MICROMATION

M/NET MULTI-PROCESSOR SYSTEM

D01 017 REV A

Micromation
Addendum
June 15, 1981

MICROMATION SYSTEM UPGRADE

The M/NET system has undergone some changes since this manual was written. Most apparent is the re-design of the cabinet bezels. Behind the frontispiece dramatic changes have been made too. The computer cabinet is enlarged to allow system expansion to 8 users. In addition, we've changed to Qume double-sided floppy disk drives and incorporated the 8" Fujitsu Winchester hard disk drive for a more compact system.

These changes are not addressed in the Installation Guide nor Operator's Manual. Use this addendum to upgrade these manuals.

NOTE: Although the orientation of the cards within the cabinet has changed, board cable connections within the computer cabinet remain the same (e.g., floppy cable to Doubler). The descriptions for cable connections remain the same for the new cabinets.

PERIPHERAL PREPARATION

HARD DISK PREPARATION

The 8" Winchester-type hard disk drive contains a lock to prevent damage to the read/write heads during shipment.

> The lock must be released before the system is powered up to function properly.

The lock is released via a white plastic bar visible through the grill work in the front of the cabinet. As you look through the grill, the bar is set at the factory pointing to the left side of the drive. To unlock the drive, the bar is rotated to the right.

To unlock the drive

- Put the entire cabinet onto its left side (as viewed from the front).
- A window has been cut in the base plate for access to the bar.
- DO NOT attempt to shove the bar laterally. The drive casing has a small lip to prevent inadvertent movement of the lock.
- Gently pull the bar toward the base plate to clear the lip. Notice that if the bar is moved too far outward, the base plate obstructs rotation.
- With the bar pulled out slightly, slowly press it clockwise toward the top of the window.

**** IMPORTANT ****

Whenever the hard disk is transported (whether in or out of the cabinet) the lock must be reset. Before reversing this operation to move the bar back to the shipping position, run the SHIP.COM program. This moves the head to the proper position within the drive to ensure safe shipment. Refer to the READ.ME file on the distribution diskette for more information.

****

Notice that the storage space on the 8" hard disk is apportioned to provide three logical drives. This is a change from the configuration of the physical drive stated in the manual - 4 logical drives. These 3 work spaces are accessed via drive names E:, F: and G:; each with more than 6.5 megabytes of storage space.

QUME DRIVE PREPARATION

To prevent the read/write heads from contacting and causing damage during shipment, Qume dual head drives contain a cardboard insert and a tie wrap.

These must be removed before the drive is powered up.

TIE WRAP REMOVAL

- Remove the 4 cover screws from the bottom of the cabinet and slide the cover toward the back enough to expose about 8" of the drives inside.

- With a pair of end nippers or other suitable instrument snip the tie wrap that secures the door latch. It is located toward the front of the cabinet on the left hand side of each drive.

- Replace the cover and cover screws.

The cardboard insert is removed by opening the door. Note that this can be done only after the tie wrap has been removed. The insert will pop out. Save this card.

Transporting Qume drives around the office or town should only require re-installation of the cardboard insert. However, if a drive is to be shipped it is strongly recommended that the door latch mechanism be resecured with a tie wrap to prevent damage to the heads.

M/NET 1.16 RELEASE

In addition to the changes in the hardware, we've upgraded the utilities provided on the distribution diskette. These changes have removed the necessity for the CP/M for M/NET diskette. All references in the M/NET Operator's Manual to this diskette can be ignored.

The most significant change is the addition of the

FDISK.COM
HDISK.COM

utilities.

FDISK This utility performs floppy disk related copy and format functions. That is, entering

FDISK

at the terminal provides the user with a menu (list of options) for copying data from one diskette to another, formatting a diskette, etc.

HDISK Similarly, HDISK provides a menu for hard disk related operations. This program also provides selections in the menu for testing hard disk operation. Some of these procedures are complex and should not be attempted unless the exact nature of the operation is understood before the execution. Damage to data stored on the hard disk may result otherwise.

To have the menu displayed once the program has been invoked, enter a question mark, ?. The use of these utilities is described in the program.

**** IMPORTANT ****

BE VERY CAREFUL IN THE USE OF THESE PROGRAMS. The format options will erase all the information of the selected floppy diskette or hard disk.

****

Note that these utilities do run in the multi-user environment. That is, they can be run while other users are on-line. Whereas this is possible, it is not recommended. Since both copying and formatting functions require a large amount of the master processor's time, system response will slow down. (These are not programs that you will be running frequently, however, so this should not affect overall system performance profoundly.)

Two other new utilities are provided on the distribution disk:

PRINTER.PRL
SHIP.PRL

In addition, the

GENSYS.PRL

has undergone some changes.

PRINTER allows users to attach a specific printer to their program. The distribution version MP/M is configured for two printers numbers Ø and 1. (Printer Ø is assigned to the serial port on the DOUBLER; Printer 1 is assigned to the Centronics parallel port on the M/NET I/O.) Up to 16 printers, labelled Ø - 15, can be attached to the system with the addition of I/O cards and printer drivers in the XIOS portion of MP/M.

There are two ways to use PRINTER. Entering

PRINTER

will indicate which printer is assigned to that terminal (and, of course, program). This assignment can be changed by entering

PRINTER n

where n is a printer number between Ø and 15.

For instance, if there are two printers on an M/NET system, one fast Centronics-type, dot-matrix printer assigned to 1 and a letter quality, serial printer to Ø, the accounting department would enter

PRINTER 1

to get a fast print-out of the monthly transactions (where speed is more important than appearance). Subsequently,

PRINTER Ø

would be entered to print a reminder to customers that their accounts are past due on the letter quality printer.

A separate "mutual exclusion queue" is maintained for each printer number to prevent two users from accessing the same printer at the same time.

SHIP is used for one purpose: to move the read/write head in the hard disk to a special location for preparation for shipping. It is not a utility that is used in normal, daily data entry and processing. The READ.ME file supplied on the distribution diskette describes its use.

The GENSYS program has been altered to make the system more versatile and easier to configure. Consequently, the description on the use of GENSYS in Chapter 5 of the M/NET MULTI-PROCESSOR MANUAL no longer corresponds to the prompts in the program. Specifically, three questions have been removed and three new ones have been added. Following a description of the changes below, a sample GENSYS run is illustrated on a separate page of this addendum.

The prompts removed are:

```
Breakpoint RST #
Z8Ø CPU (Y/N) ?
Bank switched memory (Y/N) ?
```

Since the answers to these questions are always the same in M/NET (7, Y, and N, respectively), they are automatically provided within GENSYS.

The three new prompts are:

1)
```
Enter step time for floppys
8 for Shugart, 3 for Qume
Step time          =
```

Depending upon which type of floppy drives are in the system, answer 8 or 3.

2)
```
Type of printer?
1 for standard serial
2 for Centronics - parallel
3 for user defined
Printer            =
```

This prompt assigns system printer Ø. Note that if option 2 is selected, both system printers Ø and 1 will be assigned as Centronics - parallel. Also note that option 3 references a special driver routine that must be added to the XIOS.

3)
```
Auto line-feed printer?
Usually yes for 2, no for 1 or 3
Auto line-feed (Y/N)?
```

As the prompt states, printers of types 1 and 3 usually get a N response while type 2 typically gets a Y. This is not always the case, however. Be sure to check the printer documentation before making the selection.

The remainder of the prompts in the GENSYS program are the same as those described in Chapter 5.

Some changes have been made within the MP/M too. (These will have significance to programmers only.)

- BDOS function 6 (direct console I/O) in the slaves is implemented the same as CP/M BDOS function 6 as opposed to the MP/M BDOS function 6.

- MP/M BDOS functions 3 and 4 (raw console input and output, respectively) have been changed to look like CP/M BDOS functions 3 and 4 (reader input and punch output, respectively).

An IOBYTE similar to that in CP/M is set up in the satellite's memory space at location 0003 and is implemented as follows.

Reader Field (bits 2 and 3, values in binary)

00 - reads currently attached console
01 - reads console 13
10 - reads console 14
11 - reads console 15

Punch Field (bits 4 and 5, values in binary)

00 - writes to currently attached console
01 - writes to console 13
10 - writes to console 14
11 - writes to console 15

The XIOS can be appended to add 3 miscellaneous character I/O devices and access them from a satellite using the IOBYTE and the CP/M READER/PUNCH calls. It is the responsibility of the program running in the satellite to insert the appropriate value into location 0003 in the satellite's memory.

CHANGING THE XIOS

The source listing of the XIOS.SPR portion of MP/M appears as XIOS.ASM on the distribution diskette. If it is necessary to make changes to it (e.g., to add a special printer driver), two XIOS.ASM files must be created (each with a different ORG location), assembled (to make HEX files), concatenated and run through GENMOD. Before attempting this, review the procedure for creating SPR files described on page 87 in Digital Research's MP/M Manual.

SAMPLE GENSYS SESSION

NOTE: The MP/M shipped with all M/NET systems is configured for

4 consoles
4 satellites
a serial printer assigned as Ø

There is no need to run GENSYS if this satisfies your system requirements.

In the demonstration below, the program prompt is shown on the left and the possible answers are shown in bold under RESPONSES.

DISPLAY	RESPONSES
MP/M 1.1 System Generation ==========	
Top page of memory =	**F2** *
Number of consoles =	**2**, **3** or **4** for # of terminals
No. of slave Z-64s =	**2**, **3** or **4** for # of slaves
Add system call user Stacks (Y/N)	**Y** *
Enter step time for floppys 8 for Shugart, 3 for Qume Step time =	**8** or **3** depending on drives
Type of Printer 1 for standard serial 2 for Centronics - parallel 3 for user-defined Printer =	**1**, **2** or **3**
Auto line-feed printer? Usually yes for 2, no for 1 or 3 Auto line-feed (Y/N)?	**Y** or **N**
Memory Segment bases, (ff terminates list)	
: **ØØ** : **5Ø** : **6Ø** : **7Ø** : **FF**	These are the default settings set at the factory. Others may be selected.
Select Resident System Processes: (Y/N)	
MNET **Y** ABORT **Y** SPOOL **Y** MPMSTAT **Y** SCHED **Y**	MNET must be answered Y. The remainder can be answered with a N if they are not needed.

* The M/NET system should almost always get these two responses.

MICROMATION

M/NET

MULTI-PROCESSING SYSTEM
OPERATOR'S MANUAL

Micromation has added an extra dimension to MP/M*: a separate Z-80 processor for each user. The result is M/NET, the first microcomputer multi-processor system.

MP/M was designed as a single processor, multi-user system. This conceptual framework has profound limitations, though, as the humble microprocessor, either 8080 or Z-80, is required to perform tasks it was never designed to do. The most apparent drawback is the severe impact on program execution speed as more users are added. The system is visibly, and annoyingly, slowed down as user demands increase. Truly, a giant step backward in microcomputer utilization.

The Micromation alternative is to allocate a separate processor to each user. The basic structure of MP/M and all its features are maintained (including CP/M* compatibility) and the benefits of multi-processing are added. This is possible because of Micromation's unique Z-64 board. On a single S-100 bus card are a Z-80A (operating at 4 MHz), 64K of RAM, and vectored interrupt priority encoder. Five Z-64s are used in the maximum system; one master processor to control the system and 4 slave processors for program execution.

The advantages are impressive:

- program execution speed is minimally affected with the addition of each user;

- the familiar CP/M conventions are maintained;

- an extensive library of CP/M compatible software from high level languages (like FORTRAN and BASIC) to application programs is available to suit a variety of business and program development needs;

- a crash of one program does not affect the other users on-line at that time.

* MP/M and CP/M are trademarks of Digital Research, Inc.

SYSTEM COMPONENTS AND FEATURES

HARDWARE

Master processor:	Micromation Z-64 with Z-80A and 64K RAM; runs MP/M and controls slave and master allocation and all I/O (user and disk-related for the system.
Slave processor:	Z-64 with Z-80A and 64K RAM; contains CP/M emulator to translate CP/M commands to MP/M commands (ensures compatibility for application software)
M/NET I/O Board:	4 serial ports, one for each terminal, parallel ports for parallel interface printer (e.g., Centronics), real time clock and interrupt generation
DOUBLER:	Floppy disk controller for single/double density recording, also interfaces double sided drives; extra serial port for serial interface printer.
Hard Disk:	10 or 20 megabyte Winchester-type hard disk and controller

SOFTWARE

Enhanced MP/M:	Rewritten by Micromation to maintain MP/M and CP/M compatibility and add multi-processing; contains features to spool (print) files, determine time of day (real time clock), schedule programs to run at a specific date/time, detach from programs, run several programs at a time and more.
MP/M Utilities:	A broad range of programs for file management and maintenance, system status reports, system generation, program debugging and more.
Micromation Utilities:	Additional utilities to facilitate system operation.

AND MORE

- Single user has access to all slave processors from a single terminal and can run more than one program at a time.
- System can be easily expanded from two user system (or, from single user system) to four user system at any time
- Easy to operate: complex system processes performed transparently from the user's perspective; appears to the operator that s/he is the only operator.
- One user or several users can be simultaneously running programs written in different languages (BASIC versus FORTRAN, for instance).

HOW TO USE THIS MANUAL

Since M/NET can be run without a complete knowledge of the inner workings and many users will want to get the system up and running upon installation , system operation and features are introduced without a theory of operation.

Section I of this manual begins with booting the system and step by step illustrations of the processes required to set-up the system after installation. Most of these operations need only be performed once; once they've been done, the system is configured for day to day operation.

Section II demonstrates the commonly performed operations like formatting a disk, changing the currently logged disk, transferring files from one disk to another, changing user numbers, etc. Some of the more sophisticated operations are also illustrated although they may not be utilized by all users.

Section III is for those who are not content with a black box and must know the wizardry that makes it all possible. This section identifies the interrelationship of the hardware components of the system and supplements the MP/M documentation, pointing out the similarities and differences between the Digital Research version and the Micromation enhancement.

OTHER MANUALS IN THE M/NET DOCUMENTATION SET

M/NET INSTALLATION: The installation instructions for the M/NET system are separated from this manual. Once the system has been physically set-up, the user should refer to Section I of this manual for loading and configuring the operating system to match the hardware. Subsequent sections of the Installation Guide describe upgrading the system (adding components) and converting a single user system to a multi-processor system.

MP/M AND RELATED DIGITAL RESEARCH MANUALS: Digital Research supplies an extensive set of manuals with MP/M. One is dedicated to the MP/M operating system with references to the other manuals. These other manuals describe use of the utilities (called transient commands in the CP/M literature) supplied with the O/S. Additional utilities have been written by Micromation as well. Although the use of many of these programs is discussed in Sections I and II of this manual, it is strongly recommended that the Digital Research documentation be read for a complete understanding.

The remainder of the manuals in the M/NET documentation set are reference guides for the individual hardware components and have little utility for operators. Since many of the components in the M/NET system are also used in Micromation single-user systems, some sections in these manuals are not pertinent to the multi-processing system. These sections, dealing primarily with

installation and options, should be ignored; the boards are configured at the factory for the multi-processor system. Following is a list of the manuals included.

Z-64 Processor/Memory Board
Z-64 Satellite Processor Board
Multi I/O Board
DOUBLER Floppy Disk Controller
Hard Disk Controller

TABLE OF CONTENTS

SECTION I: INITIAL M/NET SET-UP

Chapter	Description	Page
1	System Components and Maintenances	2
2	Console Line-Editing Characters	5
3	Powering Up and Booting the System	7
4	Backing Up the System Diskette	10
5	Generating a System for a Specific Environment	13

SECTION II: M/NET FACILITIES AND OPERATION

6	Console Commands	19
7	File Types and Memory Segments	22
8	M/NET Utilities	26
9	Resident System Processes	57
10	System Level Operational Procedures	70

SECTION III: REFERENCE

11	Component Descriptions	88
12	MP/M Enhancements and Differences	93

APPENDICES

A	Glossary	101
B	File Naming and Reserved File Types	107
C	Diskette Formats	111

SECTION I: INITIAL M/NET SET-UP

Chapter 1 is a very basic introduction to the M/NET system. It illustrates and describes the system components and describes some basic precautions regarding hardware and diskette maintenance.

Chapter 2 describes the console line editing characters available under MP/M. These are used to correct mistakes made while typing in commands to the system.

Chapters 3 - 5 of this section include procedures that normally need only be performed once right after initial set-up of the system. They describe powering up and loading the system, making a back-up of the distribution diskette, and generating an operating system tailored to your hardware environment. Circumstances may dictate a repeat of one or all of these though especially concerning system generation. Regarding this last operation: some system features are optional; their inclusion or exclusion will depend on factors that can only be determined from experience. Consequently, the instructions in Chapter 5 generate a general purpose system, but your experience may subsequently dictate a different system configuration.

The chapters in Section II contain instructions and examples on the use of the utilities (programs) provided on the distribution diskette along with the MP/M operating system. You may need to reference these sections frequently at first until the function of each utility and its use has been assimilated. Chapter 10 describes basic M/NET operation (with suggestions) and summarizes the use of the utilities as they relate to some common operations frequently performed on computers (preparing diskettes running programs, using the printer, etc.).

It is recommended that Sections I and II be read in their entirety before sitting down with the system to perform the exercises.

CHAPTER 1: SYSTEM COMPONENTS AND MAINTENANCE

This chapter does not include instructions for setting up the system (installing the boards in the computer cabinet and attaching the cables). If you have not already done so, refer to the M/NET INSTALLATION GUIDE then return here once completed

Each M/NET system consists of certain basic components. The number of each component will differ from system to system though. Figure I:1.1 illustrates the basic components.

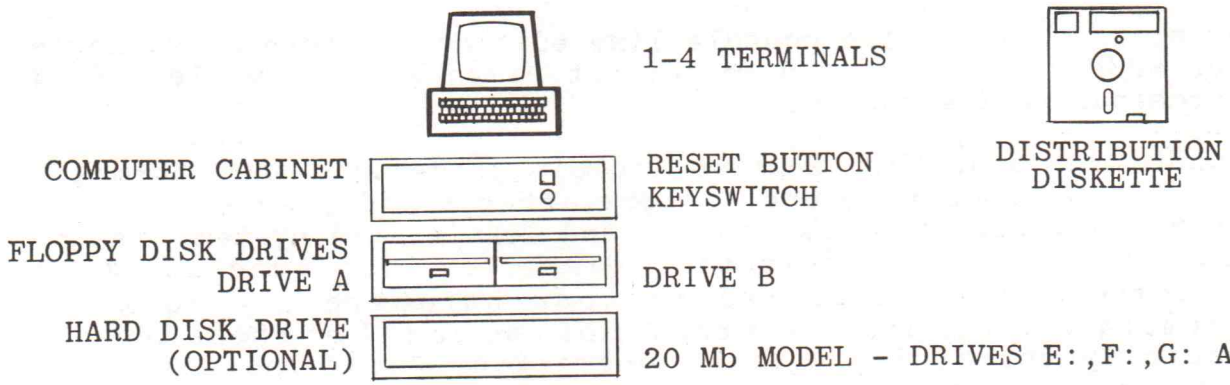


FIGURE I: 1.1 M/NET BASIC COMPONENTS

Your system may not be arranged as illustrated above, but the components have the same appearance. The features pointed out (e.g., RESET button, drive A:) will be referenced in the text.

TERMINAL: This system component allows operator entries through the keyboard and displays program outputs on the screen.

COMPUTER CABINET: This unit contains the printed circuit (PC) boards and electronics that make the system run. Most operators need know little about the functions of the boards. There is one exception, though, which is referenced throughout this manual: the master and slave processors.

MASTER PROCESSOR: In a very broad sense, this PC board controls the system. Users request a program by entering a command. The master interprets the command and decides if it will execute the program itself or pass it off to one of the slaves. In addition, the master controls the transfer of data between all the peripherals (floppy diskette to slave, terminal to slave, etc.). There is one master processor in the M/NET system.

SLAVE PROCESSOR A M/NET system may have from 2 - 4 slave processors. These PC boards are used to execute programs assigned to them by the master. In fact, they "talk" to the master processor only; they do not communicate directly with the operator at all.

FLOPPY DISK DRIVES: The standard M/NET system has two floppy disk drives, called drive A and drive B:, mounted in a single cabinet. Two more drives can be added at any time. (They would be referred to as drives C: and D:.) The drives are used to store data on floppy disks for later retrieval.

DISTRIBUTION DISKETTE: This is a special diskette shipped with the M/NET system that contains the operating system and utilities necessary for operation. Take special care of this diskette until it has been duplicated, without it the system does not run.

HARD DISK DRIVE: This is another data storage device similar to the floppy disk drives in concept but dissimilar in the amount of storage space available (20 megabytes versus 500 kilobytes) and data access speed (data is read or written to the hard disk much faster than the floppies). This is an optional component in M/NET systems but is strongly recommended.

A word about the hard disk drives: Unlike the floppy disk drives which are two distinct, physical drives, the drives in the hard disk are logical not physical and, hence, not apparent from the outside. The 20 megabyte storage space is divided into four 5 megabyte segments named drives E:, F:, G: and H:

SYSTEM MAINTENANCE

There are no strict rules for system maintenance, just some common sense precautions.

- Keep drinks and food away from the components. The warranty does not cover lettuce removal from the floppy disk drives or repair of cola encrusted printed circuit boards.

- Do not place the back of the cabinets against a wall. Good air circulation through the cabinet is essential; excessive heat cooks the integrated circuit chips and causes damage to floppy diskettes.

- If you smoke, don't let the smoke waft through the floppy drives. It will accumulate on the head mechanism (the part that reads the data from the diskette) and diskette, diminishing their ability to transfer data.

DISKETTE HANDLING AND FILE MAINTENANCE

There are some precautions that should be observed to ensure that files stored on floppy disk aren't inadvertently lost due to operator error or the slings and arrows of outrageous fortune.

1) **Handle the diskettes with care.** Keep your fingers away from the exposed areas on both sides of the disk. Store diskettes in the paper sleeve whenever they are not in the computer (dust accumulates, otherwise, and shortens the life of the read/write head on the floppy drive). Since diskettes are a magnetic media, be sure to keep them away from magnets. You should also keep the diskette out of direct sunlight and away from extreme heat. Never remove the mylar disk from its protective paper enclosure nor abuse the disk by folding or bending it. When possible, write the label before applying it to the diskette. Subsequently, use only felt-tip pens for additional notes.

2) **ALWAYS make a back-up of your files.** In many cases great time and effort is spent creating these files. It is much easier to make back-up copies on an on-going basis than to re-enter the whole thing over again. Make a back-up of any files changed **every time** they're changed. (After the program has terminated and the O/S prompt has re-appeared, of course.)

