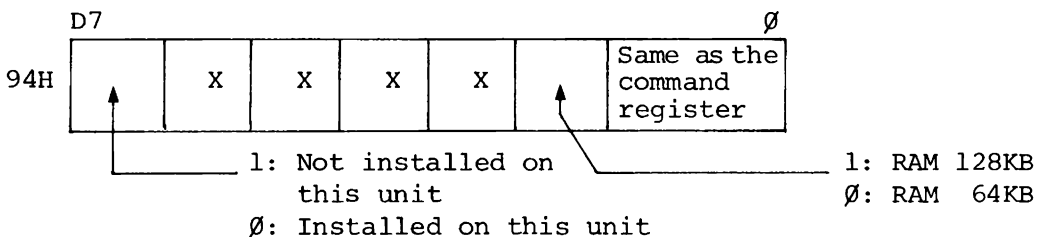
(4) Opening/closing a RAM file

It is recommended that the RAM file be opened only when accessed to save electric power. The state transitions for file opening and closing processing is shown below.

OPEN COMMAND

CLOSE   OPEN

CLOSE COMMAND
POWER OFF
RESET
HALT
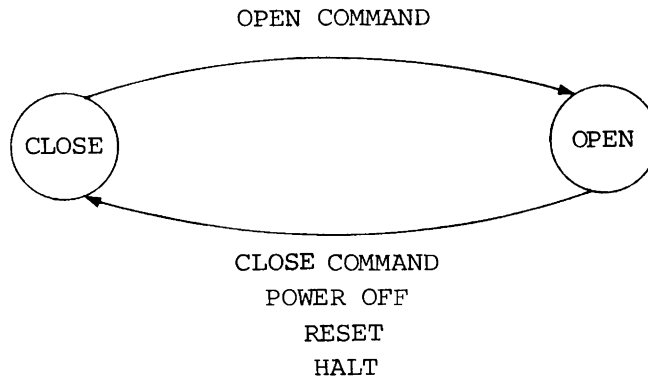
As illustrated above, an OPEN command must be executed before accessing the RAM file. RAM is self-refreshing when the file is closed and so must not be accessed in the closed state. The address register does not increment while the file is closed.

The file may be destroyed if RESET is made active while it is in the open state.

Note: No OPEN command is required when accessing the file ROM.

(5) Write protect

Write protect is enabled (write protected) after a power on or reset. In the write protect mode, RAM is disabled for write and can only be read. The address register does not increment in this mode.

16.2 Intelligent RAM Disk Unit

The MAPLE can control writes and reads to and from the intelligent RAM disk simply by sending commands as is the case with the FDD. The RAM disk capacity is 60K or 120K bytes.

(Although an intelligent RAM disk unit contains 64K- or 128K-byte RAM, 4K or 8K bytes are reserved for the unit program; that is, the user can actually use 60K or 120K bytes of RAM, respectively.)

The following options are available:

- RAM DISK UNIT 60 (Model H102A)

     60K-byte RAM disk

-RAM DISK UNIT 120 (Model H103A)

     120K-byte RAM disk

(1) Block diagram
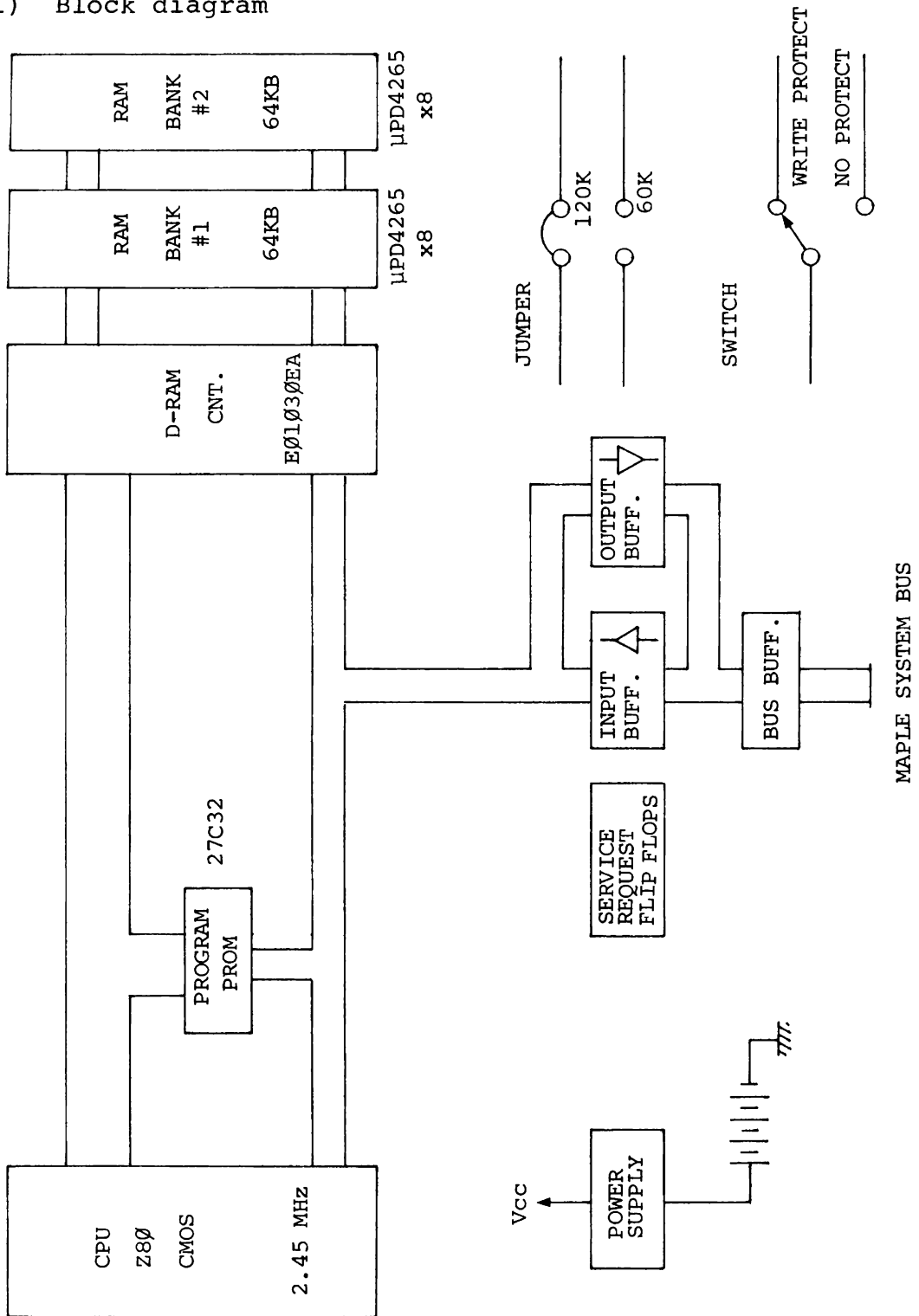
- Jumpers

The RAM disk unit has two jumper pins on its main board.
The firmware determines the memory capacity by checking
the state of these pins.

The jumper pins are labeled 60K and 120K with silk
printing, respectively. Install the jumper for 60K when
eight RAM chips are used and for 120K when 16 RAM chips
are used.

- Switch

The RAM disk unit has a switch on its main board. This
switch can be controlled externally and used as a write
protect switch. When the switch is set to on, RAM is
disabled for write and enabled only for read.

(2) Memory map

1. Memory map


```
              BANK #1                    BANK #2

      ┌───────────────┐ ┌ ─ ─ ─ ─ ┐┌───────────────┐
ØØØØH │               │             │               │
      │               │             │               │
      │               │             │               │
      │               │             │               │
      │               │             │               │
      │  User area    │             │  User area    │
      │               │             │               │
      │               │             │               │
      │               │             │               │
      │               │             │               │
FØØØH ├───────────────┤             ├───────────────┤
      │   Firmware    │             │ Cannot be used│
FFFFH └───────────────┘ └ ─ ─ ─ ─ ┘└───────────────┘
```
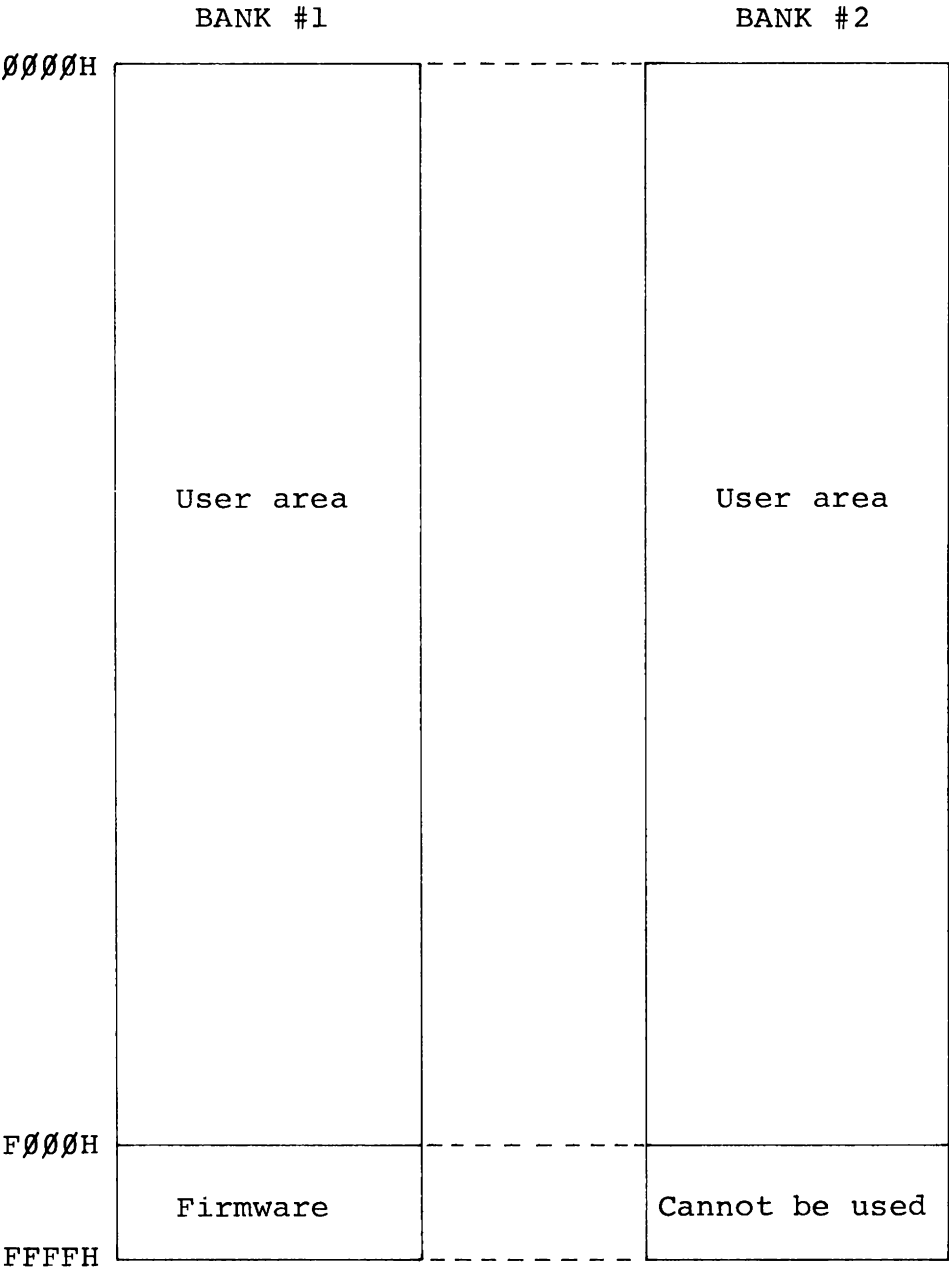
3. Firmware memory map

```
F000H ┌─────────────────┐
      │   BANK #1        │
      │   Checksum       │
F1FFH ├─────────────────┤
F200H │   BANK #2        │
      │   Checksum       │
F3FFH ├─────────────────┤
F400H │                 │
      │                 │
      │                 │
      │   Procedure     │
      │                 │
      │                 │
      │                 │
FBFFH ├─────────────────┤
      │   Vector        │
EC00H ├─────────────────┤
      │                 │
      │   Work          │
      │                 │
      ├─────────────────┤
      │   Stack         │
FFFFH └─────────────────┘
```

16-13

Disk format

The RAM disk unit is formatted as shown below.

Track						
Ø	Sector Ø		BANK #1			Sector 3F
1	Ø			#1		3F
2	Ø			#1		3F
3	Ø			#1		3F
4	Ø			#1		3F
5	Ø			#1		3F
6	Ø			#1		3F
7	Ø	#1	1F	2Ø	#2	3F
8	Ø		BANK #2			3F
9	Ø			#2		3F
A	Ø			#2		3F
B	Ø			#2		3F
C	Ø			#2		3F
D	Ø			#2		3F
E	Ø			#2		3F

Up to track 7, sector
1FH can be addressed
for 64K RAM disk.

(3) I/O map as viewed from the MAPLE

The MAPLE communicates with the RAM disk unit through
two I/O addresses 80H and 81H. The MAPLE functions for
communicating with RAM disk units are listed below.

Address	Read/ write	MAPLE operation
80H	R	Read data status
	W	Write data
81H	R	Read handshake information
	W	Write command

The MAPLE can check whether a RAM disk unit is
installed by reading I/O address 81H. Its highest two
bits are set to 00 when a RAM disk unit is installed.

(4) Communication with the RAM disk unit

1. Communication sequence

The MAPLE must always check the IBF and OBF bits
(handshake information) when communicating with a RAM
disk unit.

```
        ┌─────────┐                        ┌─────────┐
        │  READ   │                        │  WRITE  │
        └────┬────┘                        └────┬────┘
             │ ◄──────┐                         │ ◄──────┐
           ��= =Ø│                      ◇diamond   =1│
          <  IBF  >─────┘                     <  OBF  >─────┘
             │ =1                              │ =Ø
        ┌────┴────┐                        ┌────┴────┐
        │  Input  │                        │ Output  │
        │  data   │                        │  data   │
        └────┬────┘                        └────┬────┘
             ▼                                  ▼
```
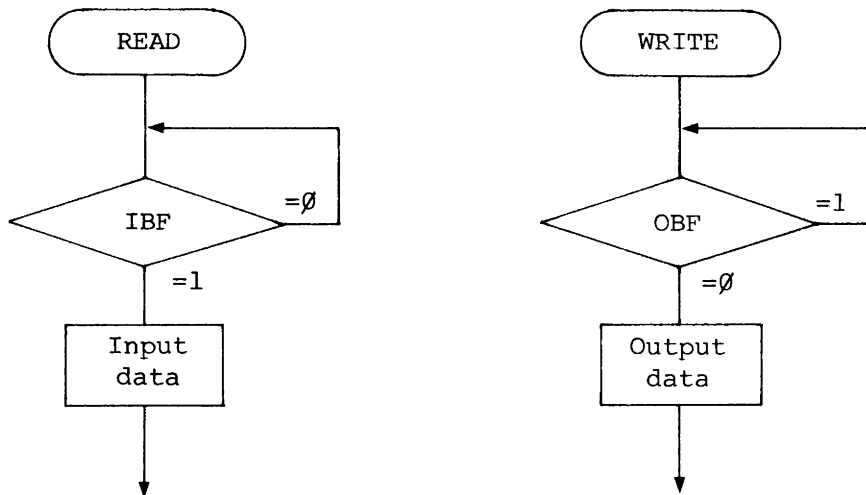
2. Handshake information

Handshake information (81H)

MSB							LSB
Ø	Ø	Un-defined	Un-defined	Un-defined	Un-defined	OBF 0: EMPTY 1: FULL	IBF 0: EMPTY 1: FULL

Loaded with ØØ when the RAM disk unit is connected.
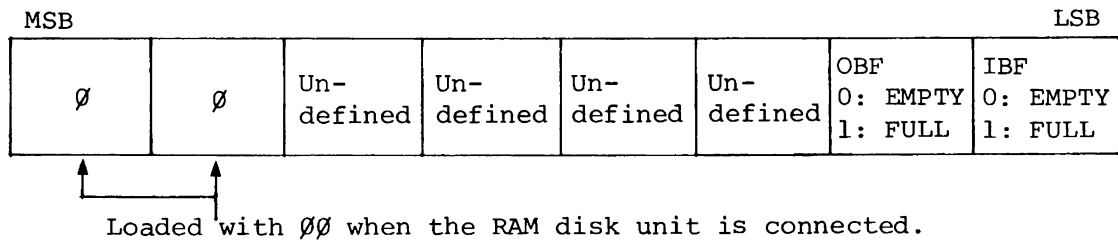
Note: The application program must always check the
         handshake information when sending or receiving
         a command, data, or status to or from the RAM
         disk unit.

OBF   Ø: Command or data transfer from MAPLE is
           enabled.

       1: Command or data transfer from MAPLE is
           disabled.

IBF   Ø: Receive data is present.

       1: Receive data is not present.

(5) RAM disk commands

The RAM disk unit processes the following six commands:

- RESET (Ø0)   Reset
- READ (Ø1)   Read Sector
- READB (Ø2)   Read Byte
- WRITE (Ø3)   Write Sector
- WRITEB (Ø4) Write Byte
- CKSUM (Ø5)   Check Entire Memory

Note 1: The RAM disk ignores commands other than the
        above as well as invalid data.
Note 2: If a new command is issued before completion of
        the current command, the RAM disk starts the
        protocol sequence of the new command ignoring
        the old command.
Note 3: When the write protect switch is set to ON, the
        RAM disk unit returns an error code without
        changing the RAM contents.
Note 4: The RAM disk contents are preserved even when
        MAPLE power is turned off.