3) **NEVER power down the disk drives during program execution nor with a disk in a drive and the door closed.** Clear the floppy drives of all diskettes before turning off the power to the drives (opening the drive door will do). Powering down during program execution will, at least, render the data files suspect or, worse, trash a couple of sectors rendering the whole file or disk useless. Turning off power with a diskette in the drive (but not during program execution) may also trash a sector or two as the head responds to powering down.

   ALWAYS power down the hard disk drive before turning off the computer power. The on/off switch for the hard disk drives is mounted on the back of the cabinet.

   NOTE: Once the M/NET system has been turned on, it can (and should) remain on. There's no need to turn it off when it's not in use.

4) **Write protect your important diskettes.** The little notch on the left side of the leading edge (see Figure I:3.1 below) causes write protect when exposed. Cover it with tape (or the silver squares provided a box of type of diskettes) and the diskette can be written on.

If these precautions are observed, very few problems will arise and those that do can be easily repaired with the back-up disks.

CHAPTER 2: CONSOLE LINE-EDITING CHARACTERS

The control characters described in this chapter are a subset of a group of console control characters recognized by MP/M. The complete list is presented in Chapter 6. You will need the few described below in the event a mistake is made when entering the commands in the exercises that follow.

Control Characters are entered just like an upper case letter (when upper and lower case entries are allowed) except, instead of the SHIFT key and letter key being pressed simultaneously, the CTRL (control) key and letter key are pressed. These characters are designated as ^x or CTRL-x (where "x" is one of the various letter keys) in this manual.

RUBOUT: Many but not all terminal keyboards have a key labelled RUBOUT. It is used to delete erroneous entries letter by letter. If the character entered is wrong, press RUBOUT and it is removed from the command entry and echoed on the screen. For instance, one of the commands to the computer used in Chapter 4 is DIR (it displays the contents of the disk on the terminal screen). Let's say your typing skills aren't the greatest and you type DER instead of DIR and realize the mistake. Pressing the RUBOUT key twice removes ER from the command line and displays that fact; that is DERRE appears on the screen. As far as the computer is concerned, all it sees thus far is D. Subsequently, type IR and a RETURN and the command is executed.

NOTE: This is not a control character. Do not press the CTRL key when pressing RUBOUT.

CTRL-H: This control character performs the same function as RUBOUT but does not echo the character erased. For instance, the user who entered DER instead of DIR above could have held down the CTRL key and pressed the H key twice. The cursor moves backward and erases the previous character for each invocation. The terminal screen will show just D in this case.

CTRL-U: ^U is used to erase a whole line of input. If, for instance, a long command is typed on the terminal before it is realized that the second character of the line is erroneous, it is far more expeditious to erase the whole line than to use ^H or RUBOUT to correct the mistake. Press the CTRL and U keys, a "#" will be displayed and the cursor will jump down to the next line. As a convenience, the cursor is indented two spaces from the left margin.

CTRL-X: ^X works similarly to ^U but causes the cursor to return to the beginning of the line it's on, erasing the contents, instead of leaving the line intact and jumping down a line.

CRTL-R: It isn't always clear what letters are there and which have been erased when the RUBOUT key is used. Entering ^R causes the screen to display those characters not erased. It leaves the

cursor at the end of the last character allowing you to enter the remainder of the command.

Use of these functions assumes that the user has realized a mistake was made before pressing the RETURN key. (A RETURN causes the command to be interpreted by the computer.) If the mistake was not caught, no damage is done. When the computer gets a command it cannot understand, it redisplays the command followed by a question mark (?) and the system prompt allowing entry of another command.

These and the rest of the line editing and output control function keys are discussed in Chapter 6.

CHAPTER 3: POWERING UP AND LOADING THE OPERATING SYSTEM

The steps in this chapter describe the procedure for cold-booting the M/NET system. "Cold-booting" means reading the operating system, MP/M, from the floppy disk and loading it into the computer. It connotes that the system was just powered up and/or the RESET button was used to begin the load process.

1) Power up ALL the units in the system. Specifically, flip the switch on

   - all terminals
   - the back of the floppy disk drive cabinet
   - the back of the hard disk drive (if present in your system)
   - the back of the computer cabinet

   The terminals take about 30 seconds before the cursor (an up arrow or rectangular box, depending upon the type of terminal) appears. It takes about 2 minutes for the hard disk drives to warm up.

2) Turn the keyswitch on the front panel of the computer cabinet clockwise 180 degrees (half circle). The drive A: light (and perhaps the drive B: light as well) goes on.

These power on switches can all remain in the ON position permanently. There's no need to turn off any component in the system in normal operation.

3) Press the system RESET button mounted on the front of the computer cabinet above the keyswitch. (If the drive B: light went on above, it should go off now.)

4) The next step is to insert the distribution diskette into the disk drive. Before doing so, however, ensure that the WRITE PROTECT notch is **exposed.** It is normally left uncovered after the system has been recorded on the diskette at Micromation. It should be checked though as a precaution.

   Orient the diskette for insertion as illustrated in Figure I:3.1. Notice that the label must be facing up and the edge with the WRITE PROTECT notch goes in first. Push it in drive A:, the leftmost in the floppy drive cabinet. Close the door.

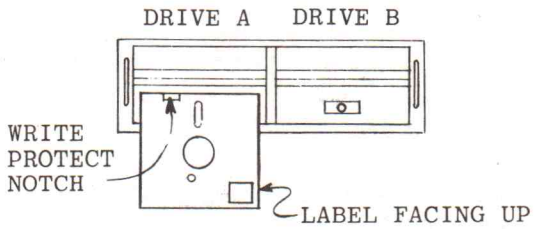


FIGURE I: 3.1 DISKETTE ORIENTATION

Assuming you are sitting at console Ø, the MP/M sign-on, configuration messages, and prompt as shown below are displayed. Note that the display will stagger occasionally as the messages are typed on the terminal screen. This is normal. The loading process takes about 2Ø seconds.

```
MP/M 1.1 Loader
===============

Number of consoles =  4
Breakpoint RST #   =  7
Z8Ø CPU
Top of memory      =  F2FF

Memory Segment Table:
SYSTEM  DAT  F2ØØH  Ø1ØØH
CONSOLE DAT  EEØØH  Ø4ØØH
USERSYS STK  EDØØH  Ø1ØØH
XIOS    SPR  DFØØH  ØEØØH
BDOS    SPR  CBØØH  14ØØH
XDOS    SPR  ABØØH  2ØØØH
Mproc   RSP  9DØØH  ØEØØH
ABORT   RSP  9CØØH  Ø1ØØH
MPMSTAT RSP  8FØØH  ØDØØH
-------------------------
Memseg  Usr  7ØØØH  1FØØH
Memseg  Usr  6ØØØH  1ØØØH
Memseg  Usr  5ØØØH  1ØØØH
Memseg  Usr  ØØØØH  5ØØØH

MP/M 1.1 Micromation M/NET Ver. 1.11
ØE>
```

NOTE: The values shown above indicate those displayed when this manual was written. Some may be different in your system.

If you are not seated at console Ø, only the following is displayed.

```
MP/M 1.1 Micromation Ver. 1
xE>
```

"x" will indicate the console number, currently 1 - 3.

If your M/NET system does not include a hard disk, the display is the same except that the last line displayed is

ØA>

assuming you are at console Ø or xA> (where "x" is the console number) if you are not.

The display lines at console Ø have the following meanings:

Number of consoles: The system generated at Micromation is created for 4 consoles (terminals). It is irrelevant to system operation if you have fewer consoles in your system. (Chapter 5 describes changing this value to reflect your hardware configuration if you would like to change it.)

Breakpoint RST #: The breakpoint restart number used by the SID and DDT utilities is indicated here. In the Micromation system, RST 7 must always be used.

Z8Ø CPU: The Z-64 uses the Z8Ø microprocessor.

Top of memory: The value displayed indicates the highest memory location available in the master for storage of the operating system.

Memory Segment Table: The location and size, respectively, of the various MP/M components as arranged in the master are displayed. The display in your system may be a bit different.

Memseg Usr: MP/M allows the master Z-64 memory area not used by the system (ØØØØH - 8FØØH in the sample display) to be apportioned into segments. The operating system shipped with M/NET has 4 sections beginning at the locations shown in the lefthand column. The righthand column indicates the size of each segment. These segments can be changed as part of the GENSYS (generate system) process described in Chapter 5. The use of the segments is described in Chapter 7.

MP/M 1.1 Micromation M/NET Ver. 1.11: This is the MP/M sign-on message. The Micromation version number will change as new revisions are distributed.

ØE>: This is the MP/M prompt. The number indicates the console number when the system is cold-booted (loaded for the first time). This number also indicates the user number. There is a difference between console number and user number which is described in Chapter 8. The letter indicates the current drive. Although you loaded MP/M from a diskette in floppy disk drive A:, the loader program performed a jump to drive E:, one of the logical drives of the hard disk.

NOTE: If your M/NET system does not include hard disk, the prompt will be "ØA>" indicating that floppy disk drive A:, the drive from which the operating system was loaded, remains as the current drive.

This is a cursory introduction to the O/S components and features. Detailed information on each is presented as it becomes relevant.

CHAPTER 4: BACKING UP THE DISTRIBUTION DISKETTE

As a safety precaution, the diskette used to boot MP/M, the distribution diskette, should be copied onto another. The original should be stored in a safe place and the copy used in subsequent operations. Keep the original handy in case the copy fails or you need to get an MP/M update from Micromation. (You must send the original to Micromation to get updates.)

The operations below call for a new or used diskette (the latter with information you can afford to erase) to receive the data from the distribution diskette. As a convenience those M/NET systems with the hard disk have the files recorded on drive E: at the factory. Consequently, the files are backed up but the system loading program is not. The program that loads MP/M into the computer is stored in a separate area of the diskette, tracks Ø and 1, than the files. For this reason, the procedure that follows creates a duplicate of both the files and the loading program. Since procedures described in subsequent chapters will require other diskettes, you may want to secure a box of them at this time.

The procedures in this chapter require the execution of several of the programs provided on the diskette. These programs, commonly referred to as utilities, will not be discussed at this time. (Chapter 8 describes them in detail.) You do not need to know how they work to run them. The program questions and options are displayed during execution, many of which are self-explanatory, and the instructions below provide the answers.

In this and other chapters in this manual, the program excerpts and user responses have the following format and special character.

- The program display is shown in darker print on the left side of the page.

- Comments are shown on the right side of the page in a slightly lighter shade print. Each line of the comment is preceded by a semi-colon, ";". The comments do not appear on the terminal screen.

- All user entries including spaces are underscored. In all examples, enter only that portion of the command that is underlined.

- A "@" is used to designate RETURN. Whenever this character is shown, press the RETURN key on the keyboard. **Do not enter the "@" character.**

One last note before proceeding: all user entries are shown in upper case only. This is for illustration purposes only. Commands may be entered in lower case, if desired.

**** CAUTION ****

The two steps in this chapter must be performed when no operators are using floppy drive B:. This shouldn't be a problem if the system has just been installed and the initial set-up procedures are being performed. However, subsequent format and copy operations may be required when other users are active, some of whom may be accessing drive B:.

If there's no diskette in drive B:, you are cleared to perform the two steps. If there is, enter

ØE>DSKRESET B:@

before removing it. Only remove it if the following message IS NOT displayed.

Disk reset denied, Drive B: Console c, Program pp

where c is a console number (Ø - 3) and pp any program. (The meaning of the DSKRESET command is described in Chapter 8.)

If the message is not displayed, insert the diskette in drive B: and enter

ØE>DSKRESET B:@

again. MP/M must be told when a diskette has been changed or inserted in a drive.

****

Step 1: Format Back-up Disk

The first step toward making a back-up diskette is to initialize a new or used diskette. Initialization in this context is a process wherein the diskette is prepared to store data. In computer jargon, this is called diskette formatting and is performed by a utility (program) named FORMAT. Briefly, this program erases all the data written on a disk and prepares it for storage of new data.

**** IMPORTANT ****

If you are going to use a diskette that has data stored on it to receive the data from the distribution diskette, the format process irrevocably erases all data stored on it. **Do not** use a diskette that has important or irreplacable data on it.

****

To begin, cover the write protect notch of the diskette to be formatted (so you can write on it), insert it into drive B (the right drive) with the same orientation as the diskette in A:, and close the door The demonstration below picks up the procedure starting with the display of the MP/M prompt. It assumes that you are at console Ø.

NOTE: If the write protect notch is not covered and a write to the disk is attempted (as by FORMAT in the first step below), "PROTECTED" is displayed on the screen and processing stops. Entering a RETURN terminates the operation and results with the display of the system prompt. Alternately, the disk can be removed (once the red drive light has gone off please), the notch covered, the disk re-inserted and the door closed. Subsequently, a RETURN will execute the command previously entered.

** Refer to the M/NET addendum for the FORMAT description **

Step 2: Copy Distribution Diskette to the Other

Once the disk has been formatted, the information on the diskette in drive A: can be transferred to that in drive B:.

** Refer to the M/NET addendum for the COPY description **

An identical copy of the distribution disk now resides in drive B:. Remove the original from drive A: and store it in a safe place, observing the precautions described in **DISKETTE HANDLING** above.

Remove the diskette in drive B: and label it. Prepare the label before sticking it to the diskette. Regarding labels, they should contain enough information to completely identify the nature of the disk. For instance the following may be useful for the new system disk.

```
BACK-UP SYSTEM DISK
62K MP/M VERSION 1.x
COPYRIGHT 1980 DIGITAL RESEARCH AND MICROMATION
MM/DD/YY
```

where "x" is the revision number displayed in the sign-on message and MM/DD/YY is the date. The copyright message must be written on each copy you make. This is the minimal information; make additional notes to indicate special programs when applicable. Use a felt tip pen to add notes to the label and do not press hard. Use of a ball point pen may damage the diskette.

CHAPTER 5: GENERATING A SYSTEM FOR A SPECIFIC ENVIRONMENT

In Chapter 4, a back-up of the distribution diskette was made. In this chapter, the MP/M operating system will be tailored to your hardware configuration with the GENSYS utility (program). Note that this program session requires a small but significant reference to the M/NET INSTALLATION GUIDE; it requires that you know the number of slave processors in your system. If you don't know how many slaves you have, refer to Chapter 2 of that manual to refresh your memory.

Since the M/NET system can be configured in any number of ways with different hardware components, all the permutations cannot be addressed by the operating system shipped from Micromation nor in this manual. To satisfy this dilemma, the O/S has been configured for the maximum system allowable (four terminals and four slave processors) and this section describes the means used to change these factory set values. Note that the system shipped from Micromation will run in any M/NET system, regardless of the hardware components. System performance is affected (i.e., it will run slightly slower), but the system will run.

Execution of the GENSYS utility creates three new files on the diskette: MPM.SYS, CONSOLE.DAT and SYSTEM.DAT. Refer to the illustration of the DIRectory above and notice that files of that name presently exist on the distribution diskette. These files contain the MP/M operating system; those on the distribution diskette are the ones made by Micromation. When the system is cold-booted, these files are read from the diskette and loaded into the master processor's memory. The files created in sample session below will replace those shown in the directory.

(For those familiar with CP/M operation, this is a bit different than its boot process where the system is stored on the first two tracks of the diskette. With MP/M, what was formerly the system tracks, Ø and 1, now contain a system loader program that reads the MPM.SYS, SPR and selected RSP files from the disk. Chapter 12 describes this in detail.)

The remainder of this chapter conducts you through a sample run of GENSYS. This run can be used to change MPM.SYS to agree with your hardware configuration or just for practice.

**** NOTE ****

If you have a hard disk, GENSYS execution will leave MPM.SYS, CONSOLE.DAT, and SYSTEM.DAT on drive E:. From here you will need to PIP them to the floppy in drive A: to load the new system (the instructions will describe this procedure).

******

If you do not have a hard disk, a couple of differences should be noted.

- the MP/M prompt will be ØA> in the sample below instead of ØE>, indicating that the current drive is floppy drive A:;

- the files MPM.SYS, CONSOLE.DAT and SYSTEM.DAT will be deposited on the diskette in drive A:, replacing those currently residing there.

All other program characteristics are the same as illustrated below. In fact, the user responses can be the same as those shown.

To demonstrate GENSYS, each program question is shown in darker print and followed by an explanation of the prompt. Those prompts that require a specific answer have the correct response shown. Those responses that are dependent upon the user's system configuration contain an "x" instead. Substitute the "x" with the appropriate answer for your system.

A REMINDER: In the GENSYS demonstration, all user entries are underscored and @ is used to indicate a RETURN should be entered.

To invoke GENSYS, type

```
ØE>GENSYS@
```

and

```
MP/M 1.1 System Generation
==========================

Top page of memory = F2@
```

is displayed.

The first GENSYS prompt requests for the top page of memory. Recall from the discussion in Chapter 3 of the console Ø MP/M configuration display that the top of memory was shown. This program prompt allows you to change this parameter. Generally, the largest size possible, that shown in the original MP/M sign-on display, is desirable. Note that only two numbers can be entered and that they are hexadecimal. These digits correspond to the first two of the four digit number. Respond with the value shown above (the same number as shown in the original MP/M configuration display).

Entry of a RETURN, causes the next program prompt to be displayed.

```
Number of consoles = x@
```

The number of consoles (1 - 4) is stipulated from this prompt. Respond to this prompt with the number of terminals attached to the M/NET computer cabinet followed by a RETURN. For instance, a user with a 2 terminal system would substitute 2 for "x" above.

```
No. of slave Z-64s = x@
```

Again, a RETURN renders the next question. Although the number of slave Z-64s usually is the same as the number of consoles, this isn't always the case. In Chapter 2 of the M/NET INSTALLATION GUIDE, you recorded the number of slave processors in system. Substitute this number for "x" in response to this prompt.

```
Breakpoint RST #    = 7@
```

The number 7 **MUST ALWAYS** be entered in response to the breakpoint prompt.

```
Add system call user stacks (Y/N) ? Y@
```

System call user stacks are explained in the MP/M User's Guide. A "Y" should be entered in response to this question.

```
Z80 CPU (Y/N) ? Y@
```

This, too, should always be answered with a "Y". The master and slave processors both feature the Z80 microprocessor.

```
Bank switched memory (Y/N) ? N@
```

Single processor systems use bank switched system memory to increase the amount of addressable memory available to the system. There is no need for this in the M/NET system since multi-processors are used instead. Consequently, always answer this prompt with "N".

```
Memory segment bases, (ff terminates list)
:00@
:41@
:6E@
:7F@
:FF@
```

As described in Chapter 3, the memory space in the master processor can be divided up into distinct segments. Each segment is available to the master processor for the execution of different programs. That is, the master processor can run several programs at once. (The slaves can only execute one at a time.) The size of each segment is determined by entering the base address of each section. Only two digit entries are allowed corresponding to the first two digits of the four digit memory location. The last two digits are filled in with 0s by GENSYS. Any number of segments can be created. However, there isn't much point in having lots of very small sections; each will be too

small to run a program. We recommend the values shown above until experience shows that other values would be more appropriate.

```
Select Resident System Processes: (Y/N)
MNET    ? Y@
```

Several programs, called Resident System Processes or RSPs, can be put in or left out of the operating systemwith one exception: MNET. This program is necessary for multi-processor operation. Of the remainder RSPs, inclusion means faster system performance at the expense of decreased master memory space available for memory segments. (The selected RSPs are appended to the beginning of the MP/M system area in memory subtracting from the area available for user programs.) Exclusion means a larger memory space available for segmentation but slower system performance. The difference in execution speed is because the master must read the program from disk when it is requested by the user instead of having it immediately accessible in system memory. The criteria to use when determining which of the resident system processes to select is, "how frequently will I use this program?" If infrequently is your response, leave it out; if frequently, answer "Y" to the program prompt. Until you are familiar with the RSPs (they are explained in Chapter 9) and know the regularity of their use, answer each with "Y".

```
ABORT   ? Y@
SPOOL   ? Y@
MPMSTAT ? Y@
SCHED   ? Y@
```

After the last question has been answered, GENSYS puts the MP/M system together and writes on the currently logged disk the files MPM.SYS, SYSTEM.DAT and CONSOLE.DAT. This takes a short while so don't be alarmed if the system appears to be in limbo a couple of seconds. When GENSYS is done, the MP/M prompt is displayed.

If you purchased M/NET without a hard disk, the MPM.SYS written to the diskette as part of the GENSYS program replaces the original provided with the system. To see what you're changes have wrought, press the RESET button on the front of the computer cabinet.

**** WARNING ****

Do not press the RESET button if there are other operators using the system unless you want to make some enemies. Pressing this button causes a system reset, terminating all programs currently in progress. Unexpected and unsolicited program terminations tend to make people very angry.

****

System reset causes the operating system to be loaded from the diskette in drive A:. The system just created will be read. If

you made changes from the original, the sign-on MP/M configuration display will be different than that shown in Chapter 3. The changes may be minimal; only manifested as a difference in the number of consoles.

If your M/NET system includes a hard disk, the MPM.SYS, SYSTEM.DAT and CONSOLE.DAT files created by GENSYS are written on hard disk drive E: and did not replace the original on floppy drive A:. To load the system just created, these files currently on drive E: must replace their equivalents on drive A:. To do this, type

```
ØE>PIP A:=E:MPM.SYS[V]@
```

and, after the ØE> prompt has been redisplayed

```
ØE>PIP A:=E:*.DAT[V]@
```

These commands may be gobbledy-gook to those uninitiated in PIP use. Chapter 8 explains each part of the command. Until then, it isn't necessary to know.

The PIP program is used to move files from one disk(ette) to another. If the destination disk (the disk to receive the files, A: in this example) already has a file with the same name as the one being transferred to it, that on the destination is erased in favor of the one being transferred. In this example, the original MPM.SYS is erased when the new one from drive E: is written. The same is true for CONSOLE.DAT and SYSTEM.DAT.

Press the RESET button on the computer cabinet after reading the WARNING above and the new system is loaded. Your changes can be seen by comparing the sign-on message against that shown in Chapter 3.

**** IMPORTANT ****

Everytime you need to cold-boot or reset the system, which should not be very frequently once it is in general use, you will need to use this diskette. You cannot load the system from the hard disk. So keep it handy; stored in its paper sleeve of course.

****

This concludes the chapters on initial operating system set-up. The chapters in Section II describe in detail the utilities (some of which you have already used), system features, and suggestions on system operation. Chances are you will not need to perform the procedures described in the above chapters again with the exception of new system generation. Once you are familiar with the Resident System Processes (RSPs), you may want to run GENSYS again to remove those not relevant to your operation.

SECTION II: M/NET FACILITIES AND OPERATION

The chapters in this section describe use of the M/NET utilities with respect to the day to day operation of the system. Exercises are provided to illustrate their application and demonstrate their form. First-time users may need to refer to these chapters frequently until the variety of applications and forms have been assimilated.

The chapters in this section are intended to supplement the Digital Research MP/M documentation, specifically the MP/M User's Guide, An Introduction to CP/M Features and Facilities, and CP/M 2.Ø User's Guide for CP/M 1.4 Owners The Digital Research manuals describe the utilities and all their various applications but are remiss in describing the why's and whatfor's of each. That is, they do not always explain why a utility is or should be used nor what it does. This section not only shows use of a utility but, in addition, states the reason for its use and what gets done. The Digital Research manuals should be referenced for a complete description of a utility after the general application has been gleaned from this manual.

CHAPTER 6: CONSOLE COMMANDS

Several console commands for line-editing and output control are available with MP/M. Chapter 2 introduced four of the line-editing characters: RUBOUT, ^H, ^U and ^R. Recall that the control characters are entered by pressing the CTRL and letter keys simultaneously, just like using the SHIFT key to get upper case letters.

These commands also have application in some of the MP/M utilities. Refer to the Digital Research manuals to determine their suitability.

LINE-EDITING FUNCTION KEYS

RUBOUT: Many but not all terminal keyboards have a key labelled RUBOUT. It is used to delete erroneous entries letter by letter. If the character entered is wrong, press RUBOUT and it is removed from the command entry and echoed on the screen. For instance, one of the commands to the computer used in Chapter 4 is DIR (it displays the contents of the disk on the terminal screen). Let's say your typing skills aren't the greatest and you type DER instead of DIR and realize the mistake. Pressing the RUBOUT key twice removes ER from the command line and displays that fact; that is DERRE appears on the screen. As far as the computer is concerned, all it sees thus far is D. Subsequently, type IR and a RETURN and the command is executed.

NOTE: This is not a control character. Do not press the CTRL key when pressing RUBOUT.

CTRL-H: This control character performs the same function as RUBOUT but does not echo the character erased. For instance, the user who entered DER instead of DIR above could have held down the CTRL key and pressed the H key twice. The cursor moves backward and erases the previous character for each invocation. The terminal screen will show just D in this case.

CTRL-U: ^U is used to erase a whole line of input. If, for instance, a long command is typed on the terminal before it is realized that the second character of the line is erroneous, it is far more expeditious to erase the whole line than to use ^H or RUBOUT to correct the mistake. Press the CTRL and U keys, a "#" will be displayed and the cursor will jump down to the next line. As a convenience, the cursor is indented two spaces from the beginning of the line.

CTRL-X: ^X works similarly to ^U but causes the cursor to return to the beginning of the line it's on, erasing the contents, instead of leaving the line intact and jumping down a line.

CTRL-R: It isn't always clear what letters are there and which have been erased when the RUBOUT key is used. Entering ^R causes the screen to display those characters not erased. It leaves the cursor at the end of the last character allowing you to enter the remainder of the command.

CTRL-E: In some situations it is desirable to have the cursor return to the beginning of a line without entering a RETURN. This is especially true when a long command needs to be entered. ^E performs this function; the cursor returns to the first position on the following line without the command going to the computer (which happens when a RETURN is entered).

CTRL-J: This control character causes a line feed. ^J can be used instead of a RETURN to terminate the command entry.

CTRL-M: This control character causes a carriage return. It, too, can be used instead of a RETURN to terminate the command entry.

CTRL-Z: ^Z indicates the end of an input string to the computer. It has application primarily in PIP and the EDitor utilities.

These control characters may seem confusing if you are not familiar with them. Our recommendation is to try out the line-editing functions as you perform the exercises.

PROGRAM CONTROL FUNCTION KEYS

CTRL-C: Under MP/M, ^C is used to abort the current program on a console. The O/S subsequently displays the system prompt. Use of ^C does not affect other user's programs nor other programs running from the same console.

NOTE: Under CP/M, ^C was used to indicate a change of diskettes (warm reboot) as well as program abort. Under MP/M, the change disk function has been assigned to a separate utility, DSKRESET.

CTRL-D: This control character is used to detach a program from the console so that another can be executed. If the detached program does not require any console input or output, it will proceed. If it requires access to the console, the program will idle until it has been re-attached. ^D is a toggle; that is, the first entry detaches the program, another re-attaches it.

Examples on the use of these control characters is contained in the examples that follow in subsequent chapters.

CONSOLE OUTPUT FUNCTION KEYS

CTRL-P: When ^P is entered, subsequent console input and output is sent to both the console and the system list device (printer). Another ^P entered from the console terminates the output to the printer (^P is also a toggle). If the list device is busy, the message "Printer busy" is displayed.

CTRL-S: ^S is used to stop the screen display or printer temporarily. This is handy when using the TYPE or PIP utilities (either of which can be used to display a file on the terminal screen and/or print a file) so that the operator can read the contents without terminating the program. Another ^S or entry of any key causes the display (printer) to continue.

CTRL-Q: ^Q allows a user to seize the printer, whether or not s/he needs it at the moment or not. No other user will have access to the printer until the ^Q originator releases it with another ^Q or ^P. This control character is also necessary if a program requires the printer but does not take it when the program is initially invoked. Entering ^Q before the program is called up, assigns the printer to that console only, other users requesting use of it get the "Printer busy" message. Whenever the program requests output to the printer, MP/M attaches the printer to that console.

NOTE: The Micromation enhanced MP/M has fixed the bug that made this control character necessary. As soon as a program has begun output to the printer, no other users or programs can access it until that program has terminated.

CHAPTER 7: FILE TYPES AND MEMORY SEGMENTS

What appear to be disparate topics for a chapter are actually closely related. Before revealing the reason, a word about file types is in order.

FILE TYPES

Under MP/M (as in CP/M), files are identified by a two part name. The first part consists of 1 - 8 letters or numbers and is called the file name. The second part consists of 1 - 3 letters or numbers and is called the file type. The two are physically separated by a period ("."). The file name is left to the whim and fancy of the operator; there is no significance to the name beyond its meaning to the user. Several file types, however, have significance. For instance, enter from the console (recall: user entries are underscored and @ indicates a RETURN should be entered)

ØE>DIR@

and a list of the files on drive E:, one of the hard disk logical drives, is displayed. If you do not have a hard disk in your system, enter the same command and the list of the drive A: files is displayed (assuming the ØA> prompt was displayed). Although some files may differ (files are added and removed by Micromation as new ones are developed) and their order rearranged, the group below will serve for illustration purposes.

```
Directory for user Ø:
E: SYSTEM   DAT : MPM      SYS : XDOS     SPR : BDOS     SPR
E: XDOSADD  SPR : XIOS     SPR : MNET     RSP : ABORT    RSP
E: MPMSTAT  RSP : SPOOL    RSP : SCHED    RSP : CONSOLE  PRL
E: DIR      PRL : DSKRESET PRL : ERAN     PRL : ERA      PRL
E: MPMSTAT  PRL : PRLCOM   PRL : REN      PRL : SUBMIT   PRL
E: TOD      PRL : TYPE     PRL : USER     PRL : SPOOL    PRL
E: STOPSPLR PRL : SCHED    PRL : ABORT    PRL : ED       PRL
E: PIP      PRL : STAT     PRL : RDT      PRL : ASM      PRL
E: GENMOD   COM : GENHEX   COM : LOAD     COM : PIP      COM
E: MPMLDR   COM : GENSYS   COM : ED       COM : DDT      COM
```

Temporarily disregarding the file names (they will be addressed in Chapters 8 and 9), note the six different file types (they are separated from the file names by a space, not a period, in the DIR listing).

DAT, SYS, SPR: These three file types are used to identify components of the operating system. Recall from Chapter 5 that execution of the GENSYS program creates a file called MPM.SYS from the features selected which, in turn, is read and loaded into memory whenever the system is cold-booted. This is but one component, the user adjustable portion, of the MP/M operating system. Other components are identified by

file types DAT and SPR. Refer to the MP/M User's Guide for a more exhaustive description of these files.

RSP: Those files with type RSP are the Resident System Processes. Recall again from Chapter 5 the GENSYS question

Select Resident System Processes: (Y/N)

This prompt addressed files with the RSP file type. They differ from other files in that they are operations that may be included in or excluded from the operating system. (Files with the file types mentioned in the preceding paragraph must be in the O/S, and those with the file types described below cannot be included.) The different files of this type are described in Chapter 9.

PRL: This file type indicates Page Relocatable files. Page relocatable means that the program can run at any page in memory. (A page of memory equals 256, or 100 hex, bytes of RAM.) In other words, the program can run from a base in system memory of 0200H ("H" indicates hex), or 2000H, or 3F00H, etc. This distinguishes files of type PRL from files of type COM which can only be run from a base location of 0000H. This feature is applicable to programs written in 8080 or Z80 assembly language. Some high level languages (FORTRAN for instance) can be used to create PRL-type programs also. Chapter 8 describes the PRL files shown in the DIRectory.

COM: As indicated in the preceding paragraph, files of type COM only run from a base location of 0000H; they cannot be run from any other page base in system memory. This is an important distinction significant in the discussion on Memory Segments.

There are several other reserved file types (i.e., they have special significance to the system and must be used to indicate the nature of the file). Appendix B identifies them and describes their meaning and application.

MEMORY SEGMENTS

The GENSYS program allows the user to subdivide the Master processor Z-64 board's memory space (that not allocated to the system, that is) into segments. NOTE: Only the Master can be subdivided; this feature is not applicable to the slave Z-64s. The distribution diskette contained 4 segments and the sample run of GENSYS also created 4. More or less and different sized divisions can be stipulated during GENSYS.

You may be asking yourself at this point, "Why do I need memory segments for anyway?" Although their significance is diminished a bit in M/NET with the addition of the slave processors, they allow several programs to run in the master processor at the same time. Depending upon the number and size of the memory segments

AND the memory requirements of the program(s) requested, one file of type COM and n-1 (where n is the total number of memory segments) files of type PRL can run simultaneously.

Figure II:7.1 illustrates the relationship between the total memory space and the memory segments within the master processor according to the allocations set at the factory. (The MP/M Loader message illustrated in Chapter 3 shows the values.) Notice that the system area for MP/M sits at the top memory leaving everything below for program execution. (This figure does not break down the system area into its constituent components. See the MP/M User's Guide for this breakdown.)

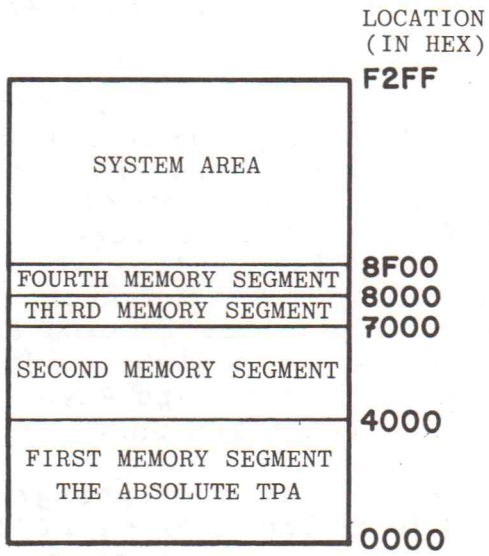


FIGURE II: 7.1 MASTER PROCESSOR MEMORY STRUCTURE

Notice in the diagram that there is only one absolute TPA (transient program area). This is the only area in the master where programs of type COM can run. This is because COM files require a base address of 0000 to run. Since most application software relies on the use of COM files, this could be a serious restriction. However, each slave processor in the system can be thought of as alternate TPAs since each has a memory space starting at 0000.

Files of type PRL can run in any of the memory segments with one restriction: the memory space must be big enough to run the program. MP/M automatically determines if the space is big enough. Note that PRL files are never run in one of the slaves (a system attribute); they are only run in memory segments within the master processor.

The advantage of having several memory segments in addition to the absolute TPA is that each can be running a different program at the same time. For instance, the world's largest manufacturer

of window latches has purchased a two-processor M/NET system. Mary (the receptionist, bookkeeper, secretary and all-around goodperson) at console Ø can perform a DIRectory of hard disk E: while at the same time Hank (the company's inventory manager) at console 1 in another office does the same thing for hard disk F:. Or Mary can grab each memory segment available by invoking a PRL-type program, detaching the console from that program, invoking another, detaching from it, and starting another. (Chapter 1Ø describes use of the detach feature.)

You may be asking yourself at this point, "Why should I, a humble operator who doesn't want to know how the computer works, care about memory segments?" Well, Mary found out she could run several programs at the same time, a real time-saver in some circumstances. Hank found out that Mary was real selfish when he got the message

```
Reloc seg not free
```

meaning "relocatable segment not free" or, "there are no more memory segments available at this time."

The factor that ties this chapter together (if you haven't figured in out) is the relationship between the memory segments and files of types PRL and COM. COM files only run in the master's first memory segment (called the absolute transient program area - TPA) and the slaves while PRL files run only in the master's memory segments.

CHAPTER 8: M/NET UTILITIES

Utilities are programs written to perform specific tasks. These tasks are necessary to computer operation and include such processes as moving files from one diskette to another, list on the terminal the files on a diskette, erase files, display the contents of a file on the terminal, among others. For instance, the operator used several utilities in Section I to format a diskette (with FORMAT), to copy the entire contents from one diskette to another (with COPY), and generate a new system (with GENSYS). In the context of M/NET, all files with a PRL or COM file type can be considered the utilities.

(The difference between files of type PRL and COM is discussed in Chapter 7. Note that COM files can be changed to PRL files and vice versa if the need arises. See the MP/M User's Guide and the discussion of the PRLCOM utility for the procedures.)

The utilities addressed below are only those that are provided on the distribution diskette shipped with the M/NET system. This is not to say there can't be others. Additional PRL and/or COM type programs can be created to satisfy a particular need not addressed by Micromation or Digital Research. In addition, high level languages (e.g., BASIC, FORTRAN, COBOL) and application software to perform specific tasks (word processing, general ledger, accounts payable, etc.) can be purchased to complete the M/NET system.

**** IMPORTANT ****

The utilities provided on the distribution diskette are the only ones of their type that should be used. Do not use utilities from previous versions of CP/M. Although they appear similar in name, the programs provided with the M/NET system have been changed to accommodate the multi-processor environment. This does not apply to programs purchased separately (high-level languages and application software); it only applies to the utilities supplied on the distribution diskette.

****

NOTE: The following PRL type files are not discussed in this chapter since they are alternate forms of resident system processes (see Chapter 9).

SPOOL.PRL
ABORT.PRL
MPMSTAT.PRL
SCHED.PRL

The utility descriptions are separated into the following categories:

1) Disk(ette) and File Maintenance Utilities

FORMAT	ERA(N)	STAT	REN
DIR	COPY	PIP	

2) System Related Utilities

USER	DSKRESET	GENSYS	CONSOLE
MPMLOADER			

3) Other Utilities

TOD	TYPE	SUBMIT	TO
STOPSPLR			

4) Program Creation and Debugging Utilities

ED	ASM	LOAD	GENHEX
GENMOD	DDT	RDT	PRLCOM
DUMP			

In this chapter, the discussion of each utility and the examples are presented in a logical order. That is, some examples in the description of each utility are contingent upon the execution of the examples in the description of the previous utility. This is done to illustrate the interrelationship between these programs.

The procedures in previous chapters have

- transferred all the files from the distribution diskette to hard disk E:,
- formatted a diskette,
- made a back-up distribution diskette and
- generated a new MP/M system.

To proceed with this chapter, have a seat at console Ø and, if it isn't already fired up, turn on the computer system and boot MP/M from the copy of the distribution disk (in floppy drive A:, the left drive). Place an erasable diskette in drive B:. All the operations in this chapter are performed on hard disk drives except FORMAT. They all can be performed on the floppy disks though. If your M/NET system does not have a hard disk, substitute ØA> or A: whenever ØE> or E: is shown in an example and ØB> or B: for ØF> or F:. The utilities and examples have equal application to both floppy and hard disks.

The descriptions of the utilities include the symbols < and >. These brackets are used for illustration purposes only to distinguish the different segments of a command line. Never include the brackets when entering commands from the terminal. And remember that in all examples,

- enter only those characters that are underscored
- the @ character indicates that a RETURN should be entered

Once the system is booted, the loader message and MP/M prompt are displayed. If you have a hard disk drive, the prompt is ØE>; if you have just floppy drives the prompt is ØA>.

1) DISK(ETTE) AND FILE MAINTENANCE UTILITIES

This category describes those utilities needed to initialize a diskette or hard disk (prepare it to record data), transfer files from one disk(ette) to another, copy an entire disk, rename files, list the files on a diskette, and determine the amount of storage available on a disk(ette).

FORMAT.COM(PRL):

NOTE: This program is used on floppy diskettes ONLY. The hard disk drive does not require this procedure.

The format program is used to initialize floppy diskettes. "Initialize" is a fancy word for preparing the diskette so that data can be recorded on it. Before a diskette is written to, it must be formatted (initialized) first. If you are unfamiliar with diskette formats, refer to Appendix C at this time and return here when done. Like an automobile, however, you don't need to know how it works to run it.

**** IMPORTANT WARNING ****

Execution of FORMAT is dangerously simple. That is, this program requires relatively few decisions and entries by the user but can do a lot of damage. You see, FORMAT erases all the information on the diskette. This doesn't matter much if you are formatting a new diskette, fresh from the factory. But, if you format a used diskette that has information stored on it, the information will not be there when FORMAT is done. There is no way to recover files erased by FORMAT.

****

** See the M/NET addendum for a description of FORMAT **

DIR.PRL

The DIR utility is used to display the contents of a disk. If it appears familiar, you're right; it was used in Chapter 4 to display the contents of the distribution diskette. There are several ways to use DIR, each for a different purpose.

(Those familiar with CP/M will notice that this is no longer built-in command and, in fact, there are no built-in commands in MP/M because of the size of the system.)

In its simplest form

DIR@

shows the contents of the currently selected disk (the disk indicated by the prompt). The contents of a different disk can be shown by adding the disk name after the command,

```
DIR A:@
```

for instance. Notice that the disk reference includes a colon. In this case, the contents of the floppy diskette in drive A: is displayed.

In some circumstances, a listing of the entire disk(ette) is not desired. This is often the case when it has a lot of files most of which are irrelevant to the operator. Files can be selected from the directory using ambiguous file references. There are two special characters for this: * and ?.

An asterisk, "*", is used in the following manner. From the terminal enter

```
ØE>DIR *.PRL@
```

and the following is displayed

```
Directory for user Ø:
E: CONSOLE  PRL : DIR      PRL : DSKRESET PRL : ERAN     PRL
E: ERA      PRL : MPMSTAT  PRL : PRLCOM   PRL : REN      PRL
E: SUBMIT   PRL : TOD      PRL : TYPE     PRL : USER     PRL
E: SPOOL    PRL : STOPSPLR PRL : SCHED    PRL : ABORT    PRL
E: ED       PRL : PIP      PRL : STAT     PRL : RDT      PRL
E: ASM      PRL
```

(The exact listing of your disk may differ.)

As this example illustrates, * means all files.

The asterisk can also be used to display all files with the same file name. For instance, entering the following causes the subsequent display.

```
ØE>DIR ED.*@
Directory for user Ø:
E: ED       PRL : ED       COM
```

One other use of the asterisk may be useful: * may be preceded by 1 - 7 characters to further distinguish between files for display. For instance, enter

```
ØE>DIR S*.PRL@
```

and the following results

```
Directory for user Ø:
E: SUBMIT   PRL : SPOOL    PRL : STOPSPLR PRL : SCHED    PRL
E: STAT     PRL
```

The question mark is used much like an asterisk except it is used to replace one character for each ? entered. For instance, enter

```
OE>DIR ERA?.PRL@
```

and the following results

```
Directory for user Ø:
E: ERAN     PRL : ERA      PRL
```

Several ?s can be entered as in the next example.

```
ØE>DIR GEN???.COM@
Directory for user Ø:
E: GENMOD   COM : GENHEX   COM : GENSYS   COM
```

(The same would have resulted if DIR GEN*.COM was entered.)

If there are no files on a disk or the ambiguous file reference goes unsatisfied, a message is printed. Enter

```
ØE>DIR F:@
```

and this message is displayed.

```
Directory for user Ø:
File not found
```

PIP.COM (PRL)

Most frequently, PIP (Peripheral Interchange Program) is used to transfer files from one disk(ette) to another. This is not its only application, however. As the name indicates, the source or destination device is not limited to floppy or hard disk drives. (If it were limited to these two types of peripherals, it might have been FLIP or HIP instead.) Other types of peripherals accommodated are the console device (in this case, PIP can be used to display a file on the terminal instead of the TYPE command discussed later) and list device. (PIP can be used instead of SPOOL to output a file to the printer.)

Unlike the FORMAT utility, this program was written by Digital Research. A complete presentation of the various uses of PIP can be found in the booklet An Introduction to CP/M Features and Facilities with enhancements in CP/M 2.Ø User's Guide for CP/M 1.4 Owners. Only the more common uses relative to file transfer between disk(ettes) are discussed in this chapter.

There are two forms of the PIP command, each for a different purpose.

PIP <command line>@

or just

PIP@

The first form should be used to transfer a single file or a group of files. A single file is PIPed by replacing <command line> with the destination drive name, the source drive name, and the file name and type in that order. For instance, enter

ØE>PIP F:=E:ED.COM@

and a few seconds later the prompt appears. When the prompt re-appears, type

ØE>DIR F:@

and the following is displayed.

```
Directory for user Ø:
F: ED       COM
```

Regarding the syntax of the command line: notice that the destination drive for the file is entered first, then an equal sign, and finally the source drive name followed by the file name. It is very important that the order of destination and source drive names is understood. Moving files in the wrong direction can cause a lot of trouble.

NOTE: The destination can also be the console or list device instead of a disk drive. The syntax of the command is the same as illustrated except that "CON:" or "LST:" replace the destination drive (F: in the example above). When used like this, PIP transfers the file specified to the console (where it is displayed) or to the printer. This feature is only possible with files that contain ASCII characters (files that contains letters and numbers only).