(5-1) RESET (ØØH)

Resets the RAM disk unit and returns its status.

```
                      MSB              LSB
        RESET (C)    │ Ø Ø Ø Ø Ø Ø Ø Ø │
```

Subsequently, the following 1-byte status information

is returned from the RAM file:

```
MSB                                  LSB
┌─────┬─────┬─────┬─────┬─────┬─────┬─────┬──────┐
│  Ø  │  Ø  │  Ø  │  Ø  │  Ø  │  Ø  │ WPT │120K/ │
│     │     │     │     │     │     │     │ /60K │
└─────┴─────┴─────┴─────┴─────┴─────┴─────┴──────┘
                                          │
                              │     0: RAM 60K SYSTEM
                              │     1: RAM 120K SYSTEM
                              │
                              │     0: WRITE PROTECT SW  OFF
                              │     1: WRITE PROTECT SW  ON
```
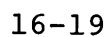
(5-2) READ (Ø1H)

Causes the MAPLE to read a sector from the RAM disk unit.

```
                    MSB                 LSB
        READ   (C) | Ø Ø Ø Ø Ø Ø Ø 1 |

        TRACK  (D) |                 |   Ø - ØEH

        SECTOR (D) |                 |   Ø - 3FH
```
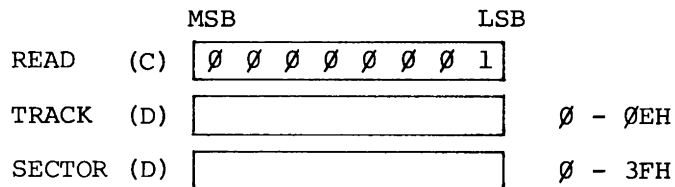
Subsequently, a 1-byte status information and 128 byte of data are returned from the RAM disk unit in that order.

128-byte data is returned even if the status byte indicates an error condition (in this case, the validity of the data is not guaranteed).

Status information

```
MSB                                             LSB
| Ø | Ø | Ø | Ø | Ø | CKSM | SECTOR | BANK |
                     ERR    ERR      ERR
```

0: OK
1: Read from invalid RAM bank

0: OK
1: Read from invalid track or sector

0: OK
1: Checksum error
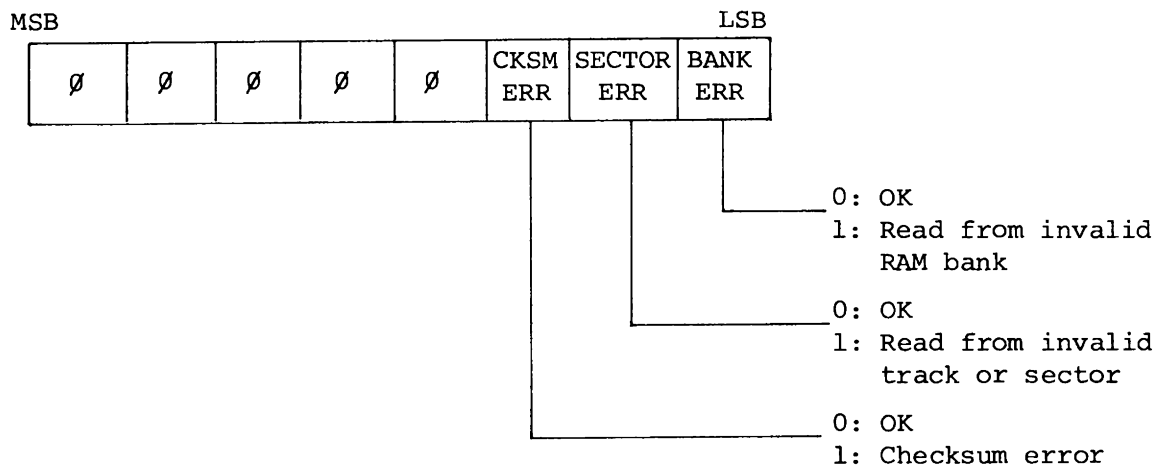
(5-3) READB (02H)

Causes the MAPLE to read a byte from the RAM disk unit.

```
READB (C) | 0 0 0 0 0 0 1 0 |

BANK  (D) | _____ |   1 or 2  (2 may be specified only when
                                        there are two banks.)

ADDRH (D) | _____ |   0 - 0EFH ⎫
                                       ⎬ Address 0F000H and higher
ADDRL (D) | _____ |   0 - 0FFH ⎭ are invalid.
```

Subsequently, a 1-byte status information and 1-byte data are returned from the RAM disk unit in that order.

Data is returned even if the status byte indicates an error condition (in this case, the validity of the data is not guaranteed).

Status information

```
MSB                                    LSB
| 0 | 0 | 0 | 0 | 0 | CKSM | ADDR | BANK |
                     ERR  | ERR  | ERR  |
```

                                        0: OK
                                        1: Read from invalid
                                           RAM bank

                                        0: OK
                                        1: Address error

                                        0: OK
                                        1: Checksum error
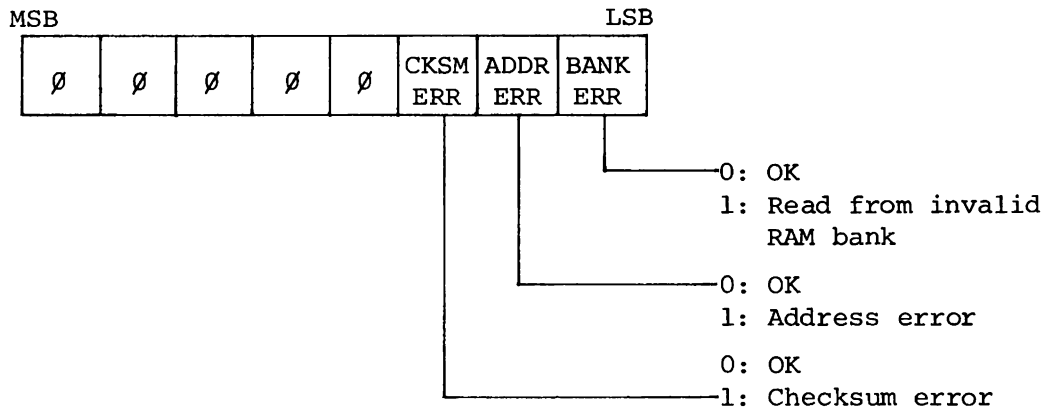
(5-4) WRITE (03H)

Causes the MAPLE to write data onto a sector in the RAM disk unit.

```
WRITE   (C)   │ 0 0 0 0 0 0 1 1 │
TRACK   (D)   │               │   0 - 0EH
SECTOR  (D)   │               │   0 - 3FH
```

Subsequently, 128-byte data is sent to the RAM disk unit. A 1-byte status information is returned from the RAM disk unit after the data is written.

The status byte contains a nonzero value if an error is detected. In this case, the RAM disk unit discards the data.

Status information

```
MSB                                    LSB

│  0  │  0  │  0  │  0  │  0  │ WPT │SECTOR│BANK │
                                 ERR │ ERR │ ERR │
                                                 └──── 0: OK
                                                       1: Write to invalid
                                                          RAM bank
                                           └──── 0: OK
                                                 1: Write to invalid
                                                    track or sector
                                 └──── 0: OK
                                       1: Write protected
```
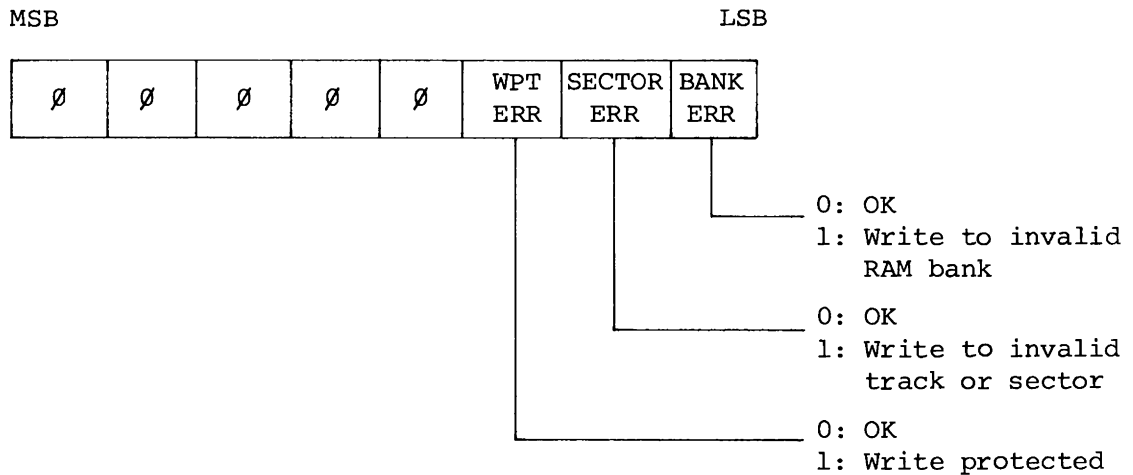
(5-5) WRITEB (04H)

Causes the MAPLE to write a byte onto the RAM disk unit.

```
WRITEB  (C)  | 0  0  0  0  0  1  0  0 |
BANK    (D)  |                        |  1 or 2   (2 may be specified only when
                                                  there are two banks.)
ADDRH   (D)  |                        |  0 - 0EFH ⎫
                                                  ⎬ Address 0F000H and higher
ADDRL   (D)  |                        |  0 - 0FFH ⎭ are invalid.
```

Subsequently, 1-byte data is sent to the RAM disk

unit. A 1-byte status information is returned from the

RAM disk unit after the data is written.

The status byte contains a nonzero value if an error is

detected. In this case, the RAM disk unit ignores the

byte.

Status information

```
MSB                                   LSB
| 0 | 0 | 0 | 0 | 0 | WPT | ADDR | BANK |
                     ERR   ERR    ERR
                      |     |       |_____ 0: OK
                      |     |                   1: Write to invalid
                      |     |                      RAM bank
                      |     |_____ 0: OK
                      |                       1: Write to invalid
                      |                          track or sector
                      |_____ 0: OK
                                           1: Write protected
```
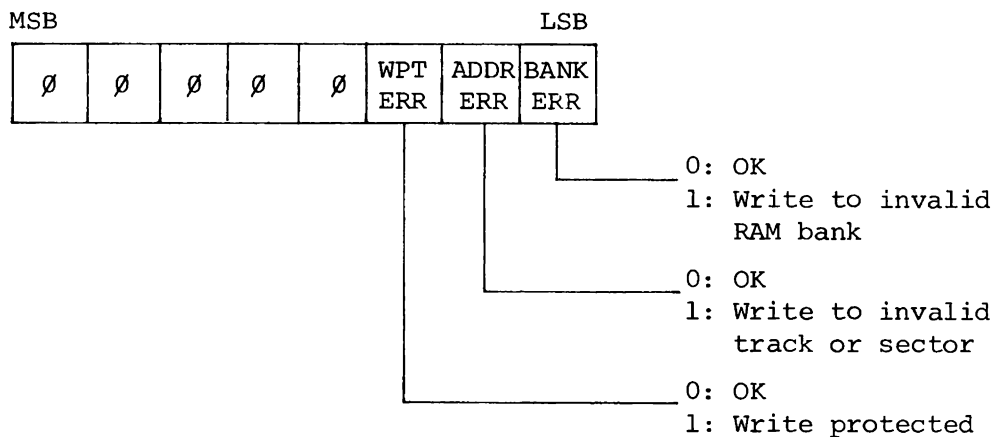
(5-6) CKSUM (Ø5H)

Causes the MAPLE to perform a checksum test on the
entire RAM on the RAM disk unit.

```
                MSB              LSB
CKSUM (C)      | Ø  Ø  Ø  Ø  Ø  1  Ø  1 |
```
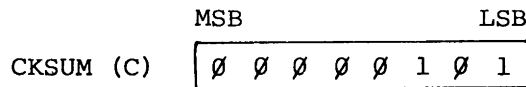
Subsequently, a 1-byte status information is returned
from the RAM disk unit.

Status information

```
MSB                                          LSB
| Ø | Ø | Ø | Ø | Ø | Ø | BANK2 | BANK1 |
                          ERR     ERR
```

0: Normal
1: Checksum error
   on bank 1

0: Normal
1: Checksum error
   on bank 2

16-24
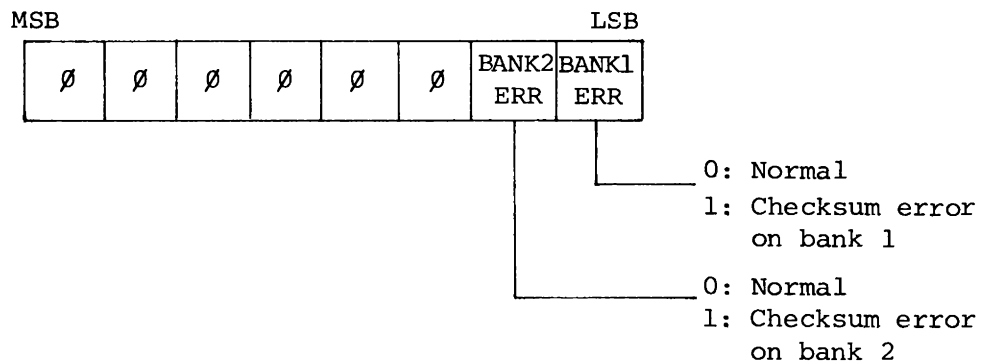
16.3 Direct Modem Unit

The direct modem is installed in the expansion units
shown below. It is available only in the U.S.A.

- Modem Unit

    Consists of a direct modem unit only.

- Multi Unit 64

    Consists of a direct modem unit, 64K RAM disk, and

    a ROM capsule.

The pages that follow explain the specifications for and
functions of the direct modem unit.

1. Outline

The direct modem unit has the following features:

(1) Modem communication function

BELL103 (ORIG/ANS) compatible. Full duplex
communication at speeds up to 300 bps is possible using
a telephone line.

(2) Can be connected to a telephone line directly or
through an acoustic coupler. The standard direct modem
unit is provided with a telephone line interface
certified by FCC. It is also connectable using an
optional acoustical coupler unit.

(3) Communication function

Audio communication using the optional handset is
possible.

(4) Monitoring function

The line status can be monitored through the speaker
in the MAPLE main unit.

(5) Automatic dialing and answering

Both pulse or tone dialing are possible. Automatic
answering is possible using a telephone ring detection
circuit.

Handset
(optional)

(MAPLE main unit)