The ambiguous file references described in DIR can be used with PIP also. To transfer all files of type COM from drive E: to drive F:, enter

ØE>PIP F:=E:*.COM@

When an ambiguous file reference is used, the name of the each file is displayed as it is transferred. (This is an important point when using the MP/M detach feature. Since PIP outputs to the console, it will not proceed when the console is detached. More on this in Chapter 1Ø.) The command above renders the following display.

```
COPYING -
GENMOD.COM
GENHEX.COM
LOAD.COM
PIP.COM
GENSYS.COM
ED.COM
DDT.COM
```

To see what you have wrought, type the underscored and the terminal will display

```
ØE>DIR F:@
```

```
Directory for user Ø:
F: GENMOD   COM : GENHEX   COM : LOAD     COM : PIP      COM
F: GENSYS   COM : ED       COM : DDT      COM
```

Notice that only one ED.COM is shown despite the fact that it has been transferred twice in the previous examples. When PIP encounters a file on the destination disk with the same name as the one being transferred, a temporary file, indicated by a $$$ file type, is created to store the data being transferred leaving the file already residing on the destination intact until the transfer is complete. For instance, if you could perform a directory of the drive F: (the destination drive) while ED.COM was being transferred the second time during the second example, two files with the name ED would appear there,

```
F: ED       $$$ : ED       COM
```

Upon completion of the transfer, the original is erased in favor of the file just moved. The file just moved is renamed (the file type is changed from $$$ to the proper type, in this case COM).

If at any time a file transferred is not allowed to complete, the incomplete file will be left with the $$$ file type.

The second form of PIP loads it into memory where it waits for the operator to enter a command line. PIP lets you know its ready to accept commands by displaying a prompt of its own, an asterisk. It looks like this.

```
ØE>PIP@
*
```

Once the PIP prompt has been displayed, you can enter a command line just like in previous PIP examples. The command line has the same syntax as in previous examples and can stipulate a single file or an ambiguous file reference. After the file has been moved, the prompt is re-displayed allowing

entry of another command line. If an ambiguous file reference is entered, the files are listed as they're transferred before the PIP prompt is displayed.

To exit this form of PIP, enter a RETURN. The MP/M prompt is subsequently displayed.

The example below shows a sample session in this mode. It picks up where the above example left off. Enter the underscored command lines one at a time as the PIP prompt is displayed.

```
(ØE>PIP@)
*F:=E:MPM.SYS@
*F:=E:*.SPR@
COPYING -
XDOS.SPR
BDOS.SPR
XIOS.SPR
*@

ØE>
```

Notice that the destination and source drives are entered for every command line. Just because you stated them the first time doesn't mean the computer's going to know you want the same thing in subsequent commands.

PIP can perform some special functions. The functions are described in the PIP section of An Introduction to CP/M features and Facilities and CP/M 2.Ø User's Guide for CP/M 1.4 Owners. Only one is discussed here, the verify feature. It may be worth your while to review the other features as well.

It is generally recommended that the PIP verify feature be used every time non-ASCII files are transferred. "What's ASCII?" you ask. ASCII files are those that contain letters and numbers only. Non-ASCII files are those that contain a program in object code; i.e., all files of types RSP, SPR, SYS, PRL and COM. ASCII files can be verified as well but experience has shown that problems don't usually occur when they're transferred.

To use the verify feature, append [V] to the end of the command line. For instance, the first example in this section where ED.COM was transferred should have been entered as

```
PIP F:=E:ED.COM[V]@
```

so that a verify would have been performed.

The verify feature reads the file from the destination disk after writing it and compares the data therein against that

on the source disk as it is transferred. Consequently, PIP with verify takes about twice as long as a PIP without.

Another special function of PIP is very important to M/NET operation. It allows the files to be transferred from one user area on a disk(ette) to another. This has little meaning now and won't until the user areas are described. This is done in the SYSTEM RELATED UTILITIES section in this chapter.

STAT.COM(PRL)

STAT does a lot of different things. Generally, these functions can be lumped into two categories.

- setting and determining disk status
- setting and determining file status and size

Setting and Determining Disk Status

STAT is perhaps most frequently used to check the amount of space available for file storage on a disk(ette). This should be done regularly to ensure that you don't run out of room during program execution (which, when it happens, usually occurs at the worst possible moment - Murphy's Law). This is the simplest invocation of STAT too. To see how much space is left on the active drives (active means that they have been accessed since the last cold-boot), enter

```
ØE>STAT@
```

and the following is displayed.

```
ØA: R/O, SPACE: 9ØK
ØB: R/W, SPACE: 483K
ØE: R/W, SPACE: 4637K
ØF: R/W, SPACE: 4928K
```

NOTE: The values that appear on your screen in this and ALL other examples using STAT are more than likely different. The numbers shown are only for illustration purposes.

The components of each line are:

Ø indicates the user number (see USER.PRL below)
A:, B:, E: and F: indicate the drive names
R/O indicates that the disk is read only (since the write protect notch is exposed)
R/W indicates that the disk is read and write enabled (since, in the case of the floppy, the write protect notch is covered)
SPACE: xxxxK indicates the amount of free space in kilobytes; that area on the disk(ette) that does not contain any files

Notice how much more space is present on the hard disk drives (ØE: and ØF:) than on the floppy disk drives (ØA: and ØB:).

The free space status for a specific drive can be determined by entering the drive name after the STAT command. For instance, to see the amount of free space on the floppy diskette in drive B:, enter

```
ØE>STAT B:@
```

and the message for drive B: only (shown above) is displayed.

Occasionally, the operator is concerned about the general characteristics of a disk(ette). This is primarily of interest when it is not known whether a floppy diskette is single or double density. For instance, although you know that the diskette in drive B: is double density (because that's how you FORMATed it), someone else might not. If they needed to know, entering

```
ØE>STAT B:DSK:@
```

would cause the following display.

```
   B: Drive Characteristics
3888: 128 Byte Record Capacity
 483: Kilobyte Drive Capacity
 128: 32  Byte Directory Entries
 128: Checked  Directory Entries
 256: Records/ Extent
  16: Records/ Block
  52: Sectors/ Track
   2: Reserved Tracks
```

The operator would know that this diskette is double density because 1) it has a 483 Kilobyte Drive Capacity (single density has about 241K bytes per diskette) or 2) it has 52 Sectors/ Track (single density has 26 sectors per track). The other parameters are not really relevant to this discussion. If you're curious, refer to the CP/M 2.Ø User's Guide for CP/M 1.4 Owners for a description.

The characteristics of all the active drives can be seen by entering

```
STAT DSK:@
```

An entire disk(ette) can be temporarily set to read only status (which means you can't write to it). It remains R/O until the next cold boot or warm reboot (DSKRESET command). Since cold- and warm-boots are not a casual affair in the M/NET system (because other operators will be very disturbed, maybe even peeved, when their program is terminated by a reset), it is suggested that this feature be used with great care, perhaps not used at all. The command is (don't enter it at this time)

```
STAT x:=R/O
```

where x indicates the drive to be set as read only. There is no command to set it as read/write after it's been set R/O.

There's one last feature of STAT vis-a-vis disk status. Entering

```
Ø>STAT USR:@
```

displays the list of the user numbers which have files on the current disk (E: in this case). The following display will result.

```
Active User : Ø
Active Files: Ø
```

This will have little meaning until you know about the USER command described in the section System Related Utilities later on in this chapter. Use of this form of STAT will be reviewed at that time.

Setting and Determining File Status and Size

It is sometimes necessary to know the size of a specific file or group of files on a disk(ette). This is frequently necessary when the amount of a free space is small and there is concern about running out of room. Also, various attributes can be assigned to files. For instance, one file on the diskette can be assigned as read only with STAT. This is a software protect not to be confused with the write protect notch on the diskette which is a hardware protect. (If this seems like Greek to you, disregard it. The difference is not significant.)

FILE SIZE: A single file or group of files can be interrogated with STAT to display the amount of space allocated. To find how much space a single file takes up, enter STAT and the file name. For instance, enter

```
Ø>STAT ED.PRL@
```

and the following is displayed.

```
Recs  Bytes  Ext  ACC
  6Ø    16k    1   R/W  E: ED.PRL
Bytes Remaining on E: 4637k
```

Recs: This category indicates the number of records (128 byte units or, simply, one record = one sector) allocated to the file.

Bytes: The number of kilobytes allocated to the file is indicated in this column. The value is always expressed in thousand byte units. If a file takes up 768Ø bytes (as does ED; i.e., 6Ø x 128 = 768Ø) for instance, the amount is rounded up to the nearest thousand. All files on the hard

disk, however, are automatically allocated 16k bytes; even if the file contains but one character. If you had looked at the ED.PRL file on the diskette in floppy drive A:, the Bytes column would show 8k.

Ext: This column indicates the number of extents used up by the file. (See the Digital Research manual for an explanation of extents.)

ACC: The status of the file, whether it's read only or read/write, is indicated here (ACC refers to access mode). R/W means that the file has read/write status; it can be read from, changed and written to. R/O indicates the file is read only; the file cannot be changed or written into.

The letter preceding the name indicates the drive it's on. Even though you didn't ask, the amount of space left on the disk is displayed at the end.

Use of an ambiguous file reference (using * and/or ?) allows the operator to see the same characteristics for a group of files. For instance, enter

```
ØE>STAT A:*.COM@
```

This example includes one other feature of STAT, the ability to specify a drive other than the current one. Above, E: is the currently logged (hard disk) drive and the preceding A: before the ambiguous file reference is the (floppy disk) drive being interrogated. The following display results.

```
Recs  Bytes  Ext  ACC
  52     7k    1   R/W A: ED.COM
  42     6k    1   R/W A: DDT.COM
   6     2k    1   R/W A: GENHEX.COM
  1Ø     2k    1   R/W A: GENMOD.COM
  6Ø     8k    1   R/W A: GENSYS.COm
  14     2k    1   R/W A: LOAD.COM
  5Ø     8k    1   R/W A: MPMLDR.COM
Bytes Remaining of A: 9Øk
```

Notice that the files are alphabetized by STAT.

The categories have the same meaning here as in the previous example. Since you looked at the diskette in floppy disk drive A:, notice that each file is not automatically allocated 16k. On floppy disks, the minimum size is 2k bytes. Hence, although GENHEX, GENMOD, and LOAD each take up 2k bytes, the actual size is more accurately reflected by the number of records (128 x Recs).

FILE STATUS: In addition to read only versus read/write

status, a file may be assigned the "system" attribute. This means that the file will not appear when a DIRectory is performed. The logic behind this is, there's no need to clutter up the DIRectory with files not relevant to the application of the disk. For example, an operator using a floppy for general ledger does not need to know that the diskette also contains the SPR files. This may be helpful when there are a lot of files on a diskette but the operator is only concerned with a handful. Assigning R/W or R/O status is described first, then assigning "system" status is discussed.

To review,

R/O (read only) means that a file can only be read; it cannot be changed either by editing or adding to it nor erased.

R/W (read/write) means that a file can be read from, altered, and rewritten to the disk(ette) or erased.

Also recall that these attributes are assigned by STAT and have nothing to do with the write protect notch on the diskette.

Notice that all the files in previous STAT examples have R/W status. To change the status of a file, the same form of STAT is used as in previous examples but $R/W or $R/O is appended to the end of the command line. For instance, the next example changes the status of the MPM.SYS on drive F: from R/W to R/O.

```
ØE>STAT F:MPM.SYS $R/O@

MPM.SYS set to R/O
ØE>
```

As with all other applications of STAT, an ambiguous file reference can be used to change a group of files. For example, enter

```
ØE>STAT F:*.SPR $R/O@

BDOS.SPR set to R/O
XDOS.SPR set to R/O
XIOS.SPR set to R/O
ØE>
```

To see what you have wrought, enter

```
ØE>STAT F:*.*@
```

```
Recs  Bytes  Ext  ACC
  47    16k   1   R/O F: BDOS.SPR
  52    16k   1   R/W F: ED.COM
  42    16k   1   R/W F: DDT.COM
   6    16k   1   R/W F: GENHEX.COM
  10    16k   1   R/W F: GENMOD.COM
  60    16k   1   R/W F: GENSYS.COm
  14    16k   1   R/W F: LOAD.COM
  50    16k   1   R/W F: MPMLDR.COM
 148    20k   1   R/O F: MPM.SYS
  72    16k   1   R/O F: XDOS.SPR
  11    16k   1   R/O F: XIOS.SPR
Bytes Remaining of F: 4928k
```

"*.*" in the command line says to the system, "All files please."

Just for the fun of it, let's assign the system attribute to the SPR files. The syntax for this is the same as assigning R/O status except $SYS replaces $R/O. Enter

```
ØE>STAT F:*.SPR $SYS@

BDOS ERR ON F: READ ONLY
```

Oops, these files are read only; no characteristic can be changed. A minor problem easily rectified. To change the files back to R/W status enter

```
ØE>STAT F:*.* $R/W
```

The files are listed as they are changed. Those files that are already R/W are not shown.

Now you can change the "system" status. Enter again

```
ØE>STAT F:*.SPR $SYS@

BDOS.SPR set to SYS
XDOS.SPR set to SYS
XIOS.SPR set to SYS
ØE>
```

The files are displayed as they are changed.

These files will no longer appear in a DIRectory of disk F:. (Try it out, type DIR F:.) The only way you will know that these files are present on the disk(ette) is by using STAT to see file space allocations. Enter

```
ØE>STAT F:*.*@
```

```
Recs  Bytes  Ext  ACC
  47    16k   1   R/W F: (BDOS.SPR)
  52    16k   1   R/W F: ED.COM
  42    16k   1   R/W F: DDT.COM
   6    16k   1   R/W F: GENHEX.COM
  10    16k   1   R/W F: GENMOD.COM
  60    16k   1   R/W F: GENSYS.COm
  14    16k   1   R/W F: LOAD.COM
  50    16k   1   R/W F: MPMLDR.COM
 148    20k   1   R/O F: MPM.SYS
  72    16k   1   R/W F: (XDOS.SPR)
  11    16k   1   R/W F: (XIOS.SPR)
Bytes Remaining of F: 4928k
```

The fact that a file has the system attribute is indicated by the enclosing parentheses, (BDOS.SPR) for instance.

A single file can be given the system attribute. In this case, replace the ambiguous file reference with the file name. For instance

```
STAT F:MPM.SYS $SYS
```

NOTE: The SYS file type of this file has nothing to do with the system attribute.

Once a file has been lent this feature it can be changed back with the following command

```
STAT <filename> $DIR
```

For example, to change the SPR files' status to the original, enter

```
ØE>STAT F:*.SPR $DIR@
```

The files are listed as they are changed back.

The Digital Research manuals, An Introduction of CP/M Features and Facilities and CP/M 2.Ø User's Guide for CP/M 1.4 Owners contain more information on this utility. Note, however, that all references to List Device Assignments do not apply with MP/M.

COPY.COM(PRL)

See the M/NET Addendum for a description of this utility.

HDCOPY.COM(PRL)

See the M/NET Addendum for a description of this utility.

The utilities described thus far will probably be the most frequently used. They are also perhaps the most complex to understand. Until you are familiar with their form and operation, we recommend referring to them whenever they are used. This is especially true of FORMAT which can do real damage if a mistake is made. (You don't want to say to yourself, "Oh my, I've just erased a year of our programmer's work!! I hope he'll understand.")

REN.PRL

Occasionally, it is necessary to change the name of a file. REName provides this facility. The general form of a REName command is

REN <new file name>=<old file name>

As with PIP, it is important to understand the syntax of this command. Notice that the new file name is entered first, then the equal sign, and finally the old file name.

As an illustration of REN's use let's say a hot-shot programmer has written a new PRL type file that does all sorts of wonderful things. To indicate just how wonderful it is, this programmer named the file ZPBQHGXM.PRL (each letter of the name indicates a different function). Now this is a very difficult name to remember and enter as the operator for whom the program was written soon found out. (Programmers sometimes overlook the human interface.) Simple solution: rename it.

REN DOIT.PRL=ZPBQHGXM.PRL@

This does not change any of the operating characteristics of the program, only the name.

Another typical use of rename is to prevent the erasure of old versions of programs by PIP when a new version, with the same name, is transferred. This is a frequent occurrence on back-up diskettes where many versions of a program are kept. (Recall that PIP erases the file on the destination disk(ette) when a file of the same name is moved there.) By renaming the file on the destination disk, the old version is not erased.

REN also has an important application with the EDitor. Refer to the booklet, ED: A Context Editor for the CP/M Disk System for a description.

Unlike most of the other utilities, REN does not accept ambiguous file references. Also, fortunately, it does not allow you to rename a file to the name of a file that already exists on the disk(ette). An error message is displayed if this is attempted.

ERA.PRL, ERAN.PRL

These two utilities erase files from the disk(ette). One, ERA, prompts the operator; the other, ERAN, does not. (ERA is called ERAQ in the MP/M documentation.)

As with other utilities, ambiguous file references are allowed. This is a double-edged feature, though, so be careful. For instance, ERAN *.* is a very simple and convenient way to erase all the files on the disk(ette), whether that is your intention or not.

To illustrate their use, first enter

```
ØE>F:@
ØF>
```

This little procedure changes the logged disk. Here it is done to prevent accidently erasing the files on ØE:. The ØF> prompt indicates the change. Follow this command with

```
ØF>DIR@

F: GENMOD  COM : GENHEX  COM : LOAD  COM : PIP  COM
F: GENSYS  COM : ED      COM : DDT   COM : MPM  SYS
F: XDOS    SPR : BDOS    SPR : XIOS  SPR
```

These files, with a few others perhaps, have been put here by the previous examples in this chapter. Let's erase a few. For instance, to erase one file enter

```
ØF>ERA GENHEX.COM
F: GENHEX  COM ? Y
ØF>
```

As you can see, ERA provides an opportunity to change your mind. ERAN does not. For instance, enter

```
ØF>ERAN *.SPR

ØF>
```

This command has just erased all the files of type SPR. Perform a DIRectory (DIR@) at this time to see the files that remain; compare the display with the original DIRectory immediately above.

If you are going to use ambiguous file references as in the previous example, be advised to use ERA instead of ERAN. Once a file has been erased it is almost impossible to get it back; so much so that you might as well consider it impossible. It's a terrible feeling to realize that a file you wanted to save just got grouped in the batch described by an ambiguous file reference. To illustrate, enter

```
ØF>ERA *.PRL

NO FILE
ØF>
```

Oh, that's right; there's no PRL files on this disk. Instead, enter

```
ØF>ERA *.COM
F: GENMOD   COM ? Y
F: LOAD     COM ? Y
F: PIP      COM ? N
F: GENSYS   COM ? Y
F: ED       COM ? N
F: DDT      COM ? Y
ØF>
```

A DIRectory of disk F: now shows just two files: PIP.COM and ED.COM.

2) SYSTEM RELATED UTILITIES

USER.PRL

This utility allows the files on a disk(ette) to divided into different groups or sections; each identified by a different USER prefix number. Recall from Chapter 3 the description of the MP/M prompt, ØE> (or ØA> if there's no hard disk in the system). It was stated that the number indicates the console number; Ø indicates console Ø, a person at console 2 would have seen 2E>, etc. This is the default assignment when the system is cold booted. Each console always has a unique number assigned to it, but this number has limited relevance in the M/NET system. Until the importance of console numbers is significant (it will be pointed out when it is), think of the number as an indicator of the current user group of files. The USER utility is used to change from one section to another.

Unfortunately, this causes some confusion in terms: user as in operator versus user as in user area of the disk(ette). To prevent ambiguity, all references to user areas will be indicated by USER.

A feature of MP/M (and CP/M version 2) is the ability to attach a USER label to each file. That is, besides the file name and type, a number in the range of Ø - 15 is also used to identify a file. All files on the distribution are assigned USER number Ø. To create files with a different USER number, the PIP utility is used to duplicate the file(s) ad, in the process, attach the new USER group number. The result is a file or set of files that are only accessible when the operator switches to that group.

To change USER numbers (in this case to 9 but any number between Ø - 15 is possible), enter the following underscored commands (wait for the prompt before entering the second)

```
ØF>USER 9@

User number =  9
9F>DIR@
Directory for User  9:
File not found
9F>
```

Notice that the prompt indicates the current USER number and that the logged disk reference is the same. The DIR command shows that no files are currently allocated to USER 9. To move files into this area, enter

```
9F>PIP F:=F:*.*[GØ]@
```

[GØ] is the special function of PIP mentioned briefly in that portion of this chapter. It means, "Get from USER Ø." Notice that the destination and source drives in the command are the same. Only two files are copied, ED.COM and PIP.COM, since there were only two files left on drive F: after the ERA and ERAN exercises. To get a few more from drive E: this time, enter

```
9F>PIP F:=E:*.PRL[GØ]@
```

Again, USER Ø must be specified because there's no USER 9 on disk E:. That is, once the operator has changed USER numbers, MP/M looks at only those files belonging to that USER, regardless of the disk reference. For instance, enter

```
9F>DIR A:@
Directory for user 9:
File not found
9F>
```

This command caused the computer to look at floppy disk A: USER area 9 where, since nothing has been put there, nothing was found.

If you're wondering, "If there were no files in USER area 9 when I entered it, how come PIP and DIR worked?" you get a gold star. That's a very good question which brings up an important feature of M/NET.

**** IMPORTANT ****

If M/NET cannot find the program requested in the USER area and logged disk(ette) indicated by the MP/M prompt, it looks to the USER Ø section of that drive for it. If the program is not there, MP/M then looks to drive E: (drive A: in floppy only systems) for the it. If it finds the program there, the command is executed; otherwise an error message.

****

Thus, a distinction is made by M/NET between the program file and the object file(s). The program file, identified by the first name of the command line, can be derived from USER/drive ØE: (ØA), but the object file(s), identified by the remainder of the command line, must be in the USER section stipulated and must be resident on the same physical disk(ette) indicated by the MP/M prompt. For instance, "PIP" is the program file in the command above while "*.PRL" (all files with type PRL) are the object files. These are also referred to as the command and the command tail in the MP/M documentation.

USER file groups are also used for a limited form of file protect. Once a USER area has been created, every operator

has access to it and the files therein regardless of their console number. For instance, although Mary (Mary who works for the window latch manufacturer) is at terminal Ø, she has access to all the files in section 9F:. Once she enters USER area 9 and begins execution of a program therein, no one else (Hank for instance) can join her until she has vacated it.

This feature prevents two operators from working on the same file at the same time, a potentially dangerous possibility. To illustrate how dangerous this is consider the following. A man and his wife have credit cards with the same account number and there's but $6Ø left before their limit is exceeded. They both go shopping, he to get some clothes, she to pick up a new tennis racquet. The pants selected cost $49.95, the racquet $35. As it turns out, both arrive at the sales counter to make the purchase at the same time and the clerks call the credit bureau simultaneously. If the master computer that handles their account does not exclude others from the file while it is being updated, both clerks would find that there's enough left in the account and allow the purchases. They've exceeded their limit by $25.Ø5. To prevent this occurrence, as soon as one clerk opens an account file (to check how much is left and subtract the cost from the balance), the other clerk has to wait until the first has finished before s/he can get into it.

The reason for setting up different USER areas is now apparent. Each application program used with M/NET should have a separate USER area assigned to it to prevent two operators from accessing the same file(s) at the same time. For instance, USER area 2 on disk E: has the general ledger programs and data files; area 3 has the accounts payable programs and data files; area 4 has the word processing files; and area 8 has all the computer games. Every operator can access any of these areas but they can only do it one at a time. (Sorry, only one operator playing games at a time.) Chapter 1Ø describes how to set up your own USER area.

Since it isn't always possible to keep track of all the USER areas on the disk, STAT has a feature that indicates them. Enter

```
9F>STAT USR:@

Active User : 9
Active Files: Ø 9
9F>
```

There's no need to create USER areas in any sequence. The active files could be Ø and 15; Ø, 9, 1Ø, 11, and 14; etc.

DSKRESET.PRL

Changing diskettes in the M/NET system can invite all sorts of problems. For instance, Mary is at console Ø, next to the floppy drives, and decides that she wants to put a different diskette in floppy drive B:. Blithely, she checks to ensure that the active light isn't on, finds it isn't, opens the door, pulls out the diskette, puts in another and closes the door. Meanwhile in another room, Hank finds his terminal has gone into limbo; all traces of his program have disappeared.

What happened? Although Hank's program wasn't accessing drive B: when Mary checked the active light, it was using that diskette. (This is standard procedure with most disk-based programs. Once they are loaded into memory they run without accessing the disk until some data is needed.) When the program called for a disk access, the files it had opened and expected to be there were gone. Crash. To prevent this situation, the DSKRESET utility is used.

Changing a diskette is a two-part process using the same command. Before removing a diskette from a drive, enter

DSKRESET x:@

where "x:" indicates the drive affected.

If there is an open file there, the reset is not performed and the following message is displayed.

Disk reset denied, Drive x: Console c Program pp

where "x" is the drive specified in the command and "c" is the console number (not USER number) of the operator using program "pp" that has opened a file. It is not safe to change the diskette at this time.

If this message is not displayed, remove the diskette, replace it and type

DSKRESET x:@

again.

The DSKRESET utility must be used every time a diskette is changed. This applies to floppy diskettes only. The hard disk drives do not need to be reset since they cannot be physically removed.

CONSOLE.PRL

If at any time you are confused as to what console you are sitting at and need to know enter

CONSOLE@

and the following display results

```
Console = x
```

where "x" is your console number. This number is always the same for a specific console regardless of the current USER number.

Note that the first console in any system is labelled console Ø. In a four user system, the consoles would be labelled Ø 3.

This number is significnat because it is used to indicate which console a program is attached to. For instance, if Hank at console 1 moves to USER section 9 diskette B: to run a program and Mary at console Ø simultaneously tries a DSKRESET of that drive, the message would read

```
Disk reset denied, Drive B: Console 1, Program xx
```

Notice that the console number (1), not the USER number (9) was displayed. This is true of other MP/M utilities that indicate or require the program to console assignments (ABORT and MPMSTAT for instance).

GENSYS.COM(PRL)

The GENSYS utility is used to create a new MP/M system. The system is only minimally affected (in fact, you may not notice the difference) if the hardware configuration does not match the operating system. For instance, the MP/M system delivered more than likely did not agree with the M/NET hardware characteristics (e.g., number of terminals, number of slave processors). However, the system ran.

It is good form to have the O/S match the hardware, though. This is one reason to run GENSYS: to (re-)configure MP/M to reconcile it with the specific number of system components.

Another reason to run GENSYS is to make changes in the resident system processes (RSPs) and/or the memory segments. Your experiences with the system may show that you don't need the SPOOL, SCHED, MPMSTAT, or ABORT RSPs (MNET.RSP must always be included in the system), or that you need fewer larger or more smaller memory segments than those stipulated in Chapter 5.

Rather than run through a GENSYS operation again here, use Chapter 5 if you need to change the MP/M system. Before you do, you may want to read through the discussion of the RSPs in Chapter 9 and the description of memory segments in Chapter 7 if their use and meaning is not apparent.

3) OTHER UTILITIES

This section precedes the description of the utilities for program creation because, chances are, they will be used by more operators. Their use is not crucial to system operation, but they do perform some interesting operations.

TOD.PRL

MP/M contains a real time clock. This program utilizes this feature to display the Time Of Day. For instance, enter

```
9F>TOD P@

Fri Ø3/14/8Ø 9:ØØ:ØØ
```

Your display will not show this date and time unless you performed a cold boot immediately before making the command. This is the default from which the time increments with every passing second after a reset. To get the MP/M prompt back so you can make another command, enter any character on the keyboard (space, RETURN, J, etc.).

The command to change from the default is

```
TOD mm/dd/yy hh:mm:ss
```

For instance, Hank is working late one night and needs to reset the system. He knows there's no one else on the system so it's safe to reset it. The day is July 2, 198Ø and it's 1Ø:ØØ pm. He enters

```
TOD 7/2/8Ø 22:ØØ:ØØ@
```

Notice that the TOD operates on a 24 hour clock basis and that months and days that can be indicated by a single digit do not require a preceding Ø. The following display results.

```
Strike key to set time
```

Hank hits any key (it doesn't matter which one) and

```
Tue Ø5/Ø1/8Ø 22:ØØ:ØØ
```

followed by the MP/M prompt is displayed. Subsequently, the time to the second is displayed when TOD is entered. Addition of "P" after TOD causes the display to update with each tick.

If the scheduler RSP, SCHED, is going to be used, the proper time must be entered for the program to be executed at the requested time. (This resident system process allows the operator to execute a program at a time of their choosing. It is described in Chapter 9.) SCHED references the time of day set by TOD.

You should set the time of day for your computer at this time.

TYPE.PRL

This utility allows the operator to display the contents of any ASCII file. ASCII files are those that contain letters and numbers. This includes files of type ASM and PRN and any created with the editor (ED) or other word processor. Files that cannot be TYPEd are those with file types COM, HEX, PRL, RSP, SYS, SPR or others that contain object code. This is a handy utility for viewing a file for quick reference.

The form of this command is

TYPE <file name>.<type>

Ambiguous file references are not allowed; only one file can be specified. To illustrate, enter

```
9F>USER Ø@
User number = Ø
ØF>TYPE A:DUMP.ASM@
```

The first step was necessary to get back to USER number Ø since DUMP.ASM is only assigned to this USER. The file on disk A: called DUMP.ASM appears on the screen. When the entire file has been displayed, the prompt is output. The output can be terminated by hitting any character on the keyboard. This, too, returns the prompt.

(DUMP.ASM is the source program of one of the Digital Research utilities described in the next section of this chapter. The contents of this file are an example of a program written in 8Ø8Ø assembly language.)

Two of the special console command characters described in Chapter 6 complement TYPE; CTRL-S (^S) and CTRL-P (^P). (Recall that these are entered by pressing the key labelled CTRL and either S or P simultaneously.)

^S temporarily stops the display from scrolling until another key is pressed. When a long file is displayed, the contents pass by on the terminal rather quickly, too quickly to be read. To stop the scrolling, enter a ^S. If you want to try this out, TYPE out DUMP.ASM again and enter a ^S after a couple of lines have been displayed. Enter another character (e.g., another ^S) and the display will continue until the end of the file or another ^S is entered. There's no limit to the number of ^Ss that can be entered so you can stop and start the display as many times as you want.

^P causes the output to go to the printer and the screen at the same time. When the file is done, enter another ^P. Note that ^P causes all console input and output to go to the printer. Consequently, unless another ^P is entered after the file has been printed, all subsequent entries from and outputs to the terminal will also be printed.

^S and ^P are toggle functions. That is, the first invocation causes a response (either stopping the display or start printing) and the second stops it (return to scrolling or no output to the printer).

If any character is entered during the display/print, all output is terminated and the MP/M prompt is displayed.

SUBMIT.COM(PRL)

A limited form of batch processing is provided with SUBMIT. "Batch processing" is the ability to string several programs together so that one is performed after the previous one has finished. This is done in MP/M by creating a file with the EDitor (or other word processor) that contains the string of commands you wish to have executed. (The file MUST have a SUB file type.) This is especially convenient if there are a sequence of commands that are performed frequently. In its simplest form, the command is

SUBMIT <file name>

Since only files of type SUB are recognized by SUBMIT, you can leave off .SUB from the file name; it is assumed.

The manual, An Introduction to CP/M Features and Facilities has a thorough discussion of the SUBMIT utility. In addition, you may want to read through ED: A Context Editor for the CP/M Disk System for instructions on creating files for use with SUBMIT.

TO.PRL

TO is a hand little utility for sending messages from one console to another. The form of the command is

TO c <message>

where "c" is the number of the destination console (not the USER number) and <message> is a string of up to 128 letters and/or numbers. After the RETURN key is struck

Message sent

and the MP/M prompt are displayed.

Messages can only be sent when the MP/M prompt is displayed. They can be received at any time, however; even when the destination console is in the middle of a program run. Receipt of a message does not affect program execution.

STOPSPLR.PRL

There is but one application of this utility: to stop the SPOOLer (SPOOL is a resident system process described in the next chapter) when it's printing a file. If you are the source of the print command (i.e., you entered the SPOOL <file name> command), enter

STOPSPLR

and the printer will stop. You can also terminate the SPOOLer from a different console. For instance, Hank at console 1 needs a program listing so he gets SPOOL to print it. Meanwhile, Mary finds out that the letter she wrote for her boss is needed right away. "Sorry Hank," Mary thinks as she enters

STOPSPLR 1

to terminate Hank's listing and proceeds to SPOOL her file. Notice that this command references the console number rather than a USER number.

4) PROGRAM DEVELOPMENT UTILITIES

The utilities listed in this section have little application beyond writing and debugging 8080 assembly language programs with the possible exception of the first, ED. The editor supplied with MP/M can be used as a word processor. However, it is not well-suited for this application; there are many word processing programs available from your dealer better-suited to letter or manual writing.

The first five utilities in this section are operationally similar to those supplied with CP/M. They are described in the CP/M manuals supplied with the M/NET system (the specific manual is identified in the description below). Note that these manuals include programs created for use with CP/M that are not applicable under MP/M. Only those identified in this chapter and present on the distribution diskette should be used in the M/NET system. The remainder of the utilities were created for use with MP/M. Their description is in the MP/M User's Guide.

ED.COM(PRL)

This is a brief description of the MP/M editor. Refer to ED: A Context Editor for the CP/M Disk System for a complete description of the features and examples in its use. (ED was originally provided with the CP/M operating system but has equal application with MP/M.)

ED is used to enter and edit ASCII files. The command to invoke the editor is

ED <file name>

or

ED <file name>.<file type>

Notice that the file name (and optional file type) must be specified and ambiguous file references are not allowed.

There are two modes of operation in ED: command and insert. Under the command mode, special characters are available for moving around in the file (from the beginning of the file to the end or vice versa, move down or up one line or many lines, etc.), displaying one or more lines, deleting characters or lines, searching and/or replacing strings (groups of characters like a word or phrase). When in the insert mode, all entries from the keyboard go directly into the file.

When an editing session is terminated, ED leaves two files on the disk(ette): a new file containing the original with the changes and the original renamed to type BAK. For instance, if file DOIT.ASM was the source file, the new, edited version

resides on the disk(ette) as DOIT.ASM and the original source file is renamed DOIT.BAK. This is handy if it is decided that the changes were erroneous; the new file can be erased and the BAK file renamed to the original.

ASM.COM(PRL)

This is the MP/M 8080 assembly language assembly utility. It takes a source program (which must have a file type of ASM) and returns a 8080 machine language file in the Intel hex format. A second file can be generated containing the source program lines preceded with the hex equivalent. These files have the same file name as the source file but have file types of HEX or PRN, respectively. The manual, CP/M Assembler (ASM) User's Guide, should be referenced for a complete description of this utility.

LOAD.COM(PRL)

LOAD takes a HEX file (i.e., one generated by ASM) and renders a COM file. This, along with ASM, is one way users can create their own utilities. A description of its use appears in An Introduction to CP/M Features and Facilities.

DDT.COM

The Dynamic Debugging Tool (DDT) allows programmers to run and test assembly language programs for the purpose of debugging. DDT is a remarkably flexible program that allows the user to edit changes into the program being debugged while it's running. Inherent commands allow assembly, display, trace, move, read, and set functions during execution. Refer to the CP/M Dynamic Debugging Tool (DDT) User's Guide for a complete description of DDT and its use.

NOTE: RST 7 MUST be used to return control to DDT if a program breakpoint is not encountered. The MP/M manual states that others are possible but this is not true in the M/NET system.

DUMP.COM(PRL)

DUMP is used to display the contents of a file on the terminal in hexadecimal form. Each line of the display, 16 bytes per line, is preceded by the absolute byte address of that line, also in hexadecimal. The form of this command is

DUMP <file name>.<file type>

where the file type is optional.

The remainder of the utilities in this chapter were developed for use with MP/M. Their use is described in the MP/M User's Guide.

RDT.PRL

This is the page relocatable version of DDT. It contains all the features of DDT but can be run in any of the memory segments of the master processor large enough to accommodate it. The decision regarding which to use is based on this factor only. That is, if you want to run a debug program in one of the slave processors (assuming one is available), use DDT.COM; if you want to use one of the master processor memory segments, use RDT.PRL.

PRLCOM.PRL

This utility is used to make COM files from PRL files. It does not work in the other direction, however; i.e., you can't make page relocatable programs (PRL) from COM files using this utility. (The MP/M User's Guide contains instructions to do this.)

GENHEX.COM

GENHEX reverses the process of ASM; it generates HEX files from a file of type COM. This is used primarily to prepare files for the GENMOD utility.

GENMOD.COM

GENMOD produces a file of type PRL from two concatenated HEX files (as created GENHEX). Note that the HEX files must be offset by 100H bytes.

CHAPTER 9: RESIDENT SYSTEM PROCESSES

All files of type RSP are programs that may or may not be included in the operating system when GENSYS is run. There is one exception, MNET.RSP. This program **MUST** be included for the system to operate. There is a trade-off if you want to include all the RSPs, however: each RSP selected decreases the amount of memory space in the master processor available for program execution. As each RSP is added, it is appended to the bottom of the system area. Figure II:9.1 illustrates the locations.

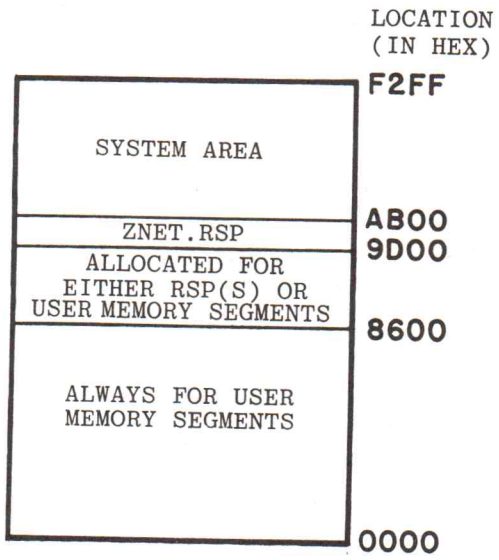


FIGURE II: 9.1 MASTER PROCESSOR MEMORY STRUCTURE SHOWING RSP AREA

If none of the RSPs is selected after MNET in the GENSYS process, 1700H bytes of memory are added to the master processor, user memory area. Each RSP selected subtracts the amount indicated in the table below.

RSP	Amount of Memory Lost
ABORT	0100H
MPMSTAT	0D00H
SCHED	0500H
SPOOL	0400H

If ABORT, MPMSTAT or SPOOL is invoked from the console but it was not included in the system with GENSYS, it will execute as a PRL type file. This is because these RSPs are duplicated as PRL type files on the distribution diskette. However, it will take longer to run. The difference in time is because the program must be read from the disk and loaded into a memory segment (assuming one is available) before it can be executed. If it was included with

GENSYS, the RSP would execute immediately from system memory. SCHED is a special exception. See the SCHED discussion of the reason.

And there's one last consideration: If a particularly earnest worker has tied up all the memory segments, there is no place for the program to run if it has not been included in the system. (In this instance, the PRL form of the program would execute.) For instance, Mary, a particularly earnest worker, has tied up all the slaves and memory segments with various programs. Hank upon receiving the "Reloc seg not free" message, enters MPMSTAT (the RSP type file which displays system status) to see who has control of the memory segments. However, MPMSTAT.RSP was not included in their system when GENSYS was performed; consequently, there's no place for MPMSTAT.PRL to run. Hank, again, receives the "Reloc seg not free" message. He will get that message until one of Mary's PRL programs ends.

As you can see, there are several factors to weigh in the decision regarding RSP inclusion:

- How often am I going to use this utility?

- Is it worth the loss of memory space for program execution?

- Can I wait the extra second or two it takes to read the file from the disk?

- Do I want to be able to run one (or all) of these programs when the system is otherwise "booked up."

Only your application and experience can answer these questions. The system generated in Chapter 5 does not consider any of these questions and should be used only until your experience dictates that another configuration would be more appropriate.

The remainder of this chapter describes the resident system processes. Only the first described must be included in the MP/M system; the remainder are optional.

MNET.RSP: This part of the operating system controls the allocation of the slave processors. In fact, without it the system doesn't know that slaves exist. There are no operator commands that use this RSP; it is only for the system's use.

MPMSTAT.RSP (and PRL): When invoked, this program displays the status of the MP/M components. Many of the processes and categories described are irrelevant to daily operation. They have meaning to system builders employing M/NET in special applications and those intent on knowing every facet of M/NET and MP/M operation. This chapter will not dwell on every detail of the MPMSTAT display; it addresses only those parts that have significance to less experienced operators. Refer to the MP/M User's Guide and Chapter 12 of this manual for a complete description.

There is only one form for MPMSTAT invocation. It is

```
ØF>MPMSTAT@
```

The reproduction of the MPMSTAT display that follows does not necessarily reflect your system configuration. Your display may differ in some respects.

```
****** MP/M 1.1 Status Display ******

Top of memory = F2FFH
Number of consoles = Ø4
Debugger breakpoint restart # = Ø7
Stack is swapped on BDOS calls
Z8Ø complementary registers managed by dispatcher

Ready Process(es):
  MPMSTAT  Idle
Process(es) DQing:
 [ABORT   ] ABORT
 [CliQ    ] cli
 [ATTACH  ] ATTACH
Process(es) NQing:
Delayed Process(es):
Polling Process(es):
Process(es) Flag Waiting:
  Ø1 - Tick
  Ø2 - Clock
  Ø4 - Tmp1
  Ø5 - Tmp2
  Ø6 - Tmp3
Flag(s) Set:
  Ø3
Queue(s):
  MLoader  ABORT    MPMSTAT  CliQ    ATTACH  MXParse  MXList
  MXDisk
Process(es) Attached to Consoles:
  [Ø] - MPMSTAT
  [1] - Tmp1
  [2] - Tmp2
  [3] - Tmp3
Process(es) Waiting for Consoles:
  [Ø] - TmpØ
  [1] - Tmp1
  [2] - Tmp2
  [3] - Tmp3
Memory Allocation
  Base = ØØØØH  Size = 41ØØH  * Free *
  Base = 41ØØH  Size = 2DØØH  * Free *
  Base = 6EØØH  Size = 11ØØH  * Free *
  Base = 7FØØH  Size = 2CØØH  * Free *
```

This display assumes a 4 console system with no active users.

The categories of primary concern are the last three,

Process(es) Attached to Consoles
Process(es) Waiting for Consoles
Memory Allocation

They show

- what job is currently active at each console
- what jobs are waiting for the associated console to be free (for either program output or user input) and
- how and to whom the memory segments are allocated

respectively. For instance, the MPMSTAT display above shows MPMSTAT active on console Ø, the terminal message processor (Tmp) waiting until MPMSTAT finishes, and all the memory segments free.

The Tmp (Terminal Message Processor) is that component of MP/M that handles user inputs from the consoles. Each console has a Tmp assigned to it. After the Tmp has processed a message (a command line),for instance, it passes control of the console to another component of the system (for those who care, the command line interpreter or Cli) and idles until the command has been executed and that program has finished. Consequently, it is always the first item waiting for the console while another program is being executed (like MPMSTAT in the sample display above).

The next example illustrates what the MPMSTAT display looks like when all the memory segments are tied up. The first steps shown fill all the memory segments. This is their only purpose (that is, you wouldn't perform these steps in normal operation). The memory segments correspond to the number and size of those created in the GENSYS example in Chapter 5. If your system is different, you can enter the same commands but the system's response may be different.

It is expected that you are the only user on the system at this time. If that is not the case, the following procedures will not interrupt other user's operations. However, some memory segments may already be filled by their programs. If this is the case, you will get the "Reloc seg not free" message earlier than shown below.

Use of the program detach feature is also illustrated. The special character CTRL-D (shown as ^D in the steps) is used to separate the console from a running program so that another command can be entered. Any program that either outputs to the console or accepts console input can be detached (most but not all programs do this). Enter the ^D in the following steps right after entering the RETURN. The timing is not crucial, though.

To begin, enter

```
ØE>MPMSTAT@
```

and note the status of the three categories of significance. If any are in use, proceed regardless.

Next, enter

```
ØF>DIR@^D
ØF>
```

The ØF> prompt may take a second or so to appear after the ^D was entered. Enter the MPMSTAT command again and the three categories should appear as follows.

```
Process(es) Attached to Consoles:
  [Ø] - MPMSTAT
  [1] - Tmp1
  [2] - Tmp2
  [3] - Tmp3
Process(es) Waiting for Consoles:
  [Ø] - TmpØ  DIR      [Ø]
  [1] - Tmp1
  [2] - Tmp2
  [3] - Tmp3
Memory Allocation
  Base = ØØØØH  Size = 41ØØH  * Free *
  Base = 41ØØH  Size = 2DØØH  * Free *
  Base = 6EØØH  Size = 11ØØH  Allocated to DIR      [Ø]
  Base = 7FØØH  Size = 2CØØH  * Free *
```

Notice that MP/M chose the smallest memory segment for DIR. It has the intelligence to select the memory segment best suited for the requested program. The criteria for selection is the size of the program and the size of the remaining memory segments. In fact, if the program requested will not fit in any of the memory segments, the "Reloc seg not free" message is displayed.