Expansion bus

Flat cable
#727

Cable
#735

Telephone
line

Acoustic coupler unit
(optional)
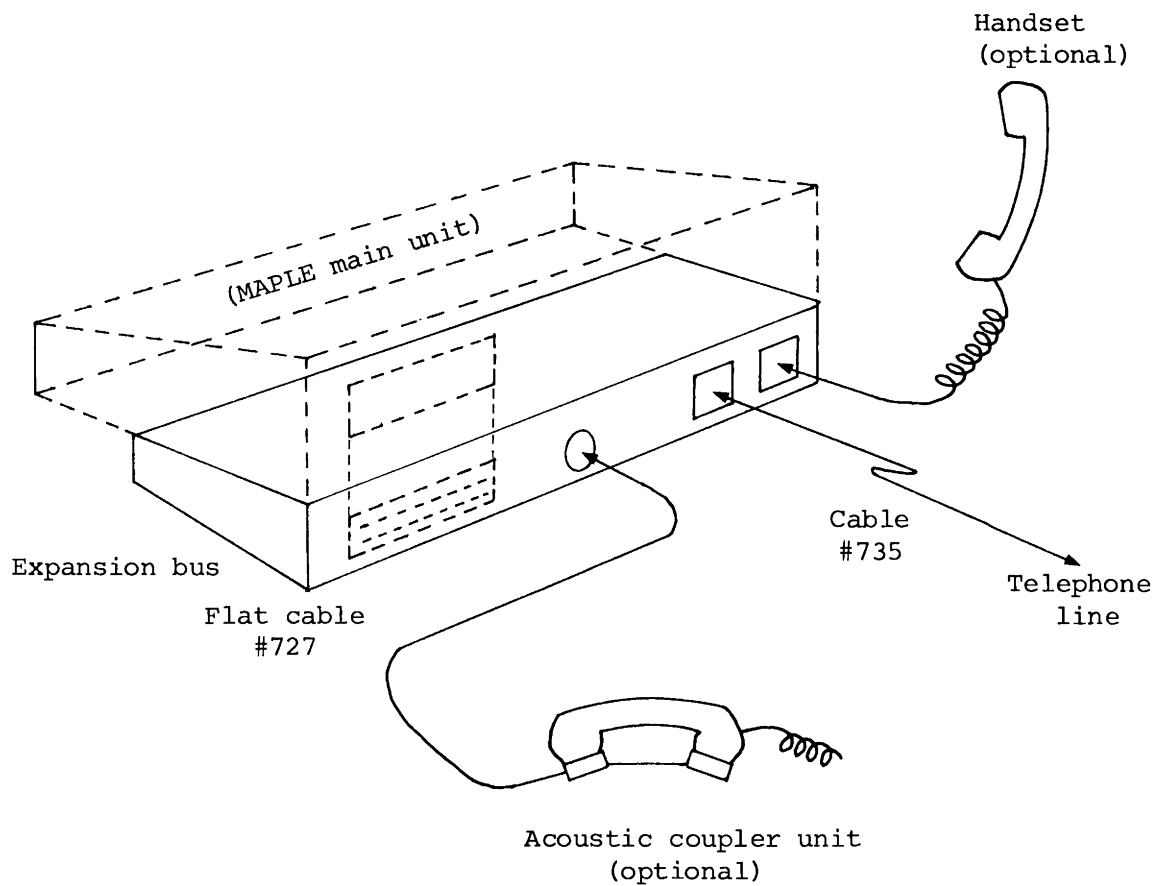
< MAPLE rear panel >
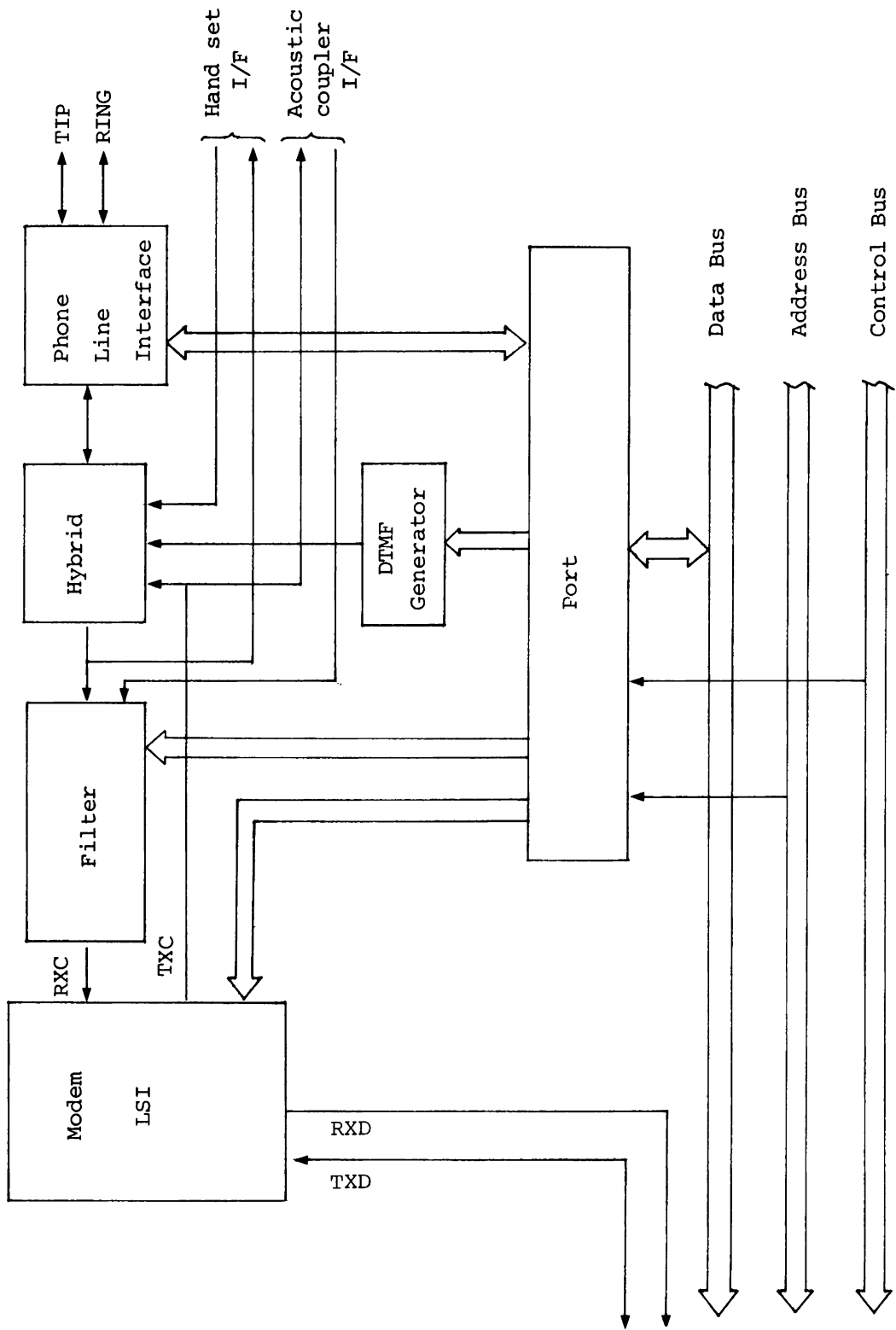
< Block diagram >

2. Basic hardware specifications

2.1 Modem

Compatibility ---------- BELL1Ø3

                        Originate or answer mode

Data rate ------------- Ø - 3ØØ bps

Modulation ------------ FSK

Handset sensitivity -- -8 - -43 dBm

Transmitter output level

                 --- B -12+3 dBm

Carrier frequencies

Mode		Mark	Space
Originate	Transmission	127Ø Hz	1Ø7Ø Hz
	Receive	2225 Hz	2Ø25 Hz
Answer	Transmission	2225 Hz	2Ø25 Hz
	Receive	127Ø Hz	1Ø7Ø Hz

2.2 Telephone line interface

(1) FCC Part 68 certification No.: BKM9A8-12717-DT-E

(2) Ringer equivalence: 0.5B

(3) Connector: RJ11C, RJ11W

(4) Impedance

    On-hook, DC: 20M ohms or more (across TIP and RING

                   electrodes and ground, 200VDC

                   bipolar)

    On-hook, AC: 20K ohms or less across TIP and RING

                   electrodes.

    Off-hook, DC: 200 ohms $\pm$ 20 % across TIP and RING

                   electrodes.

    Off-hook, AC: 600 ohms $\pm$ 20 % across TIP and RING

                   electrodes.

                   (when measured at least two seconds

                   after off-hook.)

(5) Insulation:

    1000 volts r.m.s across TIP and RING and the other

    electrodes.

(6) Surges:

    Must withstand surges with a peak voltage of 1500V, a

    rise time shorter than 10 μsec and a fall time

    shorter than 160 μsec. Surge voltage is measured

    between TIP and RING and between these terminals and

    ground.

(7) Ring detection

    Detection frequency: 16 to 68 Hz

    Voltage: 40 to 150 volts r.m.s

2.3 Acoustic coupler

The MAPLE can serve as an acoustic coupler when furnished with an optional acoustic coupler unit.

    Sound output: +5 dB max. ($1N/m^2 = 0$ dB)

2.4 Power consumption

	Typ.	Max.
1) MAPLE power off time	3 μA	10 μA
2) MAPLE power on time (when no modem is used)	0.4 mA	1 mA
3) Modem power on (line not connected w/t coupler)	50 mA	60 mA
4) Modem power on (line connected)	65 mA	80 mA

3.1 Telephone line interface

Certified by FCC Part 68.

This interface can be connected to a RJ11C or RJ11W type modular jack via the attached modular cable.

3.2 Handset interface

The MAPLE can be used as a telephone when equipped with an optional handset unit.

The handset interface can be enabled or disabled by the MAPLE controlling a dedicated output port. The connector is of modular jack type which is commonly used in U.S.A. for receivers. Electrically, however, the MAPLE is designed for connection to an EPSON-supplied receiver unit.

3.3 Acoustic coupler interface

This interface connects the optional acoustic coupler unit to the MAPLE.

Connector to be used: HSJ0863-01-440 (HOSHIDEN)
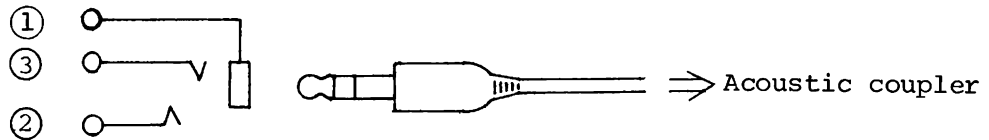
① : GND
② : ACMI (microphone)
③ : ACSP (speaker)

4. I/O Ports

4.1 Outline

The H107A is controlled directly by the MAPLE main CPU via I/O ports.

The H107A is assigned the following I/O ports:

Address	Input port	Output port
84H	Inhibited	Tone dialer control
85H	Inhibited	Model control
86H	Modem status	Inhibited
87H	Inhibited	Port mode

4.2 Address 84H

I/O: Output only

Use: For tone dialer control

D7	D6	D5	D4	D3	D2	D1	DØ
Ø	Ø	Ø	TONE	F3	F2	F1	FØ

Always set to 0.

Selects DTMF signal frequency.

Turns DTMF signal on/off
  1: Tone on
  Ø: Tone off

ØØØØ**** = Tone off
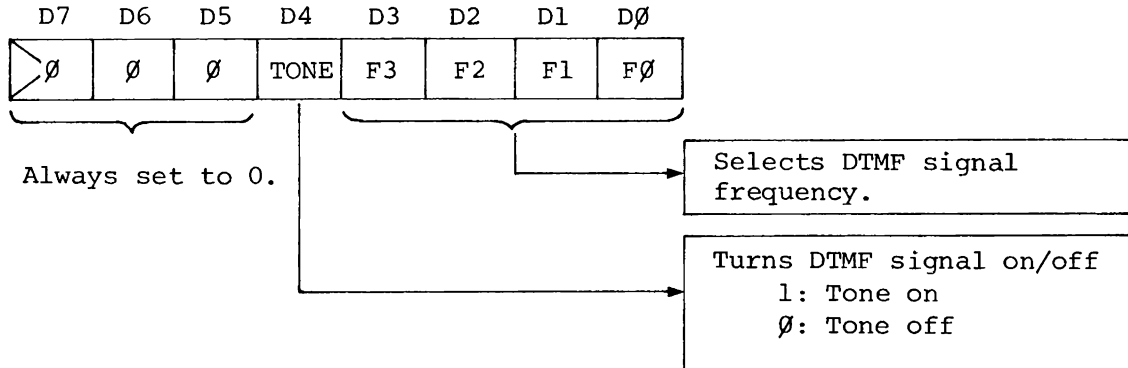
$ØØØ1F_3F_2F_1F_Ø$ = Tone on with the frequency specified by FØ through F3.

4.3 Address 85H

I/O: Output only

Use: For modem control

D7	D6	D5	D4	D3	D2	D1	DØ
CCT	PWR	TEST	ANS	TXC	MON	HSC	OHC

Off-hook control

1: Off hook
Ø: On hook

Handset control

1: Handset on
Ø: Handset off

Monitor control

1: Monitor on
Ø: Monitor off

Transmission carrier control

1: Transmission carrier on
Ø: Transmission carrier off

Modem mode

1: ANS mode
Ø: ORIG mode

Test function

1: Analog loop test
Ø: Normal operation

Modem power control

1: Power on
Ø: Power off

Line connection

1: Connected
Ø: Not connected

4.4 Address 86H

I/O: Input only

Use: For modem status read

D7	D6	D5	D4	D3	D2	D1	DØ
Ø	✕	✕	✕	✕	CD̄	✕	RNḠ

Reserved

Bell detection

1: Normal
Ø: Bell ringing

Carrier detected

1: Not detected
Ø: Detected

Modem unit installation check bit

1: Not installed
Ø: Installed
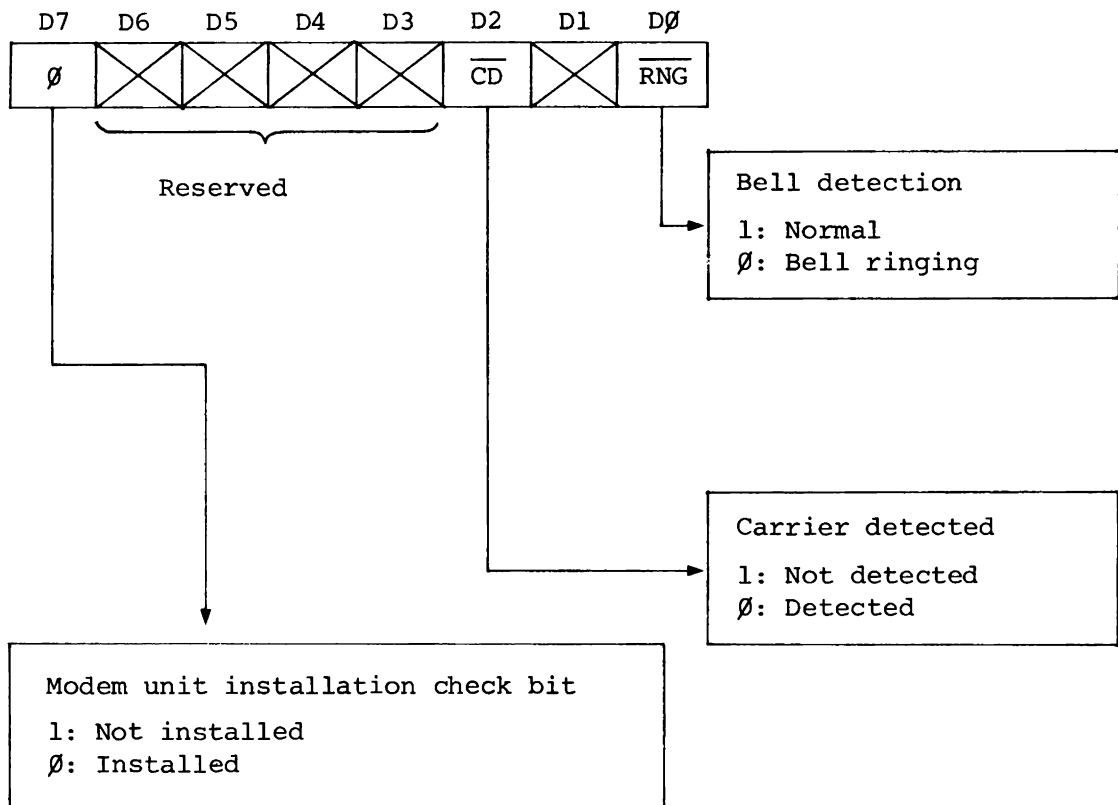
Note: All bits are Ø when modem power is off.
Read the status at least 1ØØ ms after power
is turned on.

4.5 Address 87H

I/O: Output only

Use: <Input> Inhibited

   <Output> For port (8255 mode) setting

Note: When the auto shut-off function of the MAPLE main unit is activated, power to logic electronics is turned off and all modem unit functions are stopped. Consequently, the mode setting procedure described below must also be performed when recovering the MAPLE from the shut-off state. Generally, the auto shut-off function should be disabled while the modem unit is in operation.

<Output>

Sets up the 8255 operating mode. The following data must be output to this port after the MAPLE is powered or reset:

   Data to be output: 89H

Example:

   .

   .

   LD   A, 89H

   OUT   (87H), A

   .

- 8255 setup -

     Port A = output

     Port B = output

     Port C = input

     Mode = Ø

5. Operating Specifications

5.1 Connection to a communication line

The direct modem enters the off-hook state the OHC (Off-Hook Control) bit is set to 1. At least a 2-second delay is required between the off-hook state and the time when the connection to the line is established. Set the OHC bit to 1 and wait for two or more seconds before setting the CCT bit (Coupler Cut Through) to 1.
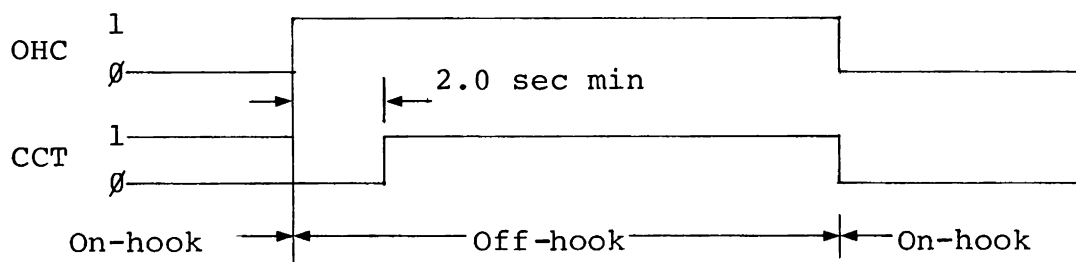
5.2 Modem operation

(i) Selecting ORIG or ANS mode

The mode is controlled by the ANS bit of the output port (85H).

$\emptyset$ = ORIG mode

1 = ANS mode

(ii) Transmission

The carrier is turned on or off in accordance with the 1/$\emptyset$ state of the TxC bit of the modem control port (85H). When the TxC bit is 1, the carrier is present all the time in the ANS mode and generated only when the counterpart carrier is being received in the ORIG mode.

5.3 Receiving a call

The direct modem unit generates outputs a *RNG signal when it detects a ringing bell. The *RNG signal can be read through the input port (86H) as a modem status flag. *RNG is a level signal and held low while the bell is ringing.

—— Timing ——

TIP/RING

RNG
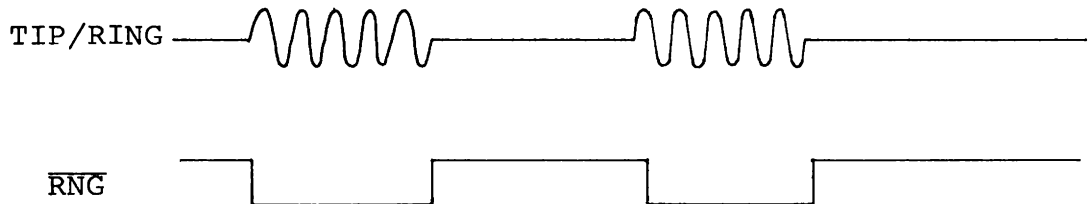
5.4 Dial operation

(a) Pulse dialing

Pulse dialing is made possible by controlling the OHC bit of the modem control port.