Enter the next two commands. Wait for the ØF> prompt before entering the second.

```
ØF>DIR@^D
ØF>DIR@^D
ØF>
```

Two more memory segments have been filled up now. Enter

```
ØF>MPMSTAT@
```

and the three categories should appear as

```
Process(es) Attached to Consoles:
  [Ø] - MPMSTAT
  [1] - Tmp1
  [2] - Tmp2
  [3] - Tmp3
Process(es) Waiting for Consoles:
  [Ø] - TmpØ  DIR     [Ø] DIR     [Ø] DIR     [Ø]
  [1] - Tmp1
  [2] - Tmp2
  [3] - Tmp3
Memory Allocation
  Base = ØØØØH  Size = 41ØØH  * Free *
  Base = 41ØØH  Size = 2DØØH  Allocated to DIR     [Ø]
  Base = 6EØØH  Size = 11ØØH  Allocated to DIR     [Ø]
  Base = 7FØØH  Size = 2CØØH  Allocated to DIR     [Ø]
```

As more programs are entered, the line-up in Process(es) Waiting expands. Notice in this instance that the first memory segment has been left open. This is because DIR didn't need a memory segment that big. There's another reason, though, which has significance when larger PRL type programs are invoked. When MP/M has two memory segments to choose from to run a PRL type program and one of them is the absolute TPA, it selects the other leaving the TPA free for a COM type program.

Finally, enter

```
ØF>DIR@^D
```

This fills up the all the memory segments. Entering this line again renders the "Reloc seg not free" message.

Entering MPMSTAT one last time shows

```
ØF>MPMSTAT@

Process(es) Attached to Consoles:
  [Ø] - MPMSTAT
  [1] - Tmp1
  [2] - Tmp2
  [3] - Tmp3
Process(es) Waiting for Consoles:
  [Ø] - TmpØ      DIR     [Ø] DIR     [Ø] DIR     [Ø] DIR     [Ø]
  [1] - Tmp1
  [2] - Tmp2
  [3] - Tmp3
Memory Allocation
  Base = ØØØØH  Size = 41ØØH  Allocated to DIR     [Ø]
  Base = 41ØØH  Size = 2DØØH  Allocated to DIR     [Ø]
  Base = 6EØØH  Size = 11ØØH  Allocated to DIR     [Ø]
  Base = 7FØØH  Size = 2CØØH  Allocated to DIR     [Ø]
```

It is best not to linger in this state too long; other operators may want to use the system. There are a couple of ways to free up

the memory segments. The ATTACH command or ^D can be used to re-attach the programs; the ABORT command (unlike ATTACH, ABORT is a RSP) can be used to terminate the program.

**** IMPORTANT ****

Although you can only attach programs assigned to your console, you can abort programs assigned to other consoles. This is very handy in the event a program crashes at that console and that terminal does not respond to any input. However, an element of danger is present: you can inadvertently abort someone else's program. The ABORT RSP description that follows goes into this in more detail.

****

To use the ATTACH command, enter

ØF>ATTACH DIR@

The listing of the USER Ø files on disk F: are subsequently displayed.

^D can also be used to re-attach programs. Once the ØF> prompt has reappeared enter

ØF>^D

Again the listing of the USER Ø files on F: are displayed. This time the message

Attach TmpØ

is displayed after the listing. This is as good as a prompt; the system will accept entries even though, in this case, the ØF> is not shown. For instance, enter another ^D after "Attach TmpØ" is displayed and another listing of the user Ø disk F: files results. Again, "Attach TmpØ" is displayed instead of the prompt.

Although this example doesn't show it, ^D re-attaches the programs in the order of appearance in the Process(es) Waiting category. ATTACH can be used to circumvent this feature if you don't want the next program.

The ABORT command has the same format as the ATTACH command. To terminate the last DIR command, enter

ABORT DIR@

This time, the ØF> prompt will be displayed.

The discussion of MPMSTAT has only addressed the use of the memory segments thus far. There's another component of the system that functions very similarly to these - the slave processors. Recall from Chapter 7 that only programs of type COM run in the slaves. COM programs are far and away the type run most frequently. (Most application software is provided in the form of COM files.) MPMSTAT does not show how the slave processors are assigned. This can be deduced, however, by analyzing the three categories described above.

The slaves are assigned on a first come, first serve basis. Slave Ø goes to the first person to invoke a COM type program; Slave 1 to the second; etc. In the event another COM-type program is invoked, the absolute TPA in the master processor is available. Once a program finishes, the slave or master absolute TPA is ready for re-assignment, regardless of its number.

To illustrate this, consider a two user system, with Mary and Hank at consoles Ø and 1 respectively, that has two slave processors. These slaves are not strictly assigned to Mary or Hank; they are allocated as the need arises. For instance, Mary can have slave Ø for one program but later have slave 1 when Hank has Ø. Or she can detach from the program in Ø and use slave 1 at the same time.

All COM files assigned to a slave that are attached to a console appear in the Process(es) Attached category in MPMSTAT. For instance, Mary and Hank are both PIPing files from one USER section of hard disk drive E: to another to keep their back-up files up to date. Mary and Hank have entered

```
(Mary)          2E>PIP E:=E:*.DAT[G1]

(Hank)          9E>PIP E:=E:*.DAT[G8]
```

For the purpose of this example, DAT indicates a data file. Notice that they both are using hard disk E: but each is referencing different USER sections.

If a MPMSTAT was performed at this time, the following would result

```
Process(es) Attached to Consoles:
  [Ø] - PIP     [Ø]
  [1] - PIP     [1]
Process(es) Waiting for Consoles:
  [Ø] - TmpØ
  [1] - Tmp1
Memory Allocation
  Base = ØØØØH  Size = 41ØØH  * Free *
  Base = 41ØØH  Size = 2DØØH  * Free *
  Base = 6EØØH  Size = 11ØØH  * Free *
  Base = 7FØØH  Size = 2CØØH  * Free *
```

Notice that all the memory segments are still free. This is because the PIPs are running in the slaves. Also notice that the numbers following PIP indicate the console number, not the current USER numbers. If Mary detaches from PIP and invokes another COM program, it will run in the master processor absolute TPA. For instance, she detaches from PIP and enters

```
STAT F:
```

to see how much storage space is available on hard disk drive F:. MPMSTAT would show

```
Process(es) Attached to Consoles:
  [Ø] - STAT     [Ø]
  [1] - PIP      [1]
Process(es) Waiting for Consoles:
  [Ø] - TmpØ       PIP      [Ø]
  [1] - Tmp1
Memory Allocation
  Base = ØØØØH  Size = 41ØØH  STAT      [Ø]
  Base = 41ØØH  Size = 2DØØH  * Free *
  Base = 6EØØH  Size = 11ØØH  * Free *
  Base = 7FØØH  Size = 2CØØH  * Free *
```

If Hank detaches from his program and attempts to run another COM type program when Mary is using STAT, this message is displayed.

```
Abs Tpa not free
```

This has the same meaning as the "Reloc seg not free" message but indicates that all the slaves and the master processor absolute TPA are currently in use. The message refers to the master processor's absolute TPA only since it is always the last allocated.

Notice from the DIRectory listings of the distribution diskette that many of the COM type files have corresponding PRL type files. If the program Hank had requested has a corresponding PRL file, it would have run in one of the memory segments (assuming one was big enough to accommodate it). This is because the M/NET version of MP/M tries to run the COM version of a program first. (This is different than the Digital Research version of MP/M; see Chapter 12.) If there are no available slaves and the master absolute TPA are busy, it looks for the PRL type and, if it finds one and an available memory segment, runs the program. To reiterate, this is true only for programs with both a COM and PRL file type.

ABORT.RSP

ABORT is used to terminate running programs. To abort a program running from your console, enter

ABORT <file name>

There is no need to enter the file type.

ABORT does all the things necessary to properly end a program. All open files are closed and slave processors or memory segments cleared for subsequent use.

Use of ABORT is only necessary if your console is detached from the program. If the console is attached, CTRL-C can be used to terminate it.

In some circumstances it is necessary to terminate a program assigned to another console. Typically, this is done to assist another operator whose program has crashed and finds themselves unable to terminate the program because the terminal is "dead" (the terminal doesn't respond to entries from the keyboard). The form of this command is

ABORT <file name> <x>

where <file name> is the name of the other operators program (an MPMSTAT will name the program if it is not apparent) and <x> is the other operators console number, not the USER number.

To see how ABORT works, enter

```
ØF>PIP F:=E:*.COM@
```

After a two of the COM files have been transferred, enter ^D to detach your console and perform a MPMSTAT. The following display is from a 4 user/slave system with no other operators active.

```
Process(es) Attached to Consoles:
  [Ø] - MPMSTAT
  [1] - Tmp1
  [2] - Tmp2
  [3] - Tmp3
Process(es) Waiting for Consoles:
  [Ø] - TmpØ      PIP     [Ø]
  [1] - Tmp1
  [2] - Tmp2
  [3] - Tmp3
Memory Allocation
  Base = ØØØØH  Size = 41ØØH  * Free *
  Base = 41ØØH  Size = 2DØØH  * Free *
  Base = 6EØØH  Size = 11ØØH  * Free *
  Base = 7FØØH  Size = 2CØØH  * Free *
```

PIP is running in one of the slaves so it doesn't appear in one of the memory segments.

Next, enter

```
ØF>ABORT PIP@
```

and another MPMSTAT. The subsequent display will not show PIP in the "Process(es) Waiting" category. Only the files that were transferred before the detach are resident on drive F:.

If you want to abort several programs, you must ABORT them one at a time. Ambiguous file references are not recognized by ABORT.

SCHED.RSP

SCHED allows an operator to schedule a program to run on a certain date at a specific time. Unlike the other resident system processes, it MUST be included in the system to have this feature. SCHED.PRL plays a supportive role in the scheduling process. When the SCHED command is entered, the PRL file is loaded into a memory segment and adds the day, time and program parameters into the SCHED.RSP queue. Several different entries can be appended to the queue.

The form of this command is

```
SCHED <date> <time> <program name> <object file name>
```

The <object file name> is optional. Not all programs require nor accept an object file.

For example, to run STAT to find the size of all COM files at 1Ø:31 am on May 31, 198Ø, you would enter

```
SCHED 5/31/8Ø 1Ø:31 STAT *.COM
```

The spaces between each parameter must be entered. Notice that the time parameter does not accept seconds.

If the sheduled program is not on the currently logged disk and USER section, these values should precede the program name. For instance, if the STAT file requested in the command immediately above only resided in USER section 2 on disk F:, the command should read

```
SCHED 5/31/8Ø 1Ø:31 2F:STAT *.COM
```

Once a program has been SCHEDuled, it cannot be aborted until it is running. Attempts to prevent the program from running on shedule via ABORT will render a "No abort" message.

SPOOL.RSP

The SPOOL facility is used to print files. Either one or several files can be lined up for printing. The command form is

SPOOL <file name>.<file type>,<file name>.<file type>, ...

Both the file name and file type must be entered so that SPOOL knows which file to print. (Of course, if the file has no type, this parameter can be left off.) A group of files to print are separated from eachother by a comma. A USER section and disk reference can precede the file name if the file is not on the current disk.

SPOOL operates differently depending upon whether it is a resident system process (i.e., incorporated into the system) or not (a PRL type file only). When it is NOT included in the system, the message

```
MP/M 1.1 Spooler
- Enter STOPSPLR to abort the spooler
- Enter ATTACH Spool to re-attach console to spooler
*** Spooler detaching from console ***
```

is displayed after the command is entered. If the printer is already spooling a file

```
*** Printer busy ***
- Spooler will wait until printer free
```

is added to the message. In this case, the command is put into the SPOOL queue (which holds up to 3 entries) and the file is printed after the current job is done.

No messages are printed if SPOOL is incorporated into the system.

(For the technically minded: Another difference between the two modes of SPOOL is that it works much more efficiently when it is NOT incorporated into the system. This is because as an RSP SPOOL only buffers one sector, 128 bytes, of the file at a time. When the PRL form is used, the entire memory segment assigned to SPOOL is filled. Thus, fewer disk accesses are performed.)

A hint about SPOOL use: If you have a group of files to print, enter the list in one command instead of entering each one with its own SPOOL command. In fact, SPOOL accepts ambiguous file references to facilitate operation. When entered with one command, the files will be printed just as quickly but memory segments or the SPOOL queue won't be tied up as they would if separate commands were entered.

For instance, if you have three ASM file to print (ASM files are the source from which COM programs are made) named DOIT.ASM, HELLO.ASM, and SOLONG.ASM, and they are the only ASM files in

that USER section, either of the following commands should be used.

```
SPOOL DOIT.ASM,HELLO.ASM,SOLONG.ASM
```

or

```
SPOOL *.ASM
```

If these files were in a different USER section (3 for instance) on disk E:, the following would be appropriate.

```
SPOOL 3E:DOIT.ASM,3E:HELLO.ASM,3E:SOLONG.ASM
```

or

```
SPOOL 3E:*.ASM
```

Notice that the USER/disk reference must precede all file names not in the currently logged USER/disk.

CHAPTER 1Ø: SYSTEM LEVEL OPERATIONAL PROCEDURES

By far the most common operation that will be performed on the M/NET system is the running of some sort of application software (general ledger, accounts payable/receivable, word processing, etc.). Refer to the manual provided with the software package for instructions regarding its operation (This manual cannot address application software operation except in a general sense since Micromation does not provide these programs.)

Before the programs can be run, some system set up is necessary. For instance, the programs should be moved from the software manufacturer's diskette to the hard disk (if one is present) or to a double density diskette. If there is a hard disk in your M/NET system, it should be used instead of the floppies for program and data storage; program execution is noticeably faster from the hard disk than from the floppies. In addition, it is recommended that different types of software packages be assigned to different USER sections on a disk. Since M/NET only allows one operator in a USER section at a time, the data files cannot be accessed by two operators at the same time. (See the discussion of the USER utility for the reason why this is not good.)

The topics in this chapter are

1) Basic M/NET operation
2) Changing the logged disk/USER section and diskettes
3) Formatting floppy diskettes
4) Making work disks and USER sections
5) Running a program and checking system status
6) Making back-ups
7) DETACHing and ATTACHing programs
8) ABORTing a program
9) Output to the printer

The discussion of each topic summarizes the use of the relevant M/NET utilities and features. The intent is to show how the various utilities relate to eachother to perform a desired task.

A reminder: In the examples that follow, all operator entries are underscored and @ indicates a RETURN should be entered.

1) Basic M/NET Operation

Once the system is fired up, it should be left on. In addition, a system reset should not be used unless extraordinary circumstances require it. Every time a system reset is performed (the reset button on the front of the computer cabinet is pressed), all programs attached to all consoles are terminated. About the only time a system reset need be performed is when the whole system comes to a crashing halt (i.e., all the consoles do not accept any entries). This should not be a frequent occurrence

and, in fact, there are only a few remote circumstances that render this state. (You would need to go far out of your way to crash the system.)

If you are not going to leave the system on all the time, ensure that **ALL** programs have ended (use MPMSTAT) and remove all diskettes from the floppy disk drives **BEFORE** turning off the power.

**** IMPORTANT WARNING ****

The hard disk drive **MUST** be turned off **BEFORE** the computer is turned off. The on/off switch for the hard disk is mounted on the back of the cabinet. If the hard disk is not turned off first, a sector or two will be trashed when the computer is powered down.

****

When it's time to power up the system, use the procedure in Chapter 3. Append to that procedure the TOD utility. When the system is turned off, the time of day keeper quits also. Upon powering up, the time of day reverts to its default setting (Fri 3/14/80 9:00:00). To summarize the procedure

- turn on all the components
- press the system reset button
- insert the copy of the MP/M distribution diskette in floppy drive A:
- close the latch

The MP/M Loader message is displayed (the process takes about 20 seconds) at console 0. The remainder of the consoles get just the MP/M sign-on message. The system prompt follows. The number indicates the console number (each console has a unique number) and the letter indicates the current drive. In M/NET systems with the hard disk, an automatic jump to drive E: is performed. In M/NET systems without the hard disk, drive A: remains as the current drive.

Once the prompt is displayed,

- enter the time of day

The TOD utility is described in Chapter 8.

The time of day will need to be re-entered every time a system reset is performed also.

2) Changing the Logged Disk/USER Section and Diskettes

Once the system is booted, the operator can move independently from the other operators to any disk(ette) and USER section. To move from the default disk(ette) drive, enter the disk name

followed by a colon. For instance, to move from hard disk drive E: to F:, enter

```
ØE>F:
ØF>
```

To move from one USER section to another, call upon the USER program. Note that the number that appears on the terminal when the system is booted is the console number; it cannot be changed. Whenever MPMSTAT is invoked, the number that follows the utilities listed is the console number, regardless of the current USER number displayed at that console.

The console number at cold boot is also the default USER number. To chage to a different USER number (in this case from Ø to 3), enter

```
ØE>USER 3

User number = 3
3E>
```

Notice that changing the USER number does not affect the active disk reference and changing the selected disk(ette) does not affect the user number.

If you ever get lost (i.e., you don't know what your console number is), enter

```
CONSOLE@
```

and your console number will be displayed.

3) Formatting Floppy Diskettes

**** IMPORTANT ****

This portion of Chapter 1Ø addresses floppy diskette formatting only. The hard disk is formatted at the factory and should not need it subsequently.

****

If your system includes the hard disk, floppy diskettes will be used almost exclusively for file back-up. Since system performance is so much faster from the hard disk, all the work files should be transferred from floppy diskettes to USER groups on the hard disk. Diskette preparation for this purpose requires little more than formatting the diskettes in double density.

If your system does not include the hard disk, diskettes will be used for program execution and data file storage as well as back-up. In this case there are two parts to the diskette preparation

process: 1) formatting the diskettes in either single or double density and 2) moving the application programs and relevant utilities to create work diskettes. (Work diskettes are those that are used on a day to day basis and contain programs and other files relevant to a specific task.)

Regardless of whether your system has a hard disk or not, the first step in diskette preparation is formatting the diskettes. To begin, purchase a box or two (10 or 20) of high-quality, single- or double-sided diskettes. We recommend those manufactured by Dysan or IBM. You will need the double-sided diskettes ONLY if you purchased floppy drives that record data on both sides of the diskette. These type diskettes do not work in drives that record on one side only. (Ask your dealer for assistance when purchasing diskettes.)

Once a batch of diskettes has been acquired, set aside a time when no other operators are active on the system to format the entire group. Do the entire batch at one time for two reasons: 1) The format program requires most of the master's CPU's time and attention. This may have little or tremendous impact on another operator's program depending upon the amount of user and disk I/O requirements of that program. 2) It's remarkably more convenient to have a diskette ready to go than to interrupt your work to format one.

Regarding which density (single or double) to use, format all the diskettes in double density. There is no reason not to and the benefit of doubled data storage capacity is significant.

Refer to the M/NET addendum for a description of Format.

4) Making USER Sections and Work Diskettes

To make system operation easier and provide file protect, each application software package should have its own USER section. Thus, when it's time to run, say, the general ledger program, you just move to USER section 3 on disk E: for instance. When you're in it, no one else can be USER 3 so your files are protected.

The procedure to set this up requires that the programs from the application software diskette be moved into a USER section on the hard disk. In systems without the hard disk, these files and the supporting MP/M utilities should be recorded in a USER section on a double density diskette. Data files will be added to this section as the program is run.

If You Have the Hard Disk, or Making USER Sections: Since it is best to run off the hard disk, the files on the diskette from the software manufacturer should be moved to a USER section on one of the drives. In this example, hard disk drive E:, USER section 3 is the destination. Any other USER section and/or drive (F:, G: or G:) can be substituted, however.

To begin, have the system running, insert the software master in drive A: (if there's no diskette already there), close the door and enter

DSKRESET A:@

If there is a diskette in drive B:, DO NOT remove it without first entering

DSKRESET A:

If the message

Disk reset denied, Drive A: Console x Program xx

is not displayed, go ahead and switch diskettes.

If this message is displayed, DO NOT switch diskettes. This means that someone else is using a program on that diskette. Either wait for drive A: to become free (enter DSKRESET A: until the message is not shown) or try this command on drive B: to tsee if it is available. If it is, substitute "B:" for "A:" in the next three instructions.

After the change has been made enter another

DSKRESET A:

to inform MP/M of the change.

If you are not on drive E: move there by typing

E:@

Move to user section 3 by entering

```
ØE>USER 3@

User number = 3
3E>
```

To transfer all the files from drive A: to USER section 3 of drive E: enter

3E>PIP E:=A:*.*[GØV]@

which means, "transfer to USER section 3 on disk E: from A: all files (*.*) in USER section Ø and verify." Each file is displayed as it is moved.

Perform a DIRectory on 3E> to ensure that all the files have been moved.

If there's an accompanying data diskette, the files from it must be transferred also. Replace the software master with the data diskette, enter another

3E>DSKRESET A:@

and re-enter the same PIP command used to move the program files. (The ambiguous file reference transfers all files regardless of their names.)

If there are programs for a number of different applications on the software master you may want to put each application in a different USER section. To do this, move yourself to the first USER section (3E> for instance) and use the interactive form of PIP or an ambiguous file reference if one can be used to transfer a select group of programs. Then, move to another USER section, transfer the next group of files, etc. Each PIP command line must include [GØV] to indicate the source USER section and to have a verify.

If You Have Only Floppy Disk Drives, or Making Work Disks: This procedure is much the same as that described for the hard disk; you still want to move the application software into different USER sections. It is different in that these sections will be on one or more floppy disks instead of one hard disk drive and you will need some of the MP/M utilities for file management. The reason these programs were not necessary above is because they were already transferred to USER group Ø on hard disk drive E:.

Gather the group of diskettes formatted in double density (as described above). Although none of these diskettes need the system files, some of the utilitie are necessary. Review the PRL and COM files described in Chapter 8 and make a list of those that you consider necessary. Several of these are a must:

PIP (to move files from one diskette to another)
STAT (to check file and diskette status)
DIR (to list the directory of the files)
USER (to change USER sections)
DSKRESET (to change diskettes)
ERA and/or ERAN (to erase files)

The remainder will depend upon the application. The first step below will be to transfer these programs to the work diskette(s).

Since the following instructions require switching diskettes frequently, set aside a time when there are no other operators on the system. If there are, the process may become horribly confused and tedious as each DSKRESET is denied. The instructions below assume that there are no active operators besides yourself.

Before changing any diskettes, enter

USER Ø@

if your are not already in this USER section and

```
A:@
```

if that is not the logged disk indicated by the MP/M prompt. Next enter

```
ØA>DSKRESET A:@
```

and

```
ØA>DSKRESET B:@
```

to ensure that no one is using the diskettes. If the "Disk reset denied ... " message is not displayed after both invocations, change diskettes. Otherwise, you will have to wait.

The first step is to transfer the appropriate utilities from the copy of the distribution diskette to the work diskette. Place the distribution diskette in drive A: and a blank, double density diskette in drive B:. Enter the DSKRESET command twice, once for each drive.

Since you probably don't need all the PRL and COM files, the interactive form of PIP should be used to move those you need. Enter

```
ØA>PIP@
```

and the PIP prompt, *, will appear. Next, using your list of appropriate utilities, enter

```
*B:=A:<filename>.*[V]@
```

(replacing <filename> in this command with the name) for each program. The "*" in the command line replaces the file type so that both the COM and PRL version of the utility is transferred. "[V]" is entered to verify the file transfer. Wait for the PIP prompt before entering subsequent commands. When the last utility has been moved, enter a RETURN only to exit PIP and get the MP/M prompt back.

Since the file type was replaced by the asterisk, the ASM copy of the utility (if it was present on the distribution diskette) was transferred. You will not need these files so enter

```
ØA>ERAN B:*.ASM@
```

to remove them.

When the work diskette has all the appropriate utilities, replace the distribution diskette in drive A: with it and enter

```
ØA>DSKRESET A:@
```

If you need several work diskettes for program storage, it would be expeditious to use the one just created as a source. That is, the utilities just transferred may be appropriate for several different applications. If this is the case, put another blank, double density diskette in drive B and enter the next two commands, waiting for the prompt before typing the second.

```
ØA>DSKRESET B:@
ØA>PIP B:=A:*.*[V]@
```

Repeat these two commands for every work diskette that needs these programs.

Keep these diskettes separate from the blanks to avoid confusion. They will be labelled when the application software is recorded.

Finally, remove the last word diskette from drive B:, insert the software master there and enter another DSKRESET B: to inform MP/M of the change.

As with the hard disk, we recommend applying a unique USER number to each set of application programs. The instruction below appends 3 to the file name. Any other number between 1 and 15 can be used though. If you select another, substitute it for 3 in the next two commands.

To change USER numbers, enter

```
ØA>USER 3@
```

To move all the files from the software master to the work diskette enter

```
3A>PIP B:=A:*.*[GØV]@
```

Each file is displayed as it is transferred and verified.

Repeat this operation for each set of application programs paying close attention to the amount of space left on the destination diskette. Enter

```
3A>STAT@
```

between PIPs to ensure there's enough room.

Many application programs require a separate diskette for data files. This diskette typically needs no utilities on it. If it does, however, put a blank, double density diskette in B: once the work diskette is completed, reset the drives and PIP the requisite utilities.

If several different programs performing different tasks are provided on the diskette, you may want to separate each task into a separate USER section. To do this, PIP only those files pertinent to each task (use the interactive form of PIP or an ambiguous file reference if it is possible), change the USER number for the diskette, and transfer the next group with PIP. Every PIP command line must have [GØV] to indicate the source USER section and to have the transfer verified.

Finally, label the diskette(s). Fill the label out before applying it to the diskette. If additional notes are needed after the label has been stuck on the diskette, use a felt tip pen only (a ball point pen or pencil may score the diskette underneath). Use the following as a model.

```
MP/M Program Disk - No system
        DD/MM/YY
General ledger - USER 3
Accounts Payable - USER 5
   "      Receivable - USER 7
```

Once the software master files have been transferred to either the hard disk or a double density floppy disk, store the original diskette in a safe place for future reference.

5) Running a Program and Checking System Status

All programs run under MP/M have a COM or PRL file type. Almost all application software will be of type COM only. As you have probably noticed by now, the form of the command to invoke these programs (whether COM or PRL, it doesn't matter which) is

<COM/PRL file name> <object file name> <object file name>

where the first file name indicates the COM or PRL file to be executed (there's no need to enter ".COM" or ".PRL", it is assumed by the computer) and the remainder of the names indicate the files to be worked on. Note that the second object file isn't frequently used. For instance, to find the size of the DDT utility, you would enter

```
STAT DDT.COM
```

Many application programs are written in the BASIC language. With this, the form of the command to run say a general ledger program named GL is

```
RUN GL
```

Notice that the form of the command is the same with RUN as with the MP/M utilities. With BASIC files, the file type of the object programs is INT (GL.INT in this case); this, too, is assumed when RUN is invoked.

The general procedure, then, is to move to the disk(ette) drive and USER section dedicated to the application and enter the command as illustrated. If you are unsure as to which USER section your programs are in, enter

```
STAT USR:
Active User : x
Active Files: x y z
```

where "Active User" indicates your current USER number and "Active Files" indicates the different USER sections on the disk(ette). Move to each USER section (with the USER command) and perform a DIRectory to determine which one your application is in.

The only problem you may encounter is the occasional instance when other operators have tied up all the slave processors and/or master processor memory segments. In this event, you will get one of the following messages:

```
Abs Tpa not free

Reloc seg not free
```

The first message indicates that all the slaves and the absolute TPA in the master processor are already allocated to other operators. The second message indicates that either all the master's memory segments are allocated or there are no memory segments large enough to run the PRL type program available. To see how the slaves and memory segments are allocated, enter

```
MPMSTAT
```

The last three categories in the display show the active programs and the console they're attached to, the programs waiting for a console and which console they're from, and the allocation of the memory segments (program and console), respectively. (The preceding categories indicate other aspects of system status. Most of this information isn't relevant to normal operation, though.)

In most circumstances, you will just have to wait until a slave (or memory segment if the program is a PRL type) opens up before you can run your program.

Note the frequency of these messages. If they occur regularly you may want to consider adding another slave processor (if the first message occurs frequently) or change the memory segments in the master (if the second recurs often).

There's one other situation that can affect program execution: a program will stop executing if it needs to output to the printer but the printer is busy. In this case, the message

```
Printer Busy
Wait or type ^C to abort
```

is displayed. You have several options: 1) you can wait until the other operator is done with the printer in which case the program will automatically start outputting to the printer. Or 2) you can enter a CTRL-C (simultaneously press the CTRL and C keys) to terminate the program. This is a normal program exit and should save all the information you have entered (if the application software manufacturer has written the program properly).

A third option is available, you can detach from the program (enter a ^D) and start another. If the original program requires no console input or output, it will proceed when the printer is free. If it does require some console input or output, the program will do nothing until it is re-attached. Assuming the printer is free when the program is re-attached, the program will start to print. You can perform an MPMSTAT to check on system status. If the printer is busy, the name of the program appears in the Queue category under MXList. (The section entitled, "Output to the Printer" in this chapter describes MXList.)

6) Making Back-Ups

ALL files should have back-ups. You backed-up the system diskette in Chapter 4. In addition, you should make a back-up of all software purchased for the system. The files you create when executing the application software should be backed-up every time a change is made. Although problems occur infrequently, you will thank your lucky stars you made a back-up when one does. (And besides, Murphy's law states that the one time you don't make a back-up you will get a bad sector message. This means the system cannot read that sector of the diskette.)

There are two ways to maintain back-ups: the files can be moved to another hard disk drive or to floppy diskettes. It is recommended that the same USER section is kept for convenience sake. Backing-up to another hard disk drive is the fastest and easiest way but it does not provide 100% security since the hard disk could go down or the files inadvertently erased. These are rare occurrences, however. Backing-up to floppy diskettes is a tad slower and more inconvenient but almost all precautions are satisfied.

Backing up to another hard disk drive: This is the fastest and most convenient way to make back-ups. We recommend using the same USER section on a different hard disk drive. For instance, have USER section 3 on drive E: for program execution and USER section 3 on drive F: for the back-ups.

This makes the file transfer process very simple. The following command can be used in this situation to transfer all the files.

```
3E>PIP F:=E:*.*
```

The [Gx] addendum to other PIP examples is not necessary since the destination is in the same USER section.

This is the command to use to transfer all the files if all files have been altered or if it isn't apparent which files were changed. If only one or two files were altered, you can transfer just the files affected with the interactive form of PIP or PIP and an ambiguous file reference.

As you continue to make back-ups with PIP to the same USER section, the previous back-up copy of the file is erased.

Backing up to a floppy diskette: For 100% security, altered files can be transferred to a floppy diskette. To begin, get a double density formatted diskette (with the write protect notch covered). If there's a diskette in drive B:, type

```
DSKRESET B:@
```

before removing it and putting in the back-up diskette. If no one is using that diskette, pull it out, insert the other, and enter another

```
DSKRESET B:@
```

Note that there's only about 500K bytes of space on a single sided, double density diskette. Drives and diskettes that record on both sides can accommodate about 1000K bytes per diskette. Although it's unlikely that one file will fill an entire diskette, the accumulated back-up files might. So before you start transferring files, enter

```
STAT B:@

Bytes remaining on B: 483K
```

Your diskette may show more or less space. It will definitely show less space if there are files recorded on it already.

When a lot of files have been copied, it would be wise to see how much the space the files to be moved require as well. For instance, diskette A: has a bunch of data files (with file type DAT) to be backed-up. The operator would type,

```
STAT *.DAT@
```

and the files and their size would be displayed. If these files haven't been transferred before (i.e., there are no files of the same name residing on B: from previous back-ups), add up the space requirements and compare this value with that displayed in the previous command. If there isn't enough room on B:, use another diskette.

If the files have been transferred to the diskette in B: before, each file from A: will replace the one with the same name on B:. All you need to do is check to ensure that there's enough room to accommodate the largest file in the batch to be copied. (Recall that PIP erases the original file after the new one is completely transferred. Consequently, at one point during the transfer process, there must be enough space to accommodate both.)

Now that the diskette in B: has been inserted and it's been determined that it contains enough space to receive the files, the transfer can begin. Move into the appropriate USER section and disk drive with the USER and disk change commands and enter

```
PIP B:=E:*.*@
```

The ambiguous file reference "*.*" causes all files to be copied. This isn't necessary but it is the easiest. If only files of type DAT (indicating data files) were altered and needed to be backed-up, the command would be

```
PIP B:=E:*.DAT
```

When the process is complete, enter another

```
DSKRESET B:
```

to ensure that no one else has started using the back-up diskette while you were moving files over to it (this is a remote possibility but a possibility nevertheless). If another operator is not using it, remove the diskette and store it in a safe place.

7) DETACHing and ATTACHing Programs

A feature of MP/M allows operators to detach from a program so that another can be started. One of two things will happen to the detached program:

- If it requires no console input or output, the program will proceed without interruption until it's done.

- If it requires console input or output, the program

will stop until the program is re-attached.

Just because a program requires no further operator input to continue executing does not mean it will proceed if it is detached. Many programs output to the terminal their status after all input has been made. (For instance, the PIP utility invoked with an ambiguous file reference displays the files as they are copied.) These programs WILL NOT proceed once detached. Only when they are re-attached and allowed to output to the console will they proceed.

Most programs can be detached at any time. Simply enter a CTRL-D (simultaneously press the keys labelled CTRL and D). This is illustrated as "^D" in the following examples. The prompt should be displayed soon thereafter.

There are two ways to re-attach a program. The operator can enter another ^D or enter

```
ATTACH <program name>
```

where <program name> is the name of the program detached. For instance, Mary has a very long file (200K bytes long) named GL1.DAT to back-up. She enters

```
4E>PIP F:=E:GL1.DAT@^D
4E>RUN AP
```

Transferring the file takes only two or three minutes but she is eager to start another program. So, right after the carriage return she enters a ^D, gets the prompt and enters the next program (for accounts payable). While she is executing the AP program PIP is transferring her file. Since no console output is performed under PIP when an unambiguous file reference is used, GL.DAT is transferred. In this case, she need not re-attach PIP for it to terminate.

While entering invoices in the accounts payable, she realizes that GL2.DAT needs to be backed-up also. She detaches from AP, gets the prompt and enters the PIP command to copy GL2.DAT. Again she detaches from PIP (it will proceed to transfer her file and end) and gets the prompt. The accounts payable has been in limbo since she left; nothing has changed in it. To get back to it, she enters

```
4E>ATTACH RUN@
```

Notice that she did not type "ATTACH AP"; RUN is the name of the program executing. If Mary had performed an MPMSTAT before re-attaching the accounts payable program, "RUN [0]" would have appeared in the "Process(es) Waiting" category.

Operators can only attach and detach programs initiated from their console. Programs invoked from one console are not accessible from another.

8) ABORTing a Program

Unlike the detach and attach features, a program can be aborted from any console, regardless of who originated it. This facility is available through the ABORT program in the form of either a resident system process or PRL type file. This feature is very useful and very dangerous in the hands of an unscrupulous operator.

For a program to be aborted, it must first be detached. The command is

ABORT <program name> <console #>

Alternately, a program can be aborted without detaching by entering a CTRL-C (^C). This is only possible while in the program, however, and only possible from the console that originated the program.

To terminate a program assigned to your console but detached from it, enter ABORT and the program name. The console # is not necessary. For instance, Mary is running the Accounts Payable program and realizes she doesn't have all the invoices. Rather than entering the few she has, she decides to wait until the complete list is available. First, she detaches from the program with a ^D and subsequently enters

ABORT RUN

All information entered is saved and all the open files are closed in the appropriate manner. Note that the AP program was written in the BASIC language and invoked with the command, "RUN AP". Hence RUN was the program to abort, not AP. The MP/M prompt is immediately returned.

Mary gathers the missing invoices and begins AP over again. Her work is suddenly interrupted with a phone call. It's Hank in another office; he was running the marvelous (but flaky) DOIT program and found a bug; the bug that crashes the program and renders the terminal oblivious to user entries (i.e., he can't escape from the program by himself). She decides to help him. First she detaches from her program then enters

ABORT DOIT 2

Control of his terminal returns to Hank. DOIT has been terminated.

It is very important to enter the right console number. In four user systems it is not unusual for each operator to be running the same program. For instance, if each operator is executing a BASIC program, RUN will appear attached to each console in an MPMSTAT in the Process(es) Attached category. If one of them crashes and must be aborted from another console, you must enter the appropriate console number; the computer cannot figure it out for you.

A program can be terminated without detaching with a ^C. This assumes, however, that the computer is still allowing entries. (Hank could not have terminated DOIT with a ^C in the example above because his terminal was locked out by the crash.) When a ^C is entered, the following message is displayed

```
ABORT (Y/N)?
```

A "Y" response ends the program and returns the MP/M prompt; "N" causes a return to the program.

9) Output to the Printer

There are several ways to output to a printer: under program control (i.e., the program outputs to the printer during execution), with the SPOOL utility, with the PIP to LST: command and with the ^P special character. (Recall the ^P causes all console input and output to be duplicated by the printer.) In all cases, MP/M references the XIOS.SPR section of the operating system in search of the printer driver routine. (The driver routine is a special program kept in XIOS.SPR that controls the printer.) Since only one program can be outputting to the printer at a time, a queue is maintained labelled MXList. This appears in the "Queue" category of MPMSTAT. The MX prefix indicates that it is mutually exclusive or accessible by only one program at a time. (This prefix has the same meaning for other labels in the Queue category.) Once a program is put in the MXList queue, it has control of the printer until it terminates. This is to prevent other programs from accessing the printer when the original is not (i.e., between outputs when the program is doing something else).

If the printer is in use when a program needs to output to it,

```
Printer Busy
Wait or type ^C to abort
```

is displayed. The operator has two options;

1) Wait until the other program has finished at which point MP/M passes control of the printer over to the next in line; or
2) Enter a ^C to end the program, waiting until the printer is free before invoking the program again.

This message only appears when the printer is accessed under program control. If you are using SPOOL and the printer is busy, a different message is displayed. In this case, the file to be printed goes into the SPOOL queue; ^C does not abort SPOOL.

The XIOS portion of MP/M presently supports a serial type printer. If you have different printer, you will need to write (or have written) a driver routine for it. It should be inserted into XIOS.SPR. Refer to the MP/M User's Guide for this procedure.

**** IMPORTANT ****

Some programs are not designed to exit to the operating system when they terminate. Instead a menu appears allowing selection of another part of the program for execution. If the printer was used in a program like, it will not be disabled; the program will keep control of it and no others operators will have access to it. If you are using such a program, be sure to exit to the operating system so that other operators will have access to the printer.

****

SECTION III

REFERENCE

The following chapters describe M/NET with more detail, filling in the technical information not necessary for day to day operation and omitted for the sake of non-technical users.

Chapter 11 describes the contribution of each printed circuit board in the system. These descriptions do not replace the board manuals found elsewhere in the documentation package; the intent is to summarize their purpose and explain the differences/addendums relevant to the M/NET system.

Chapter 12 contains a description of the MP/M vis-a-vis its implementation in the M/NET system. Micromation uses MP/M as a base to maintain CP/M compatibility. However, this foundation has been built upon dramatically to add multi-processing to the multi-user capability. Where significant differences between the Digital Research and Micromation versions of MP/M appear, the dissimilarities are identified. These should be kept in mind when changes to those portions of MP/M that allow alteration are made. (Generally, the only segment of MP/M that may need alteration is the XIOS portion. This section contains the printer driver routines.) This section does not replace the MP/M User's Guide, it supplements it.

CHAPTER 11: COMPONENT DESCRIPTIONS

The M/NET system contains five basic printed circuit boards:

Z-64 Master Processor
Z-64 Satellite (Slave) Processor
DOUBLER Floppy Disk Controller
M/NET I/O Board
Hard Disk Controller
(only present in M/NET systems with a hard disk)

In each system, there's one Master Z-64, one DOUBLER, one Hard Disk Controller if the hard disk option is purchased, and one M/NET I/O board. The number of satellites in a particular system currently varies from 1 - 4.

Z-64 MASTER PROCESSOR: The master processor in the M/NET system serves several functions. The MP/M operating system is loaded into the 60K of available memory space in this component. About 32K of memory space is left over (this amount will vary depending upon the resident system processes selected) for execution of PRL type programs and the occasional COM type program when the slaves are all allocated. In addition, the master controls all data input and output between the system and the peripherals (the floppy disk drives, optional hard disk and terminals) for the slaves.

The memory map in the master is as follows.

Location (in hex)	Description
0000-00FF	Reserved
0100-F2FF	System and User Memory Segments
F300-F3FF	Scratch Area for DOUBLER
F400-F7FF	Hard Disk Buffer
F800-F9FF	DOUBLER I/O
FA00-FBFF	Reserved
FC00-FFFF	DOUBLER PROM

The MP/M operating system is loaded into the upper portion of the masters RAM. The top address available (i.e., that area not dedicated to the DOUBLER's firmware and buffers, hard disk buffer, and other purposes) is F2FFH in all M/NET systems. This is the value that should be entered in response to the top of memory prompt in the GENSYS program. The exact locations in memory of the various MP/M system components (the SPR and RSP programs) are listed at console 0 when the system is cold booted. The display is shown in the next chapter.

The operating system occupies about half of the master's memory, leaving 32K bytes. This remainder is subdivided, according to the user's requirements, into memory segments by the GENSYS utility. Any number of memory segments may be created. However, small ones (those below 1000K) will have little utility as this space is too small for many of the PRL programs to run.

The memory segment that starts 0000H has special status. Since it starts at this location, programs of type COM can run here. Recall that COM programs are usually passed on to one of the slaves for execution. This additional space is available in the event all the slaves are occupied. PRL type programs will also run here. However, MP/M does not assign this segment unless the other segments are already assigned or none of the remaining segments are big enough to run the utility.

All data I/O between the peripherals (disk storage or terminal) and the master and slaves is dispatched by the master. It is the only component that has control of the S-100 bus. (The slave processors have no control over their bus drivers.) Data is transferred between the master and the controllers (DOUBLER and hard disk) under program control, one byte at a time. Data is transferred between the master and the slaves using the Z-80 block move instruction. This is done by transferring the data from one memory location in the master to another in the slave. The memory locations in the master that correspond to the source/destination locations in the slave are phantomed out in the master while the slave's Z-80 is put on hold. Thus, the master is always looking at 64K of memory. The block move is performed fast enough to satisfy dynamic RAM refresh requirements.

The entire system uses the eight vectored interrupt lines and the NMI (non maskable interrupt). The NMI signal (non maskable interrupt, pin 12 on the S-100 bus) to the master processor, MUST NOT be installed. (It is used by the slaves only; see below.)

The eight Z-64 interrupt signals are allocated as follows.

Signal Name	Description	Priority
VI0	DMA Controller Active	lowest
VI1	Real-Time Clock	
VI2	UART Transmitters	
VI3	UART Receivers	
VI4	Slave Processors	
VI5	Hard Disk	
VI6	Timeout Counter (for floppy disk)	
VI7	Reserved for use by DDT	highest

Z-64 Slave Processor: This altered Z-64 can be used as a slave only. It contains a CP/M emulator in the 2K PROM to translate CP/M BDOS calls to their MP/M equivalent and supply the necessary parameters. The alterations include removal of the board's bus driving capabilities and use of eight traces from the S-100 bus to serve as slave Z80 disable signals (used during data transfers between master and slave).

The CP/M emulator PROM is addressed at FB00 - FFFF (the entire 2K bytes aren't used) and is used to intercept a program's BDOS calls. Each call is translated into a message for transfer to the master processor. Messages rendered by the emulator PROM are transferred from the slave to the master via "mailboxes," special segments of the slave's memory. The mailbox for data to the master is 6 bytes long; for data from the master it's 4 bytes long. The information moved contains an operation code, the contents of certain Z-80 registers (corresponding to BDOS conventions) and, in the case of the slave to master communication, the DMA address for a data transfer. The op codes and register assignments of the mailboxes are described in Chapter 12.

An important feature of the M/NET system is that each slave contains just over 62K bytes of transient program area. There is no overhead required for the system beyond the 2K allocated to the PROM.

Eight previously unallocated lines on the S-100 bus have been taken for use in the system. Lines 59 - 66 are used as slave select for data transfers. Each line puts a different slave processor on hold (with the Z-80 /BUSREQ signal) so that it is disabled during a data transfer. Notice that eight lines are provided allowing up to eight slaves. (Currently only four are allowed due to the size of the mother board.)