(1) Procedure

\<Step\>	\<Action\>
1	Set OHC to 1.
2	Wait for at least two seconds.
3	Supply dial pulses to the OHC.
4	Set CCT to 1 with at least two-second delay after the last pulse.
5	Set OHC and CCT to 0 to disconnect the line.

6   To dial again, return to step 1 after
         holding OHC low for longer than 3.5
         seconds.

(2) Dial pulses

- One pulse consists of 60 ± 3 msec on-hook and 40 ± 2 msec off-hook states.

- Correspondence between digits and number of pulses are as follows:

Digit	Number of pulses
1 - 9	1 - 9
0	10

- Digit interval

  650 msec. min. and 3 sec. max

(3) Timing

OHC

CCT
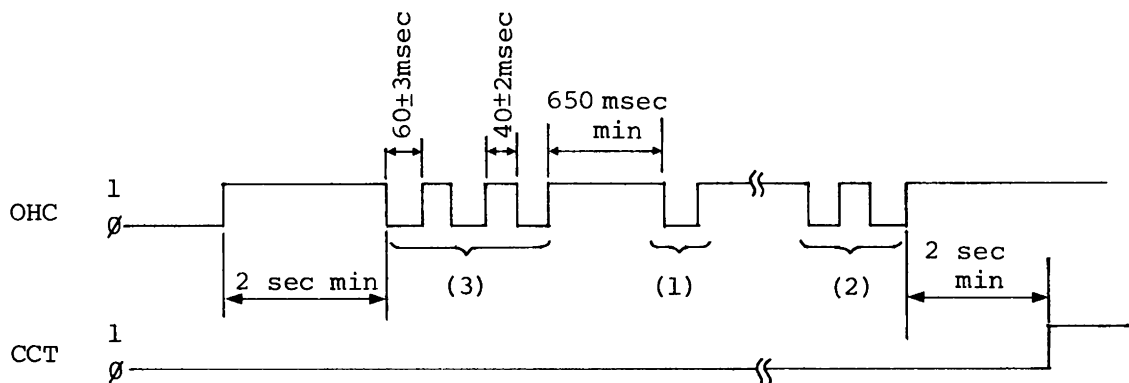
(b) Tone dialing

Dialing using the DTMF signal is enabled by controlling the output port (84H).

(1) Procedure

The same as pulse dialing except step 3. Set CCT to 1 and use the output port (84H) instead of controlling CHC.

(2) Data to the output port (84H)

The correspondence between the digits and bits F0 through F3 is listed below.

	F3	F2	F1	FØ
1	Ø	Ø	Ø	Ø
2	Ø	Ø	Ø	1
3	Ø	Ø	1	Ø
4	Ø	1	Ø	Ø
5	Ø	1	Ø	1
6	Ø	1	1	Ø
7	1	Ø	Ø	Ø
8	1	Ø	Ø	1
9	1	Ø	1	Ø
Ø	1	1	Ø	1
*	1	1	Ø	Ø
#	1	1	1	Ø
A	Ø	Ø	1	1
B	Ø	1	1	1
C	1	Ø	1	1
D	1	1	1	1

B, C, D } Usually not used.

FØ through F3 determine the tone frequency. Whether
dial tone is to be turned on or off is controlled by the
tone bit.

(3) Frequency

The table below lists the frequency of the tones
derived from the clock (3.579545 MHz). (Deviation limit
is ±1.5%.)

	Standard DTMF in Hz	Tone output in Hz	Deviation from standard (%)
f1	697	7Ø1.3	+Ø.62
f2	77Ø	771.4	+Ø.19
f3	852	857.2	+Ø.61
f4	941	935.1	-Ø.63
f5	12Ø9	1215.9	+Ø.57
f6	1336	1331.7	-Ø.32
f7	1477	1471.9	-Ø.35
f8	1633	1645.Ø	+Ø.73

(4) Level

- Nominal level per tone: -6 to -4 dBm

- Minimum level: Low frequency group: -1Ø dBm

                 High frequency group: -8 dBm

- Maximum level: Low frequency group + High frequency group = ØdBm

- Level difference between low and high frequency group
  tones:

    ØdB <= (High frequency group) - (Low frequency group) <= 4dB

(5) Timing

- Tone duration: 50 msec. min.

- Tone interval: 45 msec. min. to 3 sec. max.

- Cycle: 100 msec. min.

The procedure and timing are controlled by a system program.

(6) Correspondence between digits and output frequencies

High frequency group (Hz) / Low frequency group (Hz)	1209	1336	1477
697	1	2	3
770	4	5	6
852	7	8	9
941	*	0	#

The Multi-Unit 64 (Model H108A) contains the following units (the unit is available only in U.S.A.):

- 64K RAM disk (described in 16.1)

- Direct modem (described in 16.3)

- ROM capsule

This section describes the specifications and functions of the ROM capsule incorporated in the Multi-Unit 64.

The basic specifications for the ROM capsule are identical to those for the ROM capsule installed in the main unit. Consequently, the ROM used for the main unit may also be used on the Multi Unit 64 simply by switching the ROM select jumpers.

The major differences from the ROM capsule in the main unit are as follows:

- Either 512K-bit (64K-byte) or 1M-bit (128K-byte) ROM may be used.

- Non-CMOS type ROM cannot be used.

See Section 15.2 for the file structure in ROM.

(1) Block diagram

450mAH

V_{CH}

To MAPLE
system buffer

G.A.

DATA

64KB
DRAM
#0

128KB
(1Mb)

ROM#1

or
64KB
32KB
16KB
8KB

ADDRESS

32kHz
OSC

LATCH

82C55

MODEM

16-48
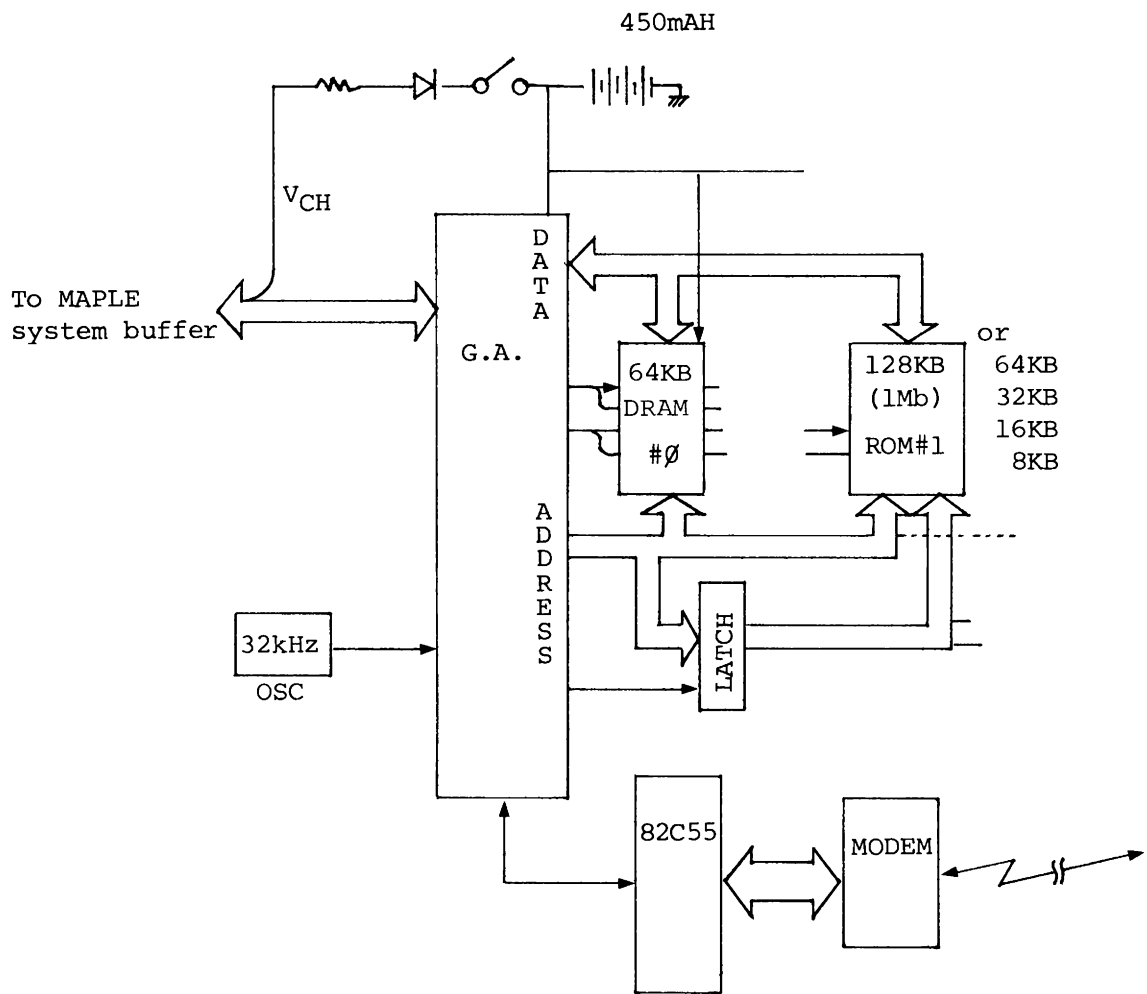
(2) Address map

The ROM capsule is allocated in RAM as shown below.
Load the starting address of ROM in the address
register.

```
          3FFFFH ┌─────────────────┐  ⎫
                 │      ROM        │  ⎬  For ROM capsule
                 │   64Kb ∿ 1Mb    │  ⎭
          2ØØØØH ├─────────────────┤
                 │/////////////////│
                 │/  Free area    /│
                 │/////////////////│
           FFFFH ├─────────────────┤  ⎫
                 │                 │  ⎬  64K RAM DISK
                 │      RAM        │  ⎭
               Ø └─────────────────┘
```
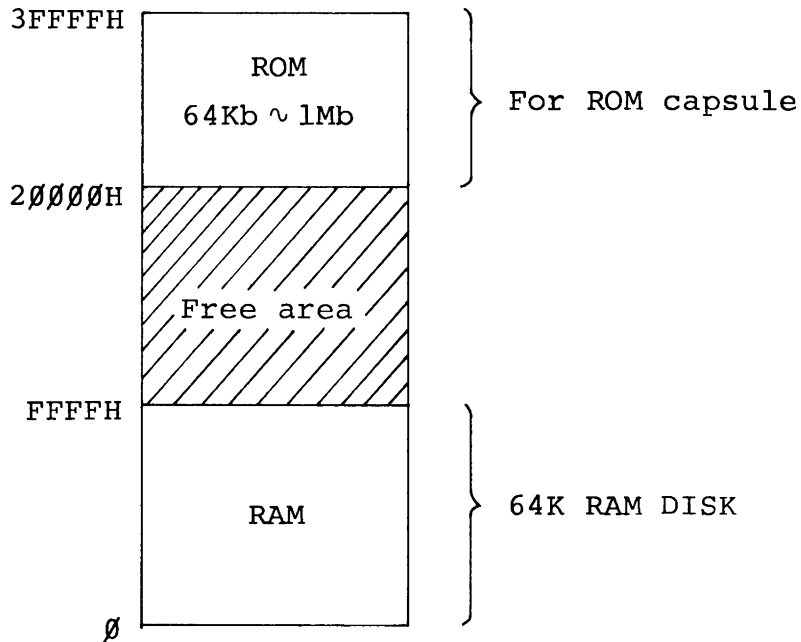
(3) ROM file

  1) The ROM file block can be accessed in the same way
  as the RAM file block. Each ROM socket has 28 pins,
  so not only 1M-byte ROM devices but also 64K-, 128K-,
  256K, and 512K-byte CMOS mask EPROM devices may be
  used.

16-49

Switch settings must be changed as follows according
to the type of ROM devices used:

ROM	SW2	SW3
64Kb	K	B
128Kb	K	B
256Kb	K	B
512Kb	1M	B
1Mb	1M *	A *

*: Settings as shipped
   from factory

Note: 512K-byte ROM is
      not released at
      present.

2) NMOS ROM cannot be used because power to ROM is
supplied from the battery through a switch.
Normal operation will not be guaranteed if NMOS is
used. NMOS ROM would consume more power and thus
decreasing life of the MAPLE battery.

3) The ROM capsule in the MAPLE main unit may
be installed on this board. To maintain
compatibility, the OS must support the ROM capsule on
this board. (Overseas OS versions B and up support
this ROM capsule.)

4) Address map

The address relationship for different ROM device types (address to be loaded in the address register) are shown below.
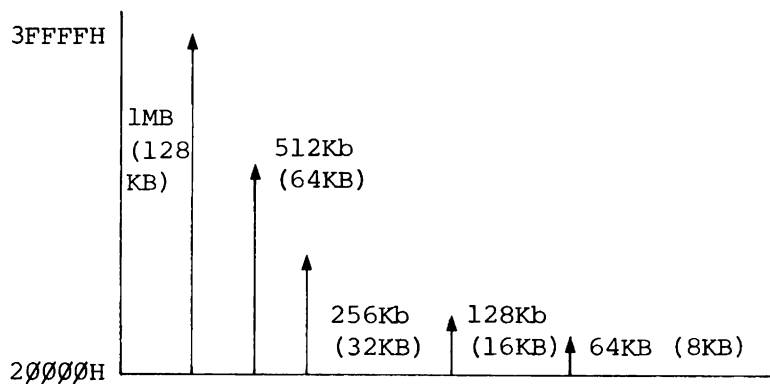
5) Writing data into ROM

Data is written into ROM in the same way as into the ROM capsule in the main unit if ROM devices smaller than 256K bytes are used.

Logical address	ROM address		
	64Kb	128Kb	256Kb
20000H (23FFFH 24000 (27FFFH	0) 1FFFH	0 (3FFFH	4000H (7FFFH 0 (3FFFH

The correspondence between the logical and physical addresses for 1M- and 512K-byte ROM is shown below.

Logical address	ROM (1M byte) address	(512Kb)
20000H	0	0
		FFFFH
3FFFFH	1FFFFH	

6) Applicable ROM devices

1M bytes	Hitachi	HN62301 or equivalent
256K bytes	SUWA SEIKO	SMM6326 or equivalent
	Fujitsu	27C256 or equivalent
128K bytes	Fujitsu	27C128 or equivalent
64K bytes	Fujitsu	27C64 or equivalent

The Multi-Unit II (Model H115A) contains the following:

- 64K RAM disk (described in 16.1)

- ROM capsule (described in 16.4)

- RS-232C interface

The Multi-Unit II contains the same components as the
Multi-Unit 64 except the RS-232C interface that is
employed instead of the direct modem. This feature
allows the Multi-Unit II to have a capability to
communicate with the MAPLE asynchronously or
synchronously. The interface on the main unit can be
used only in the asynchronous mode because of an
inadequeate number of connectors on the main unit.

This section describes the specifications for and
functions of the RS-232C interface on the Multi-Unit II.

(1) Block diagram
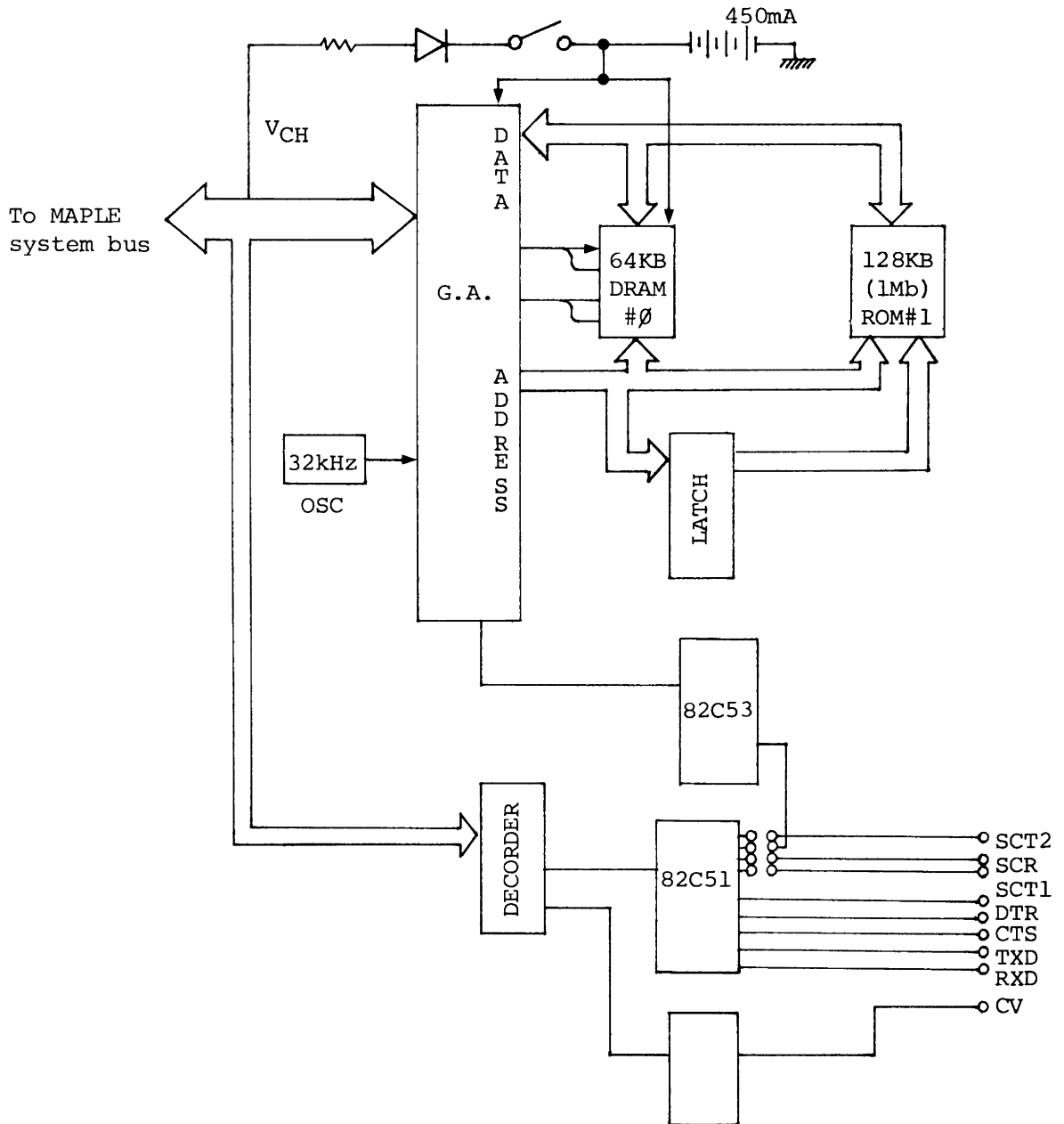
(2) RS-232C hardware

The RS-232C interface is composed of an 8251A (serial
interface) Programmable Communication Interface and a 8253
Programmable Interval Timer.

(3) RS-232C control

The MAPLE controls the RS-232C interface through the
following I/O ports:

A0H: #0 (Count register) ⎫
A1H: #1 Unused   ⎬ 82C53
A2H: #2 Unused   ⎪
A3H: Counter mode   ⎭

A4H: Data   ⎫
A5H: Control/status   ⎬ 82C51
A6H: Carrier detection signal
A7H: RS-232C power on/off

(4) Address A6H

    I/O= Input only

    Use: For carrier detection

D7	D6	D5	D4	D3	D2	D1	DØ
Ø	Ø	Ø	Ø	Ø	Ø	PWR	CD

Carrier detection signal
Ø: Received signal is in the
    specified range.
1: Received signal is beyond
    the specified range.

RS-232C installation check bit

Ø: Installed
1: Not installed

RS-232C power on/off signal

Ø: Off
1: On
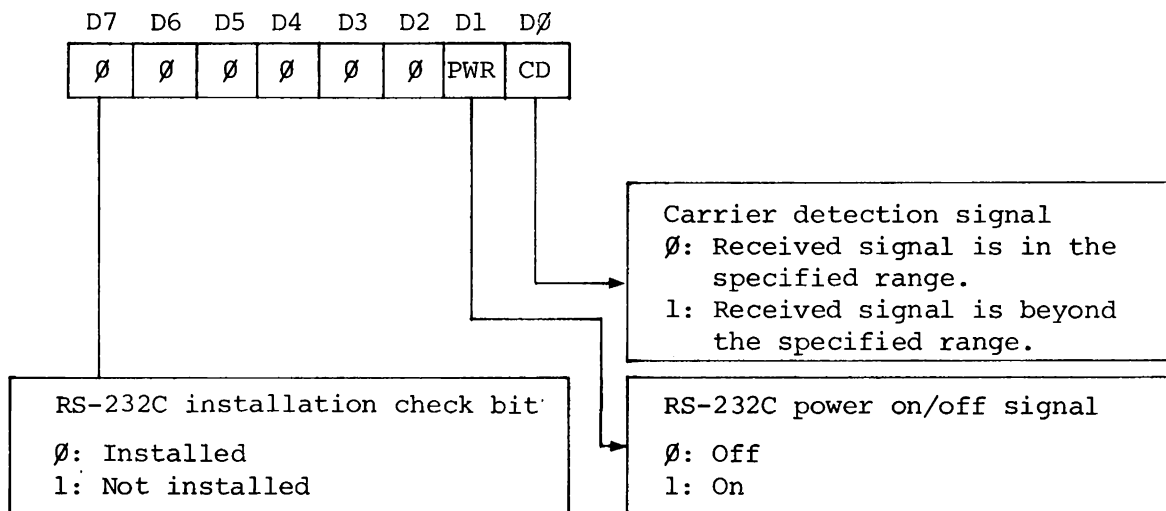
(5) Address A7H

I/O= Output only

D7	D6	D5	D4	D3	D2	D1	DØ
◻	◻	◻	◻	◻	◻	PWR	◻

Turns on or off RS-232C power.

Ø: Off
1: On
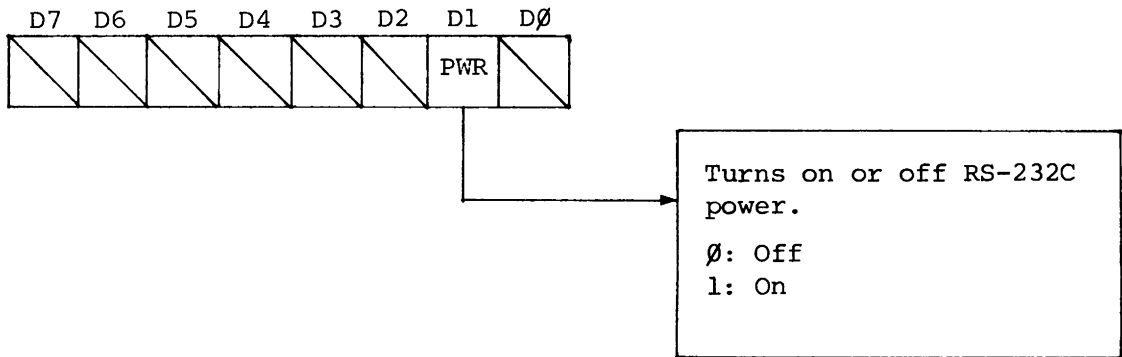
RS-232C power refers to the power to the RS-232C driver and receiver.

Note: When the auto shut-off function of the MAPLE main unit is activated, power to logic electronics is turned off and all RS-232C functions are stopped and placed into the reset state. Consequently, it is necessary to set the operating mode of the 8251 when recovering the MAPLE from the auto shut-off state.

It is necessary to supply a mode definition to the 8251 after the MAPLE is powered on or reset.

(6) 8251 serial interface

(See related manuals of 8251 for detail.)

The system clock input to the 8251 is 2.4576 MHz.

There are two 8251 control word formats:

1. Mode instruction format

2. Command instruction format

Mode instruction format

A word in this format must follow an 8251 reset
(internal or external). Once the mode instruction is
written into the 8251 from the CPU, the 8251 is ready
for receiving SYNC character(s) or a command
instruction.

Command instruction format

The first command instruction must follow a mode
instruction or SYNC characters. The command instruction
may be written at any time throughout the operation of
the 8251. The 8251 can be reset into the state in which
it is ready to receive a mode instruction by setting the
D_6 bit (IR) in the command instruction word.

* Address A5H (Mode instruction format in the asynchronous mode)

```
 D7  D6  D5  D4  D3  D2  D1  D0
┌──┬──┬──┬───┬──┬──┬──┬──┐
│S2│S1│EP│PEN│L2│L1│B2│B1│
└──┴──┴──┴───┴──┴──┴──┴──┘
```

0	1	0	1	
0	0	1	1	Baud rate
Synchronous mode	X1	X16	X64	

0	1	0	1	
0	0	1	1	Character length
5 bits	6 bits	7 bits	8 bits	

Parity enable
1: Enable
0: Disable

16-59

```
D7  D6  D5
┌───┬───┬───┐
│S2 │S1 │EP │
└─┬─┴─┬─┴─┬─┘
```

Even parity

1: Even parity
0: Odd parity

0	1	0	1	Number of
0	0	1	1	stop bits
In-valid	1 bit	1 1/2 bits	2 bits

(Synchronous mode)

```
D7   D6  D5  D4  D3  D2  D1  D0
┌───┬───┬──┬───┬──┬──┬──┬──┐
│SCS│ESD│EP│PEN│L2│L1│ 0│ 0│
└─┬─┴─┬─┴─┬┴─┬─┴─┬┴─┬┘
```

0	1	0	1	Character length
0	0	1	1
5 bits	6 bits	7 bits	8 bits

Parity enable
1: Enable
0: Disable

Even parity
1: Even
0: Odd

External sync detect
1: SYNDET = Input
0: SYNDET = Output

Single SYNC character
1: Single SYNC character
0: Double SYNC character

Switching between synchronous and asynchronous modes is controlled by setting bits D1 and D2 of the mode instruction.

Command instruction format (Address A5H)

D7	D6	D5	D4	D3	D2	D1	D0
EH	IR	RTS	ER	SBRK	RxE	DTR	TxEN

Transmit enable
1: Enable
0: Disable

Data terminal ready
1: *$\overline{DTR}$ pin = 0
0: *$\overline{DTR}$ pin = 1

Receive enable
1: Enable
0: Disable

Send break character
1: TxD pin = 0
0: Normal operation

Error reset
1: Resets error flags
0: No operation

Request to send
1: *$\overline{RTS}$ pin = 0
0: *$\overline{RTS}$ pin = 1

Internal reset
1: Reset 8251
0: No operation

Enter hunt mode
1: Starts search for
   SYNC characters
0: No operation

Status format (Address A5H)

The CPU reads the status of the 8251 by setting C/*D bit to 1 (C1H).

D7	D6	D5	D4	D3	D2	D1	D0
DSR	SYNDET/3D	FE	OE	PE	TxE	RxRDY	TxRDY

→ Note

→ Same definitions as external I/O pins

Parity error
The PE flag is set when a parity error is detected.

Overrun error
The OE flag is set when the next character is received before the 8251 completes processing the current character.

Framing error (Async only)
The FE flag is set when valid stop bit(s) are not detected at the end of a character.

Data set ready
Indicates that the *$\overline{\text{DSR}}$ is at a low level.

Baud rates

1) Asynchronous mode

Baud rate	(X1)	(X16)	(X64)
150 bps	4ØØØH	Ø4ØØH	Ø1ØØH
200	3ØØØH	Ø3ØØH	ØØØØH
300	2ØØØH	Ø2ØØH	ØØ8ØH
600	1ØØØH	Ø1ØØH	ØØ4ØH
1200	Ø8ØØH	ØØ8ØH	ØØ2ØH
2400	Ø4ØØH	ØØ4ØH	ØØ1ØH
4800	Ø2ØØH	ØØ2ØH	ØØ8ØH
9600	Ø1ØØH	ØØ1ØH	————————
19200	ØØ8ØH	ØØØ8H	————————

Load a counter value in the above table into the 8253.

2) Synchronous mode

    1200 bps   Ø8ØØH

    2400 bps   Ø4ØØH

    4800 bps   Ø2ØØH

    9600 bps   Ø1ØØH

Sample program (Synchronous mode, 4800 bps)

```
    LD   A, 36H      ; Load control word

    OUT  (A3H), A

    LD   A, ØØH      ; Load lowest 8 bits of counter value

    OUT  (AØH), A

    LD   A, Ø2H      ; Load highest 8 bits of counter value

    OUT  (AØH), A
```

(7) 8253 programmable interval timer

(See related manuals of 8253 for detail.)

Address A3H (Control word)

D7	D6	D5	D4	D3	D2	D0	D1
SC1	SC0	RL1	RL0	M2	M1	M0	BCD

BCD	
0	Binary count (16 digits)
1	BCD count (4 digits)

M2	M1	M0	Mode
0	0	0	Mode 0
0	0	1	Mode 1
X	1	0	Mode 2
X	1	1	Mode 3 *
1	0	0	Mode 4
1	0	1	Mode 5

RL1	RL0	Read/Load
0	0	Count latch operation
0	1	Read/load LSB
1	0	Read/load MSB
1	1	Read/load LSB and MSB in that order.

SC1	SC0	Select counter
0	0	Counter #0
0	1	Counter #1
1	0	Counter #2
1	1	————

* Use the 8253 in mode 3.

Program sequence for starting the 8253

```
Specify mode (control word)
            ↓
Load counter value
            ↓
```

16-64
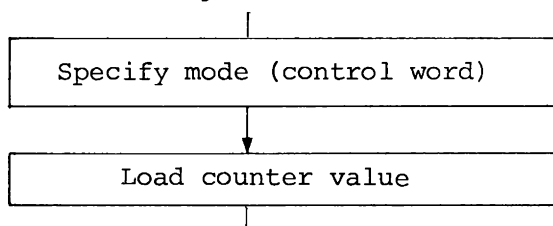
(8) Interrupts

An RS-232C interrupt is generated when either RxRDY or TxRDY of the 8251 is set to 1. Both RxRDY and TxRDY are connected to the INT5 (INTEXT) of the MAPLE main unit.

(9) Switching between RS-232C external and internal synchronous modes

DIP switches are used to switch between RS-232C internal and external synchronous modes.

Procedure for checking whether the Multi-Unit II is
installed

```
                    ( Start )
                        │
          ┌─────────────────────────────┐
          │ Read memory unit            │       I/O address: 94H
          │ installation check bit      │
          └─────────────────────────────┘
                        │
                        │              N
                  ╱─────────────╲───────────────────┐
                 ╱  Is memory unit ╲                 │
                ╲   installed ?    ╱                 │
                 ╲───────────────╱                   │
                        │ Y                          │
          ┌─────────────────────────────┐    I/O    │
          │ Read RS-232C                │    address:│
          │ installation check bit      │      A6H   │
          └─────────────────────────────┘            │
                        │                            │
                        │           N                │
                  ╱─────────────╲─────────────┐      │
                 ╱  Is RS-232C   ╲             │      │
                 ╲  installed ?  ╱             │      │
                  ╲─────────────╱              └──────┤
                        │ Y                           │
                                          ┌───────────────────────────┐
                                          │ Multi-Unit II is not      │
                                          │ installed                 │
                                          └───────────────────────────┘
          ┌─────────────────────────────┐             │
          │ Multi-Unit II is installed  │             │
          └─────────────────────────────┘             │
                        │                             ▼
                        ▼
```
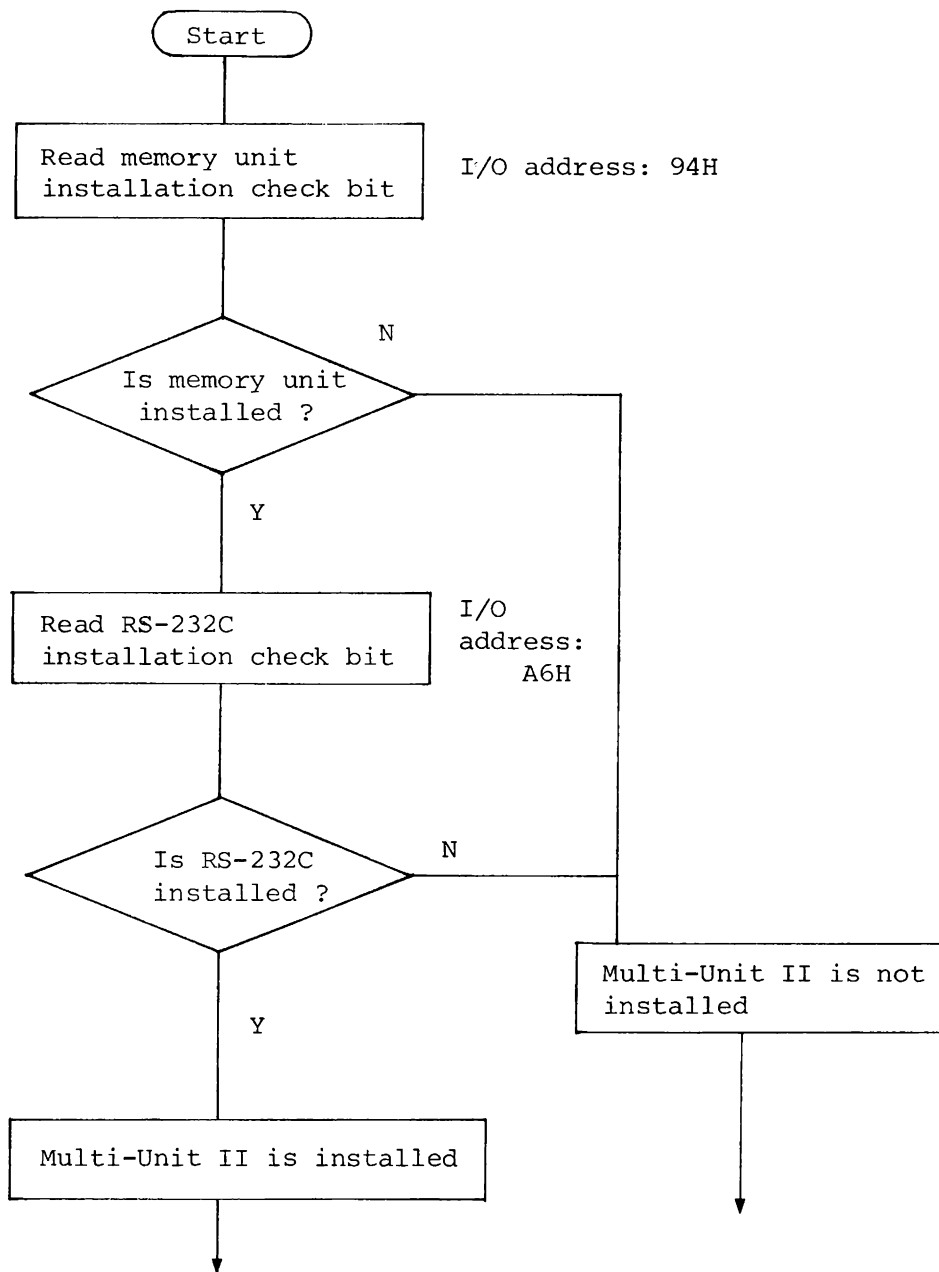
Chapter 17   How to Use User BIOS Area

17.1   Outline

A user BIOS area is reserved in MAPLE main memory to hold machine-language routines (e.g., bar code read program) or data (the scheduler stores its data here) to be shared by two or more programs.

ØØØØH

TPA	
CCP	} 8ØØH
RBDOS	} 1ØØH
RBIOS	} 1ØØH
RAM DISK	
User BIOS	Total size is 24K bytes for overseas versions and 14K bytes for Japanese-language version.
OS system area	

Top address →

Bottom address →

FFFFH
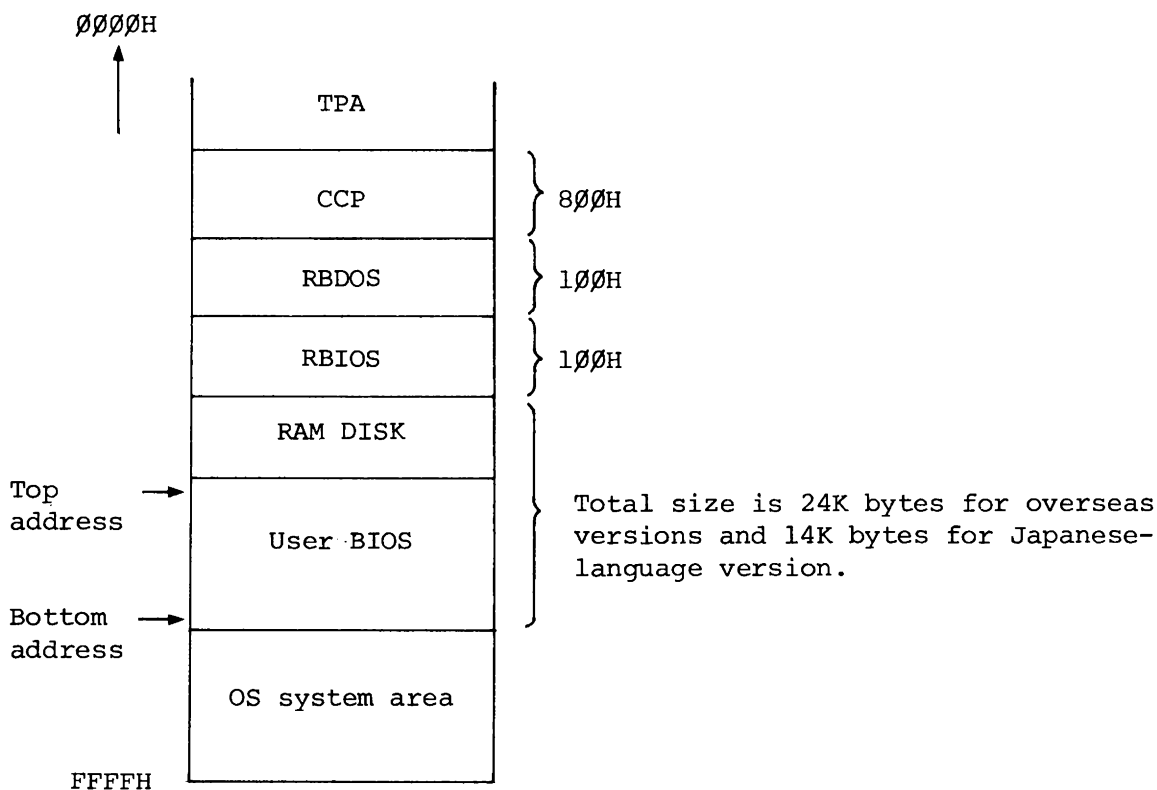
Either the BIOS or interrupt processing routines can be extended by loading their extended portions in the user BIOS area.

17.2 User BIOS Area Specifications

The specifications of the user BIOS area are described in this section.

(1) The bottom address is determined by the system.

    Overseas version (PX-8) -------------- 0EBFFH

    Japanese-language version (HC-80) ---- 0E7FFH

    Japanese-language version (HC-88) ---- 0C1FFH

(2) The size of the user BIOS area may be specified in 256-byte increments provided that the total size of this area and the RAM disk do not exceed the limit shown below.

    Overseas version (PX-8) -------------- 24K bytes

    Japanese-language version (HC-80) ---- 14K bytes

    Japanese-language version (HC-88) ---- 14K bytes

(3) The user BIOS can be specified during system initialization or by using the CONFIG program.

(4) The contents of the user BIOS area are preserved until a system initialization is performed.

(5) Generally, only one program or a block of data can occupy the user BIOS area at a time. A header must be placed in the user BIOS area to have the area shared by more than one program or block of data. Details on the header are given in Section 17.3.

(6) The sizes of the user BIOS and RAM disk are loaded in the following work areas:

- USERBIOS   Loaded with the user BIOS

$$\left(\begin{array}{l} \text{Overseas} = 0F00BH \\ \text{Japanese-language} \\ \qquad = 0ED0BH \end{array} \right)$$ area size specified in 256-byte units.

- SIZRAM   Loaded with the RAM disk

$$\left(\begin{array}{l} \text{Overseas} = 0F009H \\ \text{Japanese-language} \\ \qquad = 0ED09H \end{array} \right)$$ size specified in 1K-byte units.

(USERBIOS) + (SIZRAM) <= 24K bytes or 14K bytes

(1) Outline

As described in the previous section, the user BIOS area

may be shared by more than one program or block of data

by placing a 16-byte header at the end of the area.

The header is used by the application program to check

whether the program or data to be used is available in

the user BIOS area.

(2) Header format

①		②	③	④	⑤	⑥	⑦
"U"	"B"	Routine name	Size	Over write flag	Release adr	ØØH	Check sum

The header is always located at the following address since the bottom address of the User BIOS area is fixed (the top address differs depending on the size of the user BIOS area).

Machine	OS	Address
MAPLE	ASCII	EBFØH - EBFFH
	Kana	E7FØH - E7FFH
	Japanese-language, TXT	ClFØH - ClFFH

(2.1) Header contents

No.	Item	Size (byte)	Description
1	Header ID	2	ID for the header. Fixed to "UB".
2	Routine name	8	Name of the routine loaded in the user BIOS area. Any name may be specified in ASCII. (Specify a name which is not used for another routine.)
3	Size	1	Indicates the size of the routine loaded in the user BIOS area in 256-byte units stored in binary.
4	Overwrite flag	1	Flag for indicating whether the currently loaded routine can be overwritten. ØØH: Overwrite disabled. Others: Overwrite after release processing enabled. (See the next subsection for details.)

No.	Item	Size (byte)	Description
5	Release adr.	2	The processing routine at this address is executed before a routine currently loaded in the user BIOS area is overwritten by a new routine. This release processing routine may be executed only when the overwrite flag for the currently loaded routine is set to ØØ. The release address must fall within in the user BIOS area. The release processing routine must end with a RET instruction.
6	Not used.	1	Fixed to ØØH.
7	Checksum	1	Loaded with the result obtained by subtracting the contents of the 15 bytes (from the header top to the item preceding Checksum) from ØØH, sequentially one byte at a time. (This result is used for checking the validity of the header data.)

(3) Overwrite flag

1) 0H: For routines which inhibit loading of new
routines

Set the overwrite flag to 0H when loading a routine
which must be resident in the user BIOS area
(scheduler resident routines, for example) once it is
loaded.

This routine can be deleted from the user BIOS area
only by the program that loaded the routine.

2) Other values: For routines which allow loading of
new routines after execution of a release processing
routine

A nonzero value must be specified to allow a new
routine to be loaded into the user BIOS area when the
user BIOS area can be restored into the original state
after the execution of a release processing routine.

Set this flag to a nonzero value for user routines
(e.g., routines which change the contents of a hook or

jump vector table) which alter the system area at load
time but which can restore the system area into the
original state by executing the release processing
routine and load a new routine into the user BIOS
area.

(4) Release processing routine

A user BIOS routine which is to modify the contents of
the system area (hook or jump table, for example) must
save the original contents of the system area into the
user BIOS area before starts execution.

The release processing routine is called to restore the
system into the state before the user BIOS routine is
loaded by placing the saved contents back into the system
area and setting all header fields to 00Hs.

The header must be cleared even if the system area need
not be restored to the original state.

Note: Place the release processing routine in the
     highest 256 bytes (including the header) of the
     user BIOS area.

(5) Procedure for using the routine in the user BIOS area in the application

The application program must verify that the user BIOS routine is available before accessing that routine. The procedure shown below must be observed to check this.

```
                        ┌─────────┐
                        │  ENTER  │
                        └────┬────┘
                             │
         N             ╱ Header ╲       (A)
      ┌────────────── ╱ present? ╲
      │               ╲         ╱
      │                ╲       ╱
      │                   Y
      │                             (B)
      │              ╱ Correct user ╲    Y
      │         ┌── ╱  BIOS routine  ╲ ──────────────────────┐
      │         │   ╲       ?        ╱                        │
      │         │    ╲             ╱                          │
      │         │        N                                    │
      │         │                                             │
      │         │     ╱ Overwrite flag ╲  Y                   │
      │         │ ┌─ ╱    =00H ?       ╲ ──────┐              │
      │         │ │  ╲                 ╱        │              │
      │         │ │      N                      │              │
      │         └─┴──────┤                      │              │
      └─────────────────►│                      │              │
                  N  ╱ Routine         ╲        │              │
           ┌──────  ╱ loadable-from auxil-╲     │              │
           │        ╲ iary storage       ╱      │              │
           │         ╲      ?           ╱        │              │
           │             Y                       │              │
   ┌───────┴──────┐  ┌──────────────┐      ┌─────┴──────────┐  │
   │ Necessary user│  │ Release      │ (C)  │ User BIOS area │  │
   │ BIOS routine  │  │ processing   │      │ occupied by    │  │
   │ not found     │  └──────┬───────┘      │ another routine│  │
   └───────┬───────┘         │              └─────┬──────────┘  │
           │          ┌──────┴───────┐            │             │
      ┌────┴────┐     │ Load necessary│ (D)  ┌────┴────┐        │
      │  ERROR  │     │ user BIOS     │      │  ERROR  │        │
      └─────────┘     │ routine       │      └─────────┘        │
                      └──────┬───────┘                          │
                             │◄─────────────────────────────────┘
                        ┌────┴────┐
                        │  EXIT   │
                        └─────────┘
```
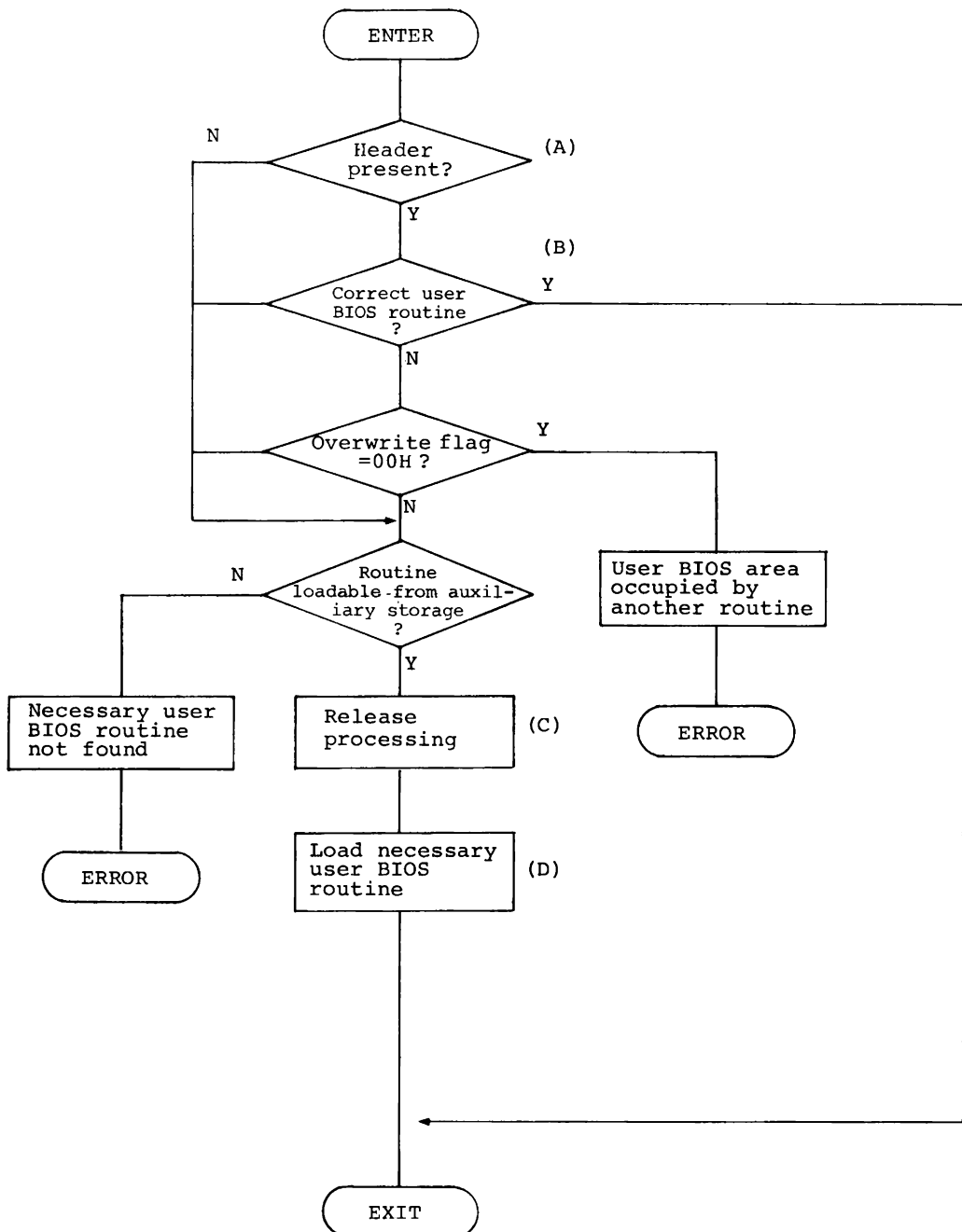
(A) Check whether the correct header is present by matching the header ID with "UB" and checksum. If the header ID field contains "US", it is unconditionally concluded that the scheduler is using the user BIOS area because the MicroPro scheduler for MAPLE defines "US" as the header ID.

(B) Check to determine whether the required user BIOS routine is loaded in the user BIOS area by checking the routine name in the header.

(C) Call the routine addressed by the release address in the header.

(D) Load a new routine to the user BIOS area and update the header contents.

Chapter 18 Memory Maps

This chapter gives memory maps for the following:

- OS ROM

- RAM memory

- Jump vectors at the beginning of OS ROM

18.1 OS ROM Memory Map

The OS ROM consists of 32K bytes of mask ROM devices
that contain the system modules described in 2.3.2. Its
memory map is given on the next page. The addresses of
the system modules are not shown in the map partly
because they differs for the overseas and domestic
versions and partly because the system modules are
normally not accessible to application programs (except
via the jump vectors described in 18.3).

ØØØØH

STARTER
INTROM
MENU
SYSCRN
RELOC
BDOS
PREBIOS
PSTBIOS
BIOS1
BIOS2
BIOS3
SCREEN
MTOS
MIOS
CCPD
RBDOSD
RSYSPR
SYSAR1
SYSAR2
SYSAR3
ROMID

7FFFH

See 2.3.2 for the function
of the individual modules.

18.2 RAM Memory Map

The memory map of the 64K-byte MAPLE RAM memory is almost identical to that of the basic CP/M system. The OS system areas are fully described at the end of this chapter.

RAM Memory Map

Overseas version

Japanese kana version

Address	Area
ØØØØH	CP/M system area
Ø1ØØH	TPA

8ØØØH and up ⟶

CCP — 8ØØH

RBDOS — 1ØØH   BDOS entry

RBIOS — 1ØØH   BIOS entry

RAM DISK — 1K-byte increments

User BIOS — 256-byte increments

Total Overseas version = 24K bytes
Japanese-language version = 14K bytes

Overseas version = ØECØØH
Japanese-language version = ØE8ØØH

This address is fixed.

OS system area

FFFFH

The size of these two areas is variable provided it does not exceed the total.

OS System Area Memory Map

Address	Block	Description
EC00H	RSYSPR	Used during BDOS/BIOS residence processing
F000H	SYSAR1	Work area initialized by System Initialize
F0B0H	SYSAR2	Work area initialized by Reset processing
F330H	SYSAR3	Work area initialized by WBOOT
F380H	SYSAR4	Work area
F800H		
F806H	KEYBUF	Keyboard buffer
F828H	TIMBUF	Timer/clock buffer
F833H	SYSFCB	System FCB
F857H	SYSDMA	System DMA
F8D7H	MCTID	MCT ID area
F90CH	MCTDIR	MCT directory area
FA8CH	TOSBUF	MTOS buffer
FB90H	COMBUF	RS-232C buffer
FD16H	MCTRSV	MCT work area
FD9FH	ALV8 *1	
FD90H	BIOTBL	
FE1CH		
FE30H	STRTSK	STARTER stack area
FE70H	INTSK	Interrupt handler stack area
FF50H	BIOSSK	BIOS stack area
FF90H	BDOSSK	BDOS stack area
FFFFH		

*1:

ALV8 and BIOTBL in the MCTRSV shown in the map on the
preceding page are available in overseas versions B and
later. The use of these two fields are given below.
ALV8: Allocation work area defined in the disk parameter
block for drive I: (expansion unit ROM capsule).
BIOTBL: Contains jump vectors through which control from
BIOS calls (invoked by OS or application programs) is
transferred to the pertinent BIOS processing routines.
The user can alter the operation of BIOS processing
routines by changing the jump vector values. Since,
however, RBIOS jump vectors are used only during BIOS
calls invoked by application programs, BIOS calls
invoked by OS itself would transfer to the original BIOS
processing routines even if this area was altered to
direct control to user-specified BIOS processing
routines.

Chapter 19　Application Notes

This chapter gives various information which will aid the user in developing application programs.

Interface

19.12 Procedure for Sending and Detecting the RS-232C

Break Signal

19.1 FILINK Communications Protocol

FILINK transmits and receives files via the RS-232C interface using the protocol illustrated below.

```
┌─────────────┐
│ Establish   │
│ a link      │
└──────┬──────┘
       │
       ▼
┌─────────────┐ ┐
│ Send/receive│ │
│ file name   │ │    Repeat these steps when sending
└──────┬──────┘ │    or receiving more than one file
       │        │
       ▼        │
┌─────────────┐ │
│ Send/receive│ │
│ data        │ │
└─────────────┘ ┘
```
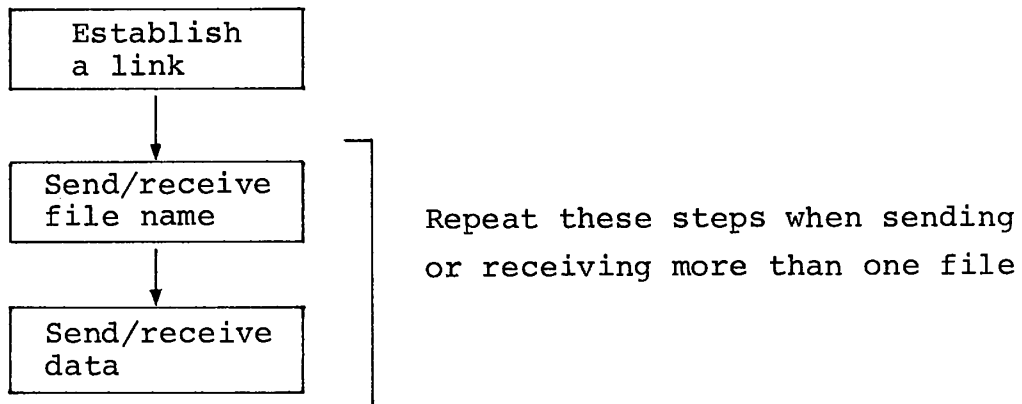
The above communications protocol is supported by the programs/commands listed below.

Machine Type	Program/Command
MAPLE	∘ FILINK.COM ∘ WS.COM T and C commands ∘ SC.COM/JSC.COM/Join, Send, and Receive commands
PINE (HC-40, PX-4 HX-40)	∘ FILINK.COM
QC - 10 QX - 10	∘ FILINK.COM
IBM-PC	∘ FILINK.COM (EXE)

The FILINK communications protocol is detailed below.

```
                        ┌─────────────┐
                        │  Establish  │
                        │   a Link    │
                        └─────────────┘
         Sender                              Receiver

(1)-Send "R".          ─────────────────►  -Send "S" if "R" is
                                              received.

            ▲                               -Send received data if data
            │                                 other than "R" is received
            │ (≠"S")
            │                  ↙
                        (="S")
(2)-Send "G" if "S" is ─────────────────►  -Proceed to "receive File
    received.                                 Name" if "G" is received.
    Proceed to "Send
    File Name."                             -Wait if other than "G"
   -Go to (1) if other                        is received.
    than "S" is received.
```

```
                        ┌─────────────┐
                        │Send/Receive │
                        │ File Name   │
                        └─────────────┘
```

           Sender   Receiver

(3)- Send 04H.   ─────────────→   -Send 08H if 04H is
        ↑   received.
        │   -Terminate FILINK if 13H
        │   (≠08H)   is received.
        │   -Otherwise, send "X" and
        │   wait (also display data).
        │   ↑
(4)-Go to (5) if 08H is   │
     received.   │   (≦1FH)
   -Go to (3) if other   │
    than 08H is received.   │

(5)-Send file name and   ─────────────→   -Send received character.
     extension, one   -If received character is
     character at a time.   1FH or smaller, send "X"
                                                       and return to step (3)
                                                       for receiver.

(6)-Compare received
     character and send
     character transmitted
     in (5).
     If match→Go to (5) to send next
              character.
     If no match→Display "?" and go
                 to (3).

              ┌─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ┐
              │ Repeat steps (5) and (6) │
              │ 11 times to send/receive │
              │ file name and extension. │
              └─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─┘

(7)-Send 05H.   ─────────────→   -Send 09H and proceed to
                                                      "Send/Receive Data" if
                                                      05H is received.
                                                    -Send "X" and return to
                                                      step (3) for receiver if
(8)-Proceed to "Send/Receive   other than 05H is
    Data" if 09H is received.   received.
   -Go to (3) if other than
    09H is received.
```

```
                      ┌─────────────────┐
                      ║ Send/Receive    ║
                      ║     Data        ║
                      └─────────────────┘
         Sender                    │         Receiver

  (9) -Read 1 record (128   ──────────→ -Send "P" and go to (11)
       bytes).                │          if 02H is received.
      -If not EOF, send 02H.  │         -Go to step (3) for
      -If EOF, send 03H.      │          receiver if 03H is
       If all files have been │          received.
       sent, send 13H to      │         -Otherwise, send "N" and
       terminate FILINK.      │          wait.
       Return to (3) if there │
       is a file to be sent.  │

 (10) -Go to (11) if "P"
       is received.
      -Otherwise, go to
       (9).


 (11) -Send 1 record (128   ──────────→ -Receive 1 record (128
       bytes) and check                  bytes) and check byte.
       byte.


 (12)                                   -Compare check bytes.
                                         If match→Send "G", writes
                                         received record into a
                                         file, and return to (9).
                                        -If no match→Send "B" and
                                         return to (9).



 (13) -Read next record and
       return to (9) if "G" is
       received.
      -Return to (9) for a retry
       if "B" is received.
      -Otherwise, wait here.
```

19.2 Procedure for Calling BDOS and BIOS Directly from BASIC

To call a BDOS or BIOS function directly from BASIC, prepare a machine-language program for interfacing to BDOS or BIOS and run it using the BASIC CALL statement.

19.2.1 Calling BDOS

(Machine-language program)

```
C5, 4E, E3, 5E, 23, 56, CD, 05,        PUSH    BC
                                        LD      C,(HL)
00, C1, 02, 03, AF, 02, C9    (15 bytes) EX     DE,HL
                                        LD      E,(HL)
                                        INC     HL
                                        LD      D,(HL)
                                        CALL    5
                                        POP     BC
                                        LD      (BC),A
                                        INC     BC
                                        XOR     A
                                        LD      (BC),A
                                        RET
```

(BASIC)

     CALL BDOS(C%, DE%, A%)

BDOS specifies the address of the machine-language routine.

C% specifies the BDOS function number (C% = 255 for dirinit).

DE% specifies the RCB address (optional).

A% holds the return code returned by BDOS (0 = normal termination; nonzero = error).

(Example)

```
100 CLEAR ,&HA000: BDOS=&HA000
110 FOR I=0 TO 14: READ X: POKE BDOS+I,X
: NEXT
120 C%=255: CALL BDOS(C%,DE%,A%)
130 IF A%<>0 THEN PRINT"Error"
140 DATA &Hc5,&H4e,&Heb,&H5e,&H23,&H56,&
Hcd,&H05
150 DATA &H00,&Hc1,&H02,&H03,&Haf,&H02,&
Hc9
```

HL → [|] c%

DE → [|] de%

BC → [|] a%

19.2.2 Calling BIOS

Change 05H, 00H (BDOS entry address) of the byte
sequence CDH, 05H, 00H in the above machine-language
program to the required BIOS entry address. To locate
the BIOS entry address, read the WBOOT entry address
stored in RAM addresses 1 and 2 with the PEEK statement
and add to it the offset value of the required BIOS
function (see Chapter 4 for details about the procedure
for calculating a BIOS entry address).

19.3 Procedure for Determining the Type and Size of RAM
    Disk

The MAPLE has the following three types of RAM disks:

    - Main memory RAM disk

    - Intelligent type RAM disk (in expansion unit)

    - Unintelligent RAM disk (in expansion unit)

RAM work area YSIZERAM (at location ØF6A8H for overseas
version and ØF42BH for Japanese-language version)
contains the size in binary form of the active RAM disk in
1K-byte increments. The value of this area also indicates
the type of RAM disk.

YSIZERAM   RAM disk type

contents

    60   60K-byte intelligent RAM disk unit

    120   120K-byte intelligent RAM disk unit

    64   64K-byte unintelligent RAM disk unit

    128   128K-byte unintelligent RAM disk unit

Other value   Indicates the size of the main memory RAM

                disk.

See Chapter 16 for the formats of the intelligent and unintelligent RAM disk units. The rest of this section describes the format of the main memory RAM disk.

```
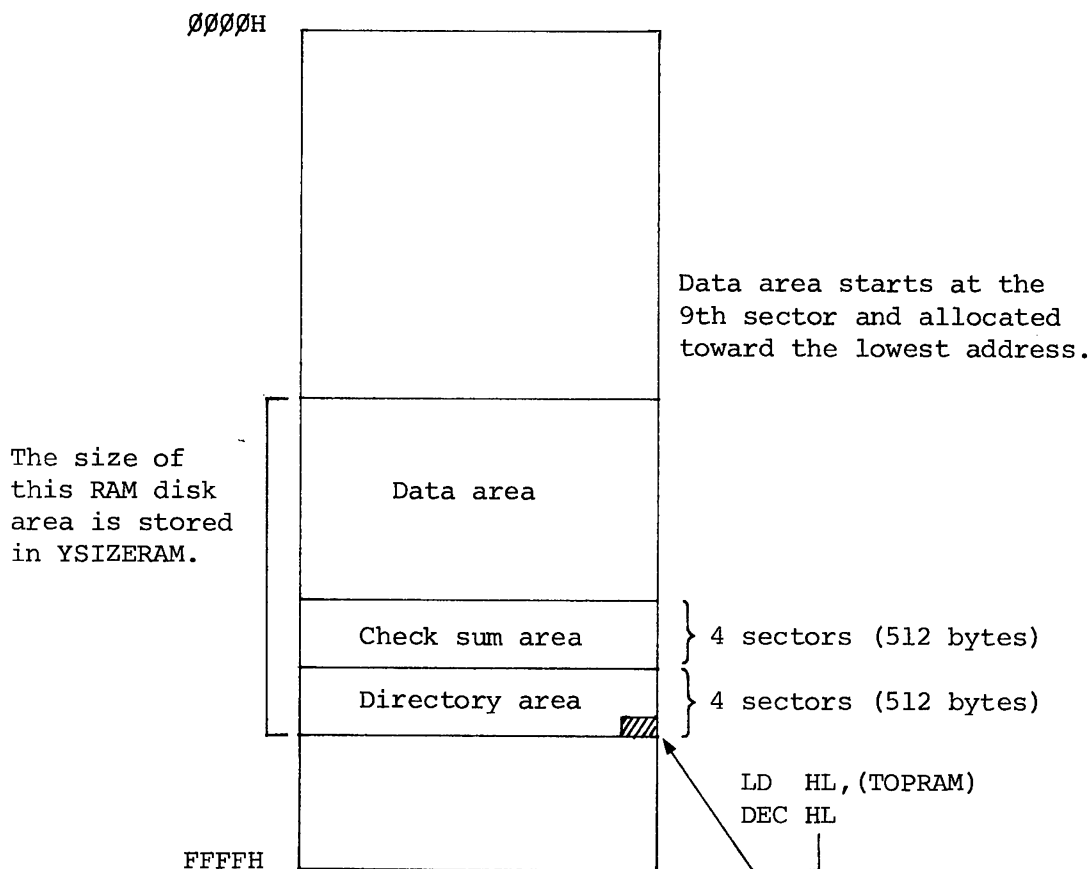         ØØØØH    ┌─────────────┐
                  │             │
                  │             │
                  │             │     Data area starts at the
                  │             │     9th sector and allocated
                  │             │     toward the lowest address.
                  ├─────────────┤
The size of       │             │
this RAM disk     │  Data area  │
area is stored    │             │
in YSIZERAM.      │             │
                  ├─────────────┤  ┐
                  │Check sum area│ ├ 4 sectors (512 bytes)
                  ├─────────────┤  ┘
                  │Directory area│////├ 4 sectors (512 bytes)
                  ├─────────────┤◄─
                  │             │      LD  HL,(TOPRAM)
                  │             │      DEC HL
         FFFFH    └─────────────┘
```

The lowest address of the RAM disk (marked ///) is calculated as the value of the 2-byte field labeled TOP RAM minus 1. The location of TOPRAM is 0F076H for overseas versions and ØED82H for Japanese-language version.

19.4 CG Fonts

The fonts used for the MAPLE are contained in SED1320 on the LCD controller (except those for Japanese-language kanji, hiragana, and katakana characters). During an OS operation, a character code issued through the BIOS CONOUT routine is converted into the corresponding code in the CG. The converted code is sent to the SED1320 via the slave CPU and used to select the corresponding font, which is then displayed on the LCD screen. A table of CG fonts are given on the next page.

The fonts for the gaiji characters corresponding to character codes 0E0H through 0FFH are stored in the slave CPU. These character codes are converted into fonts within the slave CPU and sent to the SED1320 for display.

The application program can display these CG fonts directly by sending the ESC, "%", (CG code) sequence with the CONOUT BIOS function (see Chapter 6).

To get a CG font pattern from an application program, use the slave CPU command code 1BH (see Chapter 13).

High order byte

Low order byte

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	ﾖ	ﾚ	SP	0	@	P	`	p	┼	ﾛ	SP	─	ﾀ	ﾐ		
1	ﾟ	ø	!	1	A	Q	a	q	┴	♣	｡	ｱ	ﾁ	ﾑ		
2	ﾖ	Å	"	2	B	R	b	r	┬	♥	｢	ｲ	ﾂ	ﾒ		
3	ﾖ	✖	#	3	C	S	c	s	┤	♦	｣	ｳ	ﾃ	ﾓ		
4	ﾖ	✿	$	4	D	T	d	t	├	♠	､	ｴ	ﾄ	ﾔ		
5	ﾐ	ﾖ	%	5	E	U	e	u	─	♪	･	ｵ	ﾅ	ﾕ		
6	ﾖ	É	&	6	F	V	f	v	│	ﾖ	ﾖ	ｶ	ﾆ	ﾖ		
7	‥	ﾖ	'	7	G	W	g	w	┌	♣	ｱ	ｷ	ﾇ	ﾗ		
8	Ä	¤	(	8	H	X	h	x	┐	ﾖ	ｲ	ｸ	ﾈ	ﾘ		
9	Ö	ﾖ	)	9	I	Y	i	y	┘	ﾖ	ｳ	ｹ	ﾉ	ﾙ		
A	Ü	ì	*	:	J	Z	j	z	ﾖ	ﾖ	ｴ	ｺ	ﾊ	ﾚ		
B	ﾖ	ﾖ	+	;	K	[	k	{	▨	↑	ｵ	ｻ	ﾋ	ﾛ		
C	ö	¡	,	<	L	¥	l			▦	↓	ｬ	ｼ	ﾌ	ﾜ		
D	ü	ñ	─	=	M	]	m	}	▬	×	ｭ	ｽ	ﾍ	ﾝ		
E	ß	¿	.	>	N	^	n	~	▮	÷	ｮ	ｾ	ﾎ	ﾞ		
F	£	ñ	/	?	O	_	o	△	✦	±	ｯ	ｿ	ﾏ	ﾟ		

19.5 Procedure for Identifying the OS Version from an Application Program

MAPLE/PINE (HC-40, PX-4, HX-40) application programs can prevent themselves from causing fatal errors or hanging up when executed under an unintended operating system by checking the version of the running operating system at the beginning of their execution.

To check the OS version, refer to the 6-byte CP/M serial number filed in BDOS that contains the machine type, model (country) name, and version number.

BDOS entry address stored in memory addresses 6 and 7

	-6	-5	-4	-3	-2	-1			
CP/M Serial No.	DCH	16H	Ø3H	ØØH	ØØH	Ø1H			

Fixed

○ Machine type

 ØØH --- MAPLE
 Ø1H --- PINE

○ The higher 4 bits identify the model (country).

0 --- Overseas
1 --- HASCI
2 --- ASCII
3 --- JIS
4 --- TxT

○ The lower 4 bits indicate the version number 1 to F.

How to refer to the CP/M serial number field

(1) Application programs not dedicated to the MAPLE/PINE
need not check the version number. Programs that fall
within this category include:

- Programs that reference no system area.

- Programs whose screen handling routines are not
intended for MAPLE/PINE.

- Other programs

(2) Since the MAPLE has different memory maps for its
overseas and domestic versions, application programs
for the MAPLE are also divided into two groups. These
programs must use different procedures for identifying
the OS version.

1) Application programs for overseas versions

     DCH, 16H, Ø3H, ØØH, ØØH, ØXH   (X = 1-F)

     DCH, 16H, Ø3H, ØØH, ØØH, 1XH   (X = 1-F)

     DCH, 16H, Ø3H, ØØH, ØØH, 2XH   (X = 1-F)

The application programs can run under operating systems
that contain one of the OS version field values given
above. They should signal an error condition for other
values.

2) Application programs for Japanese-language versions,

    DCH, 16H, 03H, 00H, 00H, 3XH --> JIS OS

    DCH, 16H, 03H, 00H, 00H, 4XH --> TXT OS   (X = 1-F)

The application programs can run under operating systems
that contain one of the OS version field values given
above. They should signal an error condition for other
values.

19.6 Procedure for Checking the Data Received by CCP from an Application Program

When CCP starts an application program, it loads the command line parameters in CP/M system areas at Ø5CH and 8ØH. CCP, however, deletes the name of the application program and the drive number of the disk drive from which the application program is started. Consequently, the application program cannot determine from which drive it was loaded into memory. The application program can, however, refer to the parameter data that the user specified via CCP by examining the CCP work area.

Example: A>B:BASIC TEST.BAS/

```
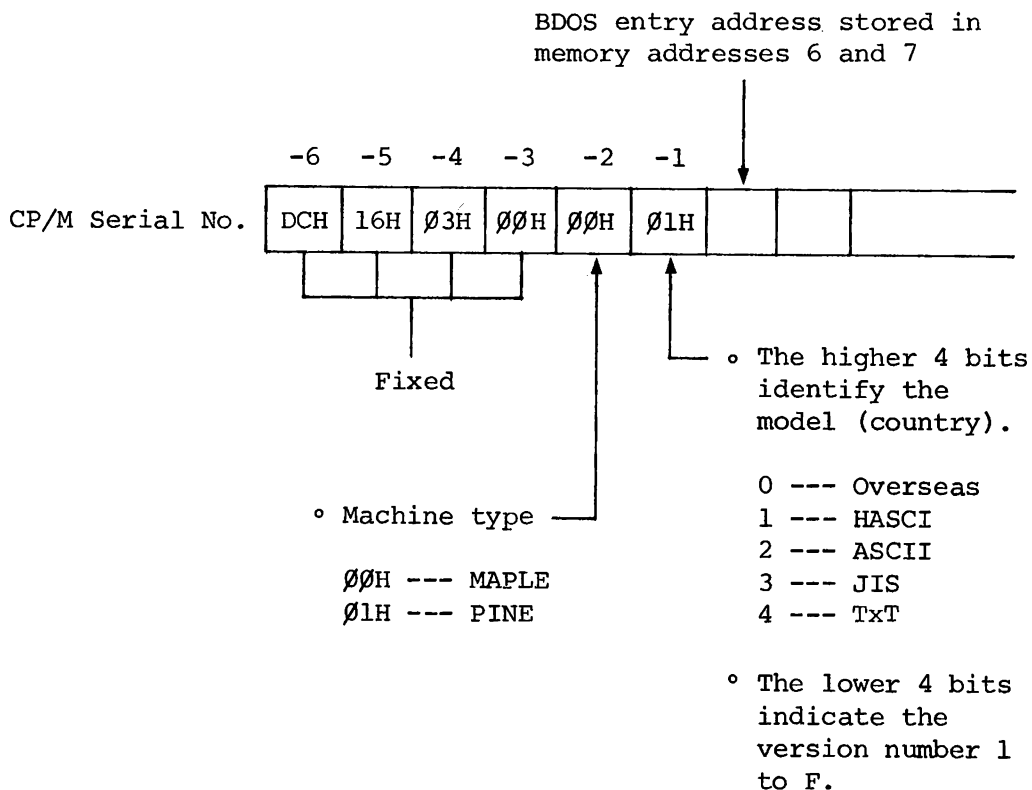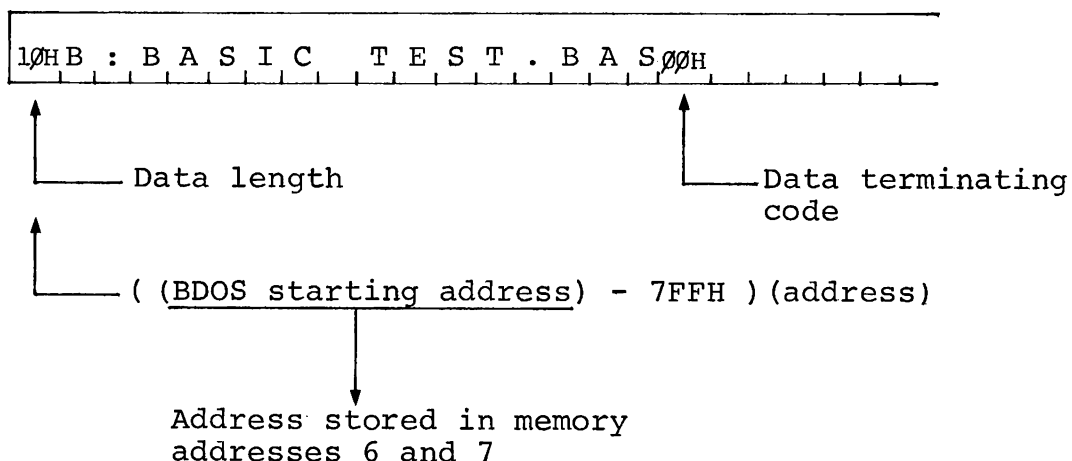┌─────────────────────────────────────────────────────┐
│ 1ØH B : B A S I C   T E S T . B A S ØØH              │
└┴──┴──┴──┴──┴──┴──┴──┴──┴──┴──┴──┴──┴──┴──┴──┴──┴──┴──┘
```

↑ ———— Data length   ↑ ——Data terminating code

↑ ———— ((BDOS starting address) - 7FFH)(address)

Address stored in memory addresses 6 and 7

19.7 Procedure for Detecting the Depression of the CTRL/STOP keys

When the STOP key is pressed, the OS usually clears the keyboard buffer and places a 03H (STOP key code) in the keyboard buffer. When the STOP key is pressed simultaneously with the CTRL key, however, the OS terminates the currently executing I/O operation as well as it clears the keyboard buffer and places a 03H code in the keyboard buffer. This allows the user to gain control when the program is placed in a loop waiting, for example, for receive data in the RS-232C receive routine. In this case, however, the application program must know that the CTRL and STOP keys have been pressed to terminate itself. This can be accomplished by examining the flag field described below.

> CSTOPFLG (ØF1ØBH for overseas version; ØEE25H for
> Japanese-language version)
>> = ØØH: The CTRL and STOP keys are not pressed.
>> ≠ ØØH: The CTRL and STOP keys have been pressed.

After the CTRL and STOP keys are pressed simultaneously, the keyboard buffer contains only 03H code and CSTOFLG

is set to a nonzero value. CSTOPFLG is cleared by the
OS when the keyboard buffer is emptied.

19.8 Procedure for Assigning Printer Output to RS-232C or Serial Interface

Output to the printer can be directed to either RS-232C or serial port by changing the contents of the I/O byte (at memory address 3).

I/O byte, bits 7 and 6 ... 1, Ø --> RS-232C

I/O byte, bits 7 and 6 ... Ø, Ø --> Serial

Output data will be placed on the specified port when the following list-related commands or routines are executed after the I/O byte is altered as shown above:

(1)   BIOS level

LIST routine (WBOOT + ØCH)

(2)   BDOS level

Function 5 (list)

(3)   BASIC level

LPRINT command

19.9 Procedure for Restoring the Screen into the State Set up by CONFIG

The screen remains in the state set up by an application program when WBOOT is performed at the end of the application program whereas the screen is restored into the state that is defined by CONFIG when BOOT is invoked by pressing the RESET switch. Any application program which reconfigures the screen should restore the screen into the original configuration at the end of its execution.

The OS stores in RAM memory the ESC sequence data related to the screen state defined by CONFIG. The application program can restore the screen configuration into the original state by sending this data to the screen using CONOUT.

The ESC sequence data is stored in the two areas given below. The ESC sequence data in each area is terminated by ØFFH code, so the application program need only send the data bytes until an ØFFH is encountered.

CONSCRN1 (ØFØDDH for overseas version; ØEDBDH for

Japanese-language version): Contains the current screen
mode and virtual screen identification.

CONSCRN2 (ØFØF1H for overseas version; ØEDD1H for
Japanese-language version): Contains data related to the
cursor.

[Sample program]

```
        DI
        LD   HL,CONSCRN1    ;screen mode, select screen
        CALL LCDOUT         ;scroll mode, function key display
        LD   HL,conscrn2    ;cursor kind and on/off
        CALL LCDOUT
        EI
        .
        .
LCDOUT:
        LD   A,(HL)
        INC  HL
        INC  A              ;end of string?
        RET  Z              ;then return
        DEC  A
        LD   C,A
        PUSH HL
        CALL CONOUT         ;display character
        POP  HL
        JR   LCDOUT
```

19.10 Procedure for Configuring the System Environment from an Application Program

19.10.1 Auto Power Off (common to both overseas and Japanese-language versions)

See 7.6 "Auto Power Off Feature."

19.10.2 CP/M Function Key (common to both overseas and Japanese-language versions)

See paragraph (6) "Programmable Function keys" in 5.6 "Special Keys."

19.10.3 Cursor & Function Key Display (common to both overseas and Japanese-language versions)

(1) Use the CONOUT ESC sequence function to set up the cursor and function key display modes.
- Cursor tracking: ESC, 95H
- Cursor display: ESC, "2" or ESC, "3"
- Cursor type: ESC, D6H
- Function key display: ESC, D3H

(2) The current settings can be located by checking the following work areas:

- Cursor tracking:

  LSMODE (ØF2D4H for overseas version; ØFØØ4H for

  Japanese-language version)

  = ØØH: Tracking mode

  ≠ ØØH: Nontracking mode

- Cursor display:

  LUSSTS (ØF2D7H for overseas version; ØFØØ7H for

  Japanese-language version)

  Bit Ø = Ø: Off

  Bit Ø = 1:   On

- Cursor type:

  LUSSTS (ØF2D7H for overseas version; ØFØØ7H for

  Japanese-language version)

  Bit 1 = Ø: Nonblink

  Bit 1 = 1:   Blink

  Bit 2 = Ø: Underline

  Bit 2 = 1:   Block

- Function key display

  LUFKDSP (ØF2DØH for overseas version; ØFØØ2H for

  Japanese-language version)

  = 7:   Display

  = 8:   No display

The screen may not actually change its configuration

when the above work areas are simply changed. This is

attributed to reasons associated with the interactions

between the screen and other resources. Nevertheless,
the screen configuration must be set up using the CONOUT
ESC sequence functions.

19.1Ø.4 Date and Time (common to both overseas and
Japanese-language versions)

Use the TIMDAT BIOS function (WBOOT + 4BH) (see Chapter
4).

19.1Ø.5 Disk Drives (common to both overseas and
Japanese-language versions)

Refer to or change the table in RAM associating the
physical and logical drives.

        Physical drive codes
            ØØH: RAM disk
            Ø1H: ROM capsule 1
            Ø2H: ROM capsule 2
            Ø3H: FDD1
            Ø4H: FDD2
            Ø5H: FDD3
            Ø6H: FDD4

The logical drives are indicated in the table shown
below.

```
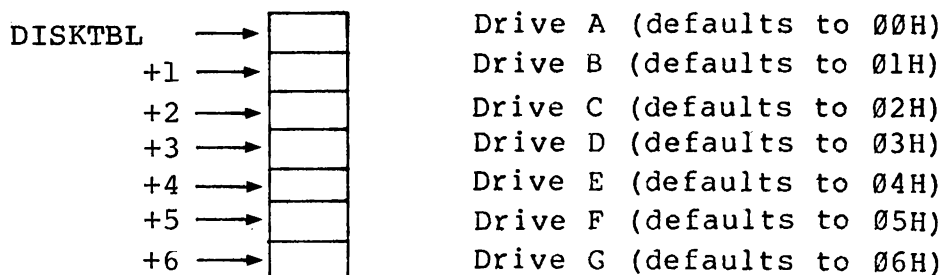DISKTBL    ⟶  [   ]     Drive A (defaults to 00H)
       +1  ⟶  [   ]     Drive B (defaults to 01H)
       +2  ⟶  [   ]     Drive C (defaults to 02H)
       +3  ⟶  [   ]     Drive D (defaults to 03H)
       +4  ⟶  [   ]     Drive E (defaults to 04H)
       +5  ⟶  [   ]     Drive F (defaults to 05H)
       +6  ⟶  [   ]     Drive G (defaults to 06H)
```

DISKTBL: (0F1D2H for overseas version; 0EEEBH for

   Japanese-language version)

Change the contents of the DISKTBL to redefine the

association between the physical and logical drives

(e.g., reassigning an FDD to drive A).

Notes:

1. Do not specify values other than 00H through 06H as physical drive codes.

2. Do not specify a physical drive code in duplicate.

3. Specify logical drives between A: and G:

4. The redefined specifications remain valid until the RESET switch is pressed.

19.10.6 Printer (common to both overseas and Japanese-language versions)

Use the I/O byte (see 19.8).

19.10.7 RS-232C (RS-232C (1) for Japanese-language version)

The 5-byte field starting at RDSDAT (0F00FH for overseas version and 0ED0FH for Japanese-language version) contains the parameter values from bit rate through special parameter that are set up by the BIOS RSIOX OPEN function. See Chapter 4.

All conditions pertaining to the RS-232C interface (for BIOS RSOPEN, RSOUT, LIST, PUNCH, etc.) are controlled by the data in this field except when using the RS-232C interface after opening it with the RSIOX OPEN function.

19.10.8 Screen mode (common to both overseas and Japanese-language versions)

(1) Send the ESC 0D0H sequence using the BIOS CONOUT function (WBOOT + 9) to setup the screen mode. See Chapter 6.

(2) The current screen modes can be identified by checking the following work areas:
- Screen mode
    LSCMODE (0F2C9H for overseas version; 0EFFBH for

Japanese-language version)

= 00H: Screen   mode 0

= 01H: Screen   mode 1

= 02H: Screen   mode 2

= 03H: Screen   mode 3

- Virtual screen 1

> LV1SCT + 4 (ØF71FH for overseas version; ØF56DH for
> Japanese-language version): Contains the
> number of virtual screen 1 columns in binary
> form.

> LV1SCT + 5 (ØF72ØH for overseas version; ØF56EH for
> Japanese-language version): Contains the
> number of virtual screen 1 rows in binary
> form.

- Virtual screen 2

> LV2SCT + 4 (ØF725H for overseas version; ØF573H for
> Japanese-language version): Contains the
> number of virtual screen 2 columns in binary
> form.

> LV2SCT + 5 (ØF726H for overseas version; ØF574H for
> Japanese-language version): Contains the
> number of virtual screen 2 rows in binary
> form.

- Selected screen:

> LDSPVS (ØF2CAH for overseas version; ØEFFCH for
> Japanese-language version)

> = ØØH: Displays virtual screen 1.

= 01H: Displays virtual screen 2.

- Separation character

    LBOUNDP (0F2D9H for overseas version; 0F009H for

    Japanese-language version): Contains the character

    code proper.

19.10.9 Serial (common to both overseas and Japanese-language versions)

DHSDAT (ØFØ14H for overseas version; ØED14H for
          Japanese-language version)

    = Ø1H: 4,8ØØ bps

    = Ø2H: 6ØØ bps

    = Ø3H: 15Ø bps

19.10.1Ø Country (overseas version only)

YLDFLTC (ØF6A1H for overseas version)

    = ØFH: ASCII

    = ØEH: France

    = ØDH: Germany

    = ØCH: England

    = ØBH: Denmark

    = ØAH: Sweden

    = Ø9H: Italy

    = Ø8H: Spain

    = Ø6H: Norway

When this work area is altered, only the display modes
are changed and no keyboard mode is changed. After a

WBOOT, the character fonts of the selected country are enabled for display.

19.11 XON/XOFF Control for the Currently Open RS-232C Interface

The following work area is referenced to determine how XON/XOFF control is exercised for the currently open RS-232C interface:

SKXFLG (ØF6C4H for overseas version; ØF447H for
        Japanese-language version)

    Bit 4 = Ø: XON/XOFF control disabled
    Bit 4 = 1:   XON/XOFF control enabled

    Bit 6 = Ø: XON has been sent.
    Bit 6 = 1:   XOFF has been sent.

    Bit 7 = Ø: XON has been received.
    Bit 7 = 1:   XOFF has been received.

19.12 Procedure for Sending and Detecting the RS-232C
    Break Signal

19.12.1 Sending the RS-232C Break Signal

Run the following program:

        LD   A, 3FH

        OUT (ØDH), A

        Call software timer to provide a delay
        required for sending the code

        LD   A, 37H

        OUT (ØDH), A

19.12.2 Detecting the RS-232C Break Signal

Use the following program:

        IN   A, (ØDH)

    Areg. bit 6 = Ø: No Break signal

    Areg. bit 6 = 1: Break signal detected

See an 8251 manual for further information.