Pins 12 and 70 of the S-100 bus are also used in a special manner. Pin 12, typically associated with the NMI signal (non-maskable interrupt) is used for a software reset of the slave. This is done by first putting the slave on hold (with /BUSRQ), passing the appropriate code to location 0066H in the slave from the master, sending a NMI to the slave, then releasing it. A jump to location 0066H is always performed by an NMI. The signal from pin 70 of the S-100 bus provides the /MEMRQ equivalent (NANDed SMEMR and MWRITE signals indicating a memory read or write) to activate the slave RAM when its resident Z-80 is on hold. The source of these signals is the M/NET I/O Board.

DOUBLER Floppy Disk Controller: This DOUBLER controls data transfers between the master and the floppy diskettes. It is a remarkably flexible controller as it can determine if the drive is single or double sided (reads and writes on one side or both sides of the diskette) and if the diskette is formatted in single or double density.

No commands need be made by the operator to indicate to the DOUBLER the recording density of a diskette. It reads the standard 3740 single density format: 77 tracks each with 26 sectors of 128 bytes. However, double density may or may not be compatible with the double density format from other computer manufacturers. Micromation uses a double density format of 77 tracks composed of 52 sectors of 128 bytes. The MFM (modified frequency modulation) technique is used.

The DOUBLER used in the M/NET system is similar to that used in Micromation single user systems with differences in the locations of several functional components. The resident firmware now resides at FC00 - FFFFH with a page of scratch at F300 - F3FFH and I/O ports between F800 - F9FFH. This is a reorganization from the standard single user configuration. (If you upgrade your DOUBLER for use in M/NET, new firmware and address select PROMs are provided. See the M/NET Installation Guide for their placement.)

Data is transferred between the DOUBLER (floppy diskette) and the master under program control. A slight modification has been made to maximize utilization of the master's valuable time. If the sector read from the diskette during an access does not match the sector desired, the time required to reach it is calculated and the master goes on to something else. When the time has expired, the master is interrupted by the timeout counter interrupt (VI6) and services the DOUBLER. If the sector read matches the one desired, the data transfer proceeds uninterrupted.

In addition to the disk interface circuitry, a UART is installed on the DOUBLER for output to a serial type printer.

M/NET I/O Board: Besides providing the four UARTs for the terminals and parallel output ports for a Centronics type printer, this board serves several other, important functions. The M/NET I/O Board contains a 8253 programmable timer used to provide the baud rate for the UARTS, the ticks for the real time clock maintained by MP/M (via vectored interrupt VI1), and the timeout counter for the floppy drives (VI6). Recall from the discussion of the DOUBLER that a timeout (calculated by multiplying the number of sectors between the one read from the diskette and the one desired by the time it takes to move from one sector to the next) is inserted to allow the processor to go on to something else while the floppy is rotating to the required sector. Vectored interrupt line VI6 is used to interrupt the master processor when the sector is underneath the read/write head.

The M/NET I/O Board also generates two signals used to DMA data between the master and slaves. S-100 bus lines 59 - 66, labelled SP1 - SP8, are used to disable one of the slave processors to allow the DMA. To accommodate the slave Z-80, this signal becomes

/BUSREQ (bus request). The Z-80 is put on hold so that the master Z-80 has control of the slaves memory space.

When the slave processor is on hold, the requisite signals for memory enable must be provided off-board. One of these signals is generated by the M/NET I/O Board on pin 70 of the S-100 bus. It roughly corresponds to a NAND of the memory read/write (SMEMR/MWRITE) signals with a couple of other signals added to render a signal generated exclusively for the slaves. (Only one other signal, PDBIN, is required for the master to read from or write to the slave. It is generated by the master.)

Finally, the M/NET I/O board contains a high current 8-bit parallel output port and provides one of the 8255 programmable peripheral interface ports to output to a Centronics-type printer.

Hard Disk Controller: The hard disk interface is an optional component only necessary if a hard disk drive is installed in the system. Although this component is not necessary, it is strongly recommended. System performance is dramatically enhanced because of the decreased access times and increased data transfer rate inherent to the hard disk.

Data is transferred between the master processor and the disk drive through a buffer. For a write to disk, the master processor fills up the buffer first and the controller takes over. An interrupt is generated (on VI5) to indicate that the write is finished. For a read from disk, the buffer is filled by the controller and the interrupt is generated (also VI5) indicating that the data is ready to be read by the master.

CHAPTER 12: MICROMATION MP/M IMPLEMENTATION

The Micromation enhanced MP/M is very similar to the Digital Research release in appearance and operation. To implement multi-processing necessitated the installation of the Z/NET resident system process (RSP). It is this component of the operating system that assigns the requested COM-type program to one of the slave processors. (In fact, this RSP can be left out during a GENSYS and the system will work but from the master only.)

This is the only change to MP/M that is visually apparent. The remainder of the enhancements are in the form of alterations to system components already present in the operating system. These are presented as they become pertinent to the discussion below.

This chapter is divided into 4 sections:

1) The MP/M Loader
2) Process Descriptors, Queues and MPMSTAT
3) Program Processing
4) Communication between Master and Slave

The discussions focus on a particular aspect of MP/M. Through these presentations, it is hoped the big picture can be gleaned. Like other chapters in this manual, it is not our intention to replace the Digital Research MP/M documentation. Both should be referenced for a complete understanding of the system.

1) The MP/M Loader

MP/M is loaded differently than CP/M. In the latter operating system, the complete O/S is contained on the first two tracks (Ø and 1) of the system diskette. Because MP/M is so much larger, this approach was no longer viable. Instead, a primitive form of CP/M Ver. 2 is written on the first two tracks. It contains a boot program in the first sector followed by the BDOS and an abbreviated BIOS and CCP. The remainder of the first two tracks contain a program called MPMLDR. (This system component is duplicated in the none system tracks by a file of the same name.) MPMLDR reads the MPM.SYS file created by GENSYS which contains the parameters for loading the remainder of the system components. Hence, the display at console Ø staggers a bit as each component is first read from the diskette then loaded into the master's memory.

The console Ø display shown when the system is loaded summarizes the components and states their location and size.

```
SYSTEM   DAT    F2ØØH ØlØØH
CONSOLE  DAT    F1ØØH ØlØØH
USERSYS  DAT    FØØØH ØlØØH
XIOS     SPR    ECOOH Ø4ØØH
BDOS     SPR    D8ØØH 14ØØH
XDOS     SPR    B9ØØH ØlFØH
```

The remainder of the display summarizes the location and size of the Resident System Processes (RSPs) selected during GENSYS. Refer to the MP/M manual for a description of each system component. Notice that two components were made unnecessary because of the multi-processing approach: ODOS and BNKBDOS. These are only necessary if bank switching is used to expand the memory capacity in single processor systems.

2) Process Descriptors, Queues, and MPMSTAT

The MPMSTAT resident system process (or PRL form if it is not selected in GENSYS) and the Queues it manifests summarize the MP/M components and their interaction. This section focuses on that portion of the display not discussed in previous chapters; i.e., all categories physically above the "Process(es) Attached to Console" category shown in the MPMSTAT display.

Before describing the categories a few words about Process Descriptors and Queues are appropriate. Every program operating within MP/M gets a **Process Descriptor** assigned to it. This includes a name and set of characteristics that describe the program. As implemented in the Z/NET system the name is the same as that used by the operator to invoke the program. For instance, the DDT program has the process descriptor DDT assigned to it once it has been assigned a slave processor. (The same is done for a PRL type program when it is assigned a memory segments.) In addition, there are some processes that go on in the system beyond the control of the operator. Tick and Clock in the MP/M display are examples. The factor that distinguishes these from the former is the ability of the operator to abort the process. The visual manifestation of the difference is the presence in the process descriptor of a lower case letter. If the descriptor has at least one lower case letter, it cannot be aborted; if it does not (like DDT above), it can be aborted. The characteristics given to each process are described in the MP/M User's Guide.

Queues are first in first out mechanisms used in MP/M for several purposes: to exchange information (messages) between processes, to synchronize processes and to provide mutual exclusion. As with the process descriptors, the presence of lower case letters in the queue name has significance. Those queues that contain upper case only letters are available to the user; i.e., an entry can be made to the queue by the operator directly from the TMP (terminal message processor) via the command line interpreter (CLI) in the same manner as invoking a program. Those that contain upper and lower case letters are used by the system and do not have this characteristic (more on this later).

The reproduction of an MPMSTAT display that follows is for a four user system as invoked from console Ø. Although your display may be different, the dissimilarities are not significant for this discussion.

```
***** MP/M 1.1 Status Display *****

Top of memory = F2FFH
Number of consoles = 4
Debugger breakpoint restart # = Ø7
Stack is swapped on BDOS calls
Z8Ø complementary registers managed by dispatcher

Ready Process(es):
MPMSTAT Idle
Process(es) DQing:
 [ABORT   ] ABORT
 [CliQ    ] cli
 [ATTACH  ] ATTACH
 [Sched   ] Sched
Process(es) NQing:
Delayed Process(es):
Polling Process(es):
Process(es) Flag Waiting:
  Ø1 - Tick
  Ø2 - Clock
  Ø4 - Tmp1
  Ø5 - Tmp2
  Ø6 - Tmp3
Flag(s) Set:
  Ø3
Queue(s):
  MLoader  ABORT    MPMSTAT   CliQ    ATTACH    MXParse   MXList
  MXDisk

 ...
```

The remainder of the MPMSTAT display details the processes attached to the consoles, those waiting, and the allocation of the memory segments (see Chapter 9 for the discussion of their use).

The first five lines of the display summarize the system configuration vis-a-vis the top of memory, number of consoles (which frequently but not necessarily indicates the number of slave processors also), etc. The first category that describes the activity of the system is "Ready Process(es)."

Ready Process(es) are those programs that are running or ready to run but waiting for the master CPU. In this display, MPMSTAT was running so it is listed first. If the system could be viewed without running MPMSTAT when no operators were active, Idle would be the only item shown. Idle is one of the processes in MP/M not accessible to the operator. All programs are listed in the "Ready Process(es)" category; those running in the slaves as well as in the master. Thus, unlike single processor MP/M systems where only one program is running at a time while the others wait for the CPU, all the programs listed here can be running.

Process(es) DQing are those that are waiting for a message to be written in the queues shown; ABORT, CliQ, ATTACH, and Sched in the example above. ABORT, ATTACH and Sched correspond to the RSPs described in previous chapters. CliQ is the Command Line Interpreter queue. The CLI interprets the command passed to it from the TMP and determines if the command references a queue, a COM type file or a PRL type file. Notice that CliQ and Sched have lower case letters in their descriptors. This means that they are not accessible directly from the TMP. The CliQ is like Idle in that it is a queue within MP/M that is inaccessible to the operator. The Sched queue is not directly accessible by the TMP, the combination of the SCHEDULE RSP and PRL file make entries to this component. ABORT and ATTACH access the equivalent queue directly from the TMP though as indicated by the presence of upper case letters only.) In the display, the queue name is bracketed, the process descriptor is shown to the right. The occasion is rare that you will encounter any items other than the ones shown above because processes are present only briefly in each.

Process(es) NQing are those that are waiting for a free buffer to write a message to the specified queue. When the system is busy with several operators, a buffer may not be available. For instance, two operators may be attempting to access the ATTACH queue at the same time. In this event, the queue name is listed in brackets followed by the name of the process waiting for the buffer.

Delayed Process(es) are those that are waiting for a specific number of ticks of the system time unit to elapse before continuing. This category does not indicate those process(es) waiting in the schedule queue. Only those programs that use the Delay XDOS system call appear here when the delay is in effect.

Polling Process(es) are those that are polling a I/O device seeking device ready status. In the Z/NET system, the only device that would fall into this category is the output LST: device; console I/O is interrupt driven. Consequently, if PIP to LST:, SPOOL or CTRL-P is used to print a file, the process appears in this category between output characters.

Process(es) Flag Waiting indicates those interrupt driven processes that are not active at that time. Tick and Clock are processes like Idle mentioned above that are beyond the control of the operator. They are used for the time of day and delay functions. The Tmp1 - Tmp3 processes (Terminal Message Processor) correspond to each terminal in the system. When the terminal is not in use, the flag is waiting.

Flag(s) Set contains those interrupt driven processes that are active. In the display above, console Ø was used to invoke MPMSTAT; hence the Ø3 flag, assigned to TmpØ, is set.

Queue(s) is a list of the queues in the system. Notice that several are in upper case only, some contain upper and lower case letters. This indicates whether the queue is accessible by the TMP via the CLI or not. Whenever a process accesses a queue, the name of the process is listed in brackets after the queue name in this category. It is very infrequent that you will encounter a process listed after any of the queues displayed except the MXList and MXDisk. These two designate the process using the system list device and disk interface, respec-tively. Note that these are mutually exclusive queues, indicated by the MX prefix. This means that only one process can control this queue at a time. Whenever a file is being spooled to the list device, [SPOOL] would appear after MXLIST, for instance. Similarly, a process performing a disk access would appear in brackets after the MXDisk queue.

3) Program Processing

Whenever a command is entered from the terminal keyboard when the MP/M prompt is displayed, the following procedure results.

The Terminal Message Processor (TMP) reads the input characters and echoes them to the CRT. When a RETURN is entered, the console buffer is passed from the TMP to the Command Line Interpreter queue (CliQ). This "wakes up" the CLI and it parses the command line into the command and command tail. Only one command line can be in the CliQ at a time.

The command can be one of three types: a queue, a COM-type program or a PRL-type program. The CLI reads the command and first checks for a queue name that matches it (ABORT or ATTACH for instance). If a match occurs, the command tail is passed to the queue and control of the console returns to the TMP.

If a queue of that name is not found, the CLI searches for a COM-type program of that name. If one is found, the command is passed to the MLoader queue. This begins a subroutine within MP/M (unique to Z/NET) that looks for an unallocated slave processor. If one is found, the remainder of the command line is parsed, the program is given a process descriptor and the FCBs are created. If no slaves are available, the absolute TPA in the master is checked. A vacancy there renders the same result as though a slave was found.

If neither slave nor absolute TPA is available, CLI searches for a PRL-type file of that name. This is a last ditch attempt to run the program in a memory segment instead of a TPA and the reason for having PRL as well as COM versions of the same file. If an unallocated memory segment big enough to accommodate the program's needs is located, the process descriptor is assigned and the FCBs created.

Once the process has been created, the CLI's job is done. Control of the console passes back to the TMP (in the event the command specified a queue) or on to the process created. In the event CLI was unable to place the program either in a TPA or memory segment, it passes one of the following messages to the CRT.

Abs TPA not free

Reloc seg not free

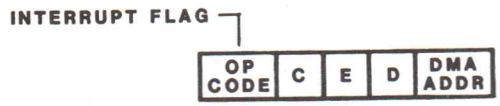
The first message is displayed if the program indicated exists in COM form only. The second message is displayed if the file exists in both PRL and COM form or PRL form only.

This procedure for process assignment is different than that in the Digital Research release of MP/M. Their version looks for the PRL form of a program first in an attempt to execute it one of the memory segments instead of the TPA. Only when this is unsuccessful does CLI check for the COM form of the program. Since Z/NET features multi-processing, it is more desirable to execute the program in a slave with a 62K TPA and a processor of its own than to run it in the master. Consequently, the COM form is sought out first. This also minimizes the need to convert programs to PRL form.

4) Communication between Master and Slave

Once a slave has been assigned a process, it communicates with the master via "mailboxes." There are two for each slave; one for information to be sent to the master and one for information from the master. The mailboxes have the following form

Mailbox from Slave:
6 bytes, FBØ4 - FBØ9



where OP CODE (operation code) is one of the following

	Originate/ Response
Ø - no operation	
1 - BDOS call	O
2 - data sent	R
3 - data in buffer	R
4 - terminate	O

Notice that the high order bit of the OP CODE is also the interrupt flag to the master. The bytes labelled C, E, and D,

contain the values in the Z80A registers of the same name when a BDOS call is made (op code 1) in the program under execution.

Originate/response indicates whether the slave is initiating the operation or responding to an operation begun by the master.

The slaves CP/M emulator PROM intercepts the BDOS calls and supplementary data and stuffs this information in the mailbox for transfer to the master where the call is executed.

Mailbox to slave:
4 bytes, FB0A - FB0D

where OP CODE is one of the following

	Originate/ Response
0 - no operation	
1 - BDOS return	R
2 - send data to HL (A bytes)	O
3 - get data from HL (A bytes)	O
4 - start execution	O

Originate/Response in this case pertains to the master; i.e., whether the master is initiating the operation or responding to an operation initiated by the slave.

The bytes labelled A, L, and H, correspond to the Z80A registers of the same name and contain the values that are returned in response to certain BDOS calls.

Information is passed between the master and slave via direct memory access by the master into or out of the slave's memory space. Since the master only looks at 64K of addressable memory space, the page of memory in the master which corresponds to the source or destination page in the slave is "phantomed out." Concurrently, the slave Z80A is put on hold with the /BUSREQ signal (generated by the master along one of signals SP1 - SP8). In response to /BUSREQ, the slave Z80A generates /BUSAK which enables the slave's S-100 bus receivers/transmitters. Thus, the S-100 bus is only used for inter-processor data transfer when the slave is on hold.

This is contingent on one crucial factor: the destination address in the slave must be in a different page than the source address

in the master (when data is passed from master to slave) and vice versa. The only problem with this approach is the occasional situation when the destination and source pages in master and slave are the same. To accommodate this conflict, a special buffer area has been set aside in the slave to receive or source the data.

Once the slave Z80A has been put on hold, the Z80A block move instruction is used to transfer the data from one location in memory to another. This instruction is used regardless of the amount of data to be moved; whether just 4 bytes (as with the mailbox to the slave) or an entire sector (128 bytes) from a floppy diskette. The only restriction is that the transfer must occur within a time frame that allows the slave's RAM refresh to proceed relatively unimpeded. (When the Z80A is on hold, it does not generate the refresh signal to maintain the contents of the dynamic memory chips.) This restraint is accommodated by the operating system.

APPENDIX A

GLOSSARY

This appendix defines some the more technical words and expressions used in this manual.

APPLICATION SOFTWARE: Programs written to perform tasks like word processing, general ledger, accounts payable, etc. is typically referred to as application software. It differs from utilities in that the tasks are not for disk(ette) and file maintenance.

ASCII: This is an acronym of American Standard Code for Information Interchange. It means that a committee got together and assigned a unique 8-bit binary code (combinations of Øs and 1s) for letters, punctuation marks, numbers and several special characters. Unlike the work of many committees, this one's was accepted and is now an industry standard among micro- and larger computers for information transfer.

BINARY: A method of counting where the only two digits are Ø and 1. All computers work in binary. Data is stored in byte form; groups of 8 binary digits. The definition of HEXADECIMAL has a table that illustrates the relationship between binary, decimal and hexadecimal numbers.

BUFFER: A memory component in a computer system where data is stored temporarily. This is frequently necessary when a peripheral (like a hard disk) has a data transfer rate that is too fast for the processor to accommodate. Another use is the temporary storage of data while it is being transferred from one device to another (system memory to a printer for example) to maximize efficiency by decreasing the number of disk accesses. Buffers are also used in the Z/NET for moving data between the slave and master.

BYTE: A byte is composed of 8 binary bits and is the basic unit of information in microcomputers. There are 256 permutations of Øs and 1s in 8 bits. Unique combinations are used to designate letters and numbers (see ASCII) and processor instructions to build programs for execution in RAM or storage on disk(ette).

COLD BOOT: A master reset of the entire Z/NET system. It is performed by pressing the reset button on the computer cabinet. Subsequently, the MP/M operating system is read from the diskette in floppy drive A: and loaded into the master processor's memory. This is not normally necessary and not recommended especially when other operators are running programs. (It will terminate their program much to their dismay.) It should only be used if the entire system has crashed.

CRASH: This term is a euphemism for other, more harsh descriptors of an unexpected program termination. Typically, it means that a program or hardware bug has been uncovered. The unfortunate consequence is the inability to communicate with the computer (i.e., give it commands) which necessitates a system reset (cold boot) to reload the operating system.

DATA: This term has two meanings. In a general sense, it refers to any information reduced to byte form. In a more specific sense, it indicates informaton that will be processed by a program.

DIRECT MEMORY ACCESS: DMA refers to the process wherein one system component passes data directly into or from system memory circumventing the processor. In the traditional sense, this commonly referred to passing data directly between a peripheral device such as a floppy diskette or a printer and memory. The appearance of multi-processor systems has necessitated an addendum to this definition. In Z/NET it refers to the direct access of the slaves memory space by the master without the interference of the slave processor.

FIRMWARE: Programs permanently stored in devices called PROMs (Programmable Read Only Memory) that perform system related tasks are referred to as firmware. In the Z/NET system, there are two sets firmware: the DOUBLER floppy disk controller has firmware for transferring data between the diskette and system memory and the slaves have it to assist in the transfer of messages between itself and the master. These programs reside in the system memory space so they decrease the amount of RAM (random access memory) available.

FLOPPY DISKETTE: A plastic disk enclosed in a paper package and covered with a magnetic recording surface (similar to the surface of a recording tape). It is used to record data for future retrieval by the computer. Typically, a diskette that has one recording surface (the bottom) holds either 256,000 or 512,000 (256K or 512K) bytes of data, depending upon whether it is formatted in single or double density (see RECORDING DENSITY); a diskette that has two recording surfaces (top and bottom) contains twice these amounts again depending upon the recording density. The surface of the diskette is broken down into tracks and sectors for data storage (see Appendix C: Diskette Format).

FILES: A file is a logical organization of data stored on a diskette. Each file is uniquely identified by a name composed of two parts: the file name (1 - 8 letters or numbers) followed by a period and a file type (1 - 3 letters or numbers). (See Appendix B for file naming restrictions.) The file type is optional and if it is not used the period can be left off. This term is not specific regarding the nature of the contents. For instance, file can be used refer to a program as well as an organized sequence of data used by a program.

FLOPPY DISK DRIVE: The hardware component in the Z/NET system for reading data from and writing data to the floppy diskette. The drive contains a spindle to rotate the diskette at a reasonably constant speed and a read/write head that moves from the outside toward the middle of the diskette. Most drives are capable of writing on one side of the diskette only, the bottom. Some can write on both sides doubling the storage capacity of the unit. Each drive has a name assigned to it. The first drive (the leftmost) is named A: and the second B:. If two more drives are added, they would be labelled C: and D:.

HARD DISK DRIVE: An optional (but strongly recommended) hardware component in Z/NET systems for storing large amounts of data for subsequent retrieval by the system. The mechanism includes both the disk drive elements and read/write head as well as the recording media (called the platter). The platter cannot be removed as with floppy diskettes. The unit used in the M/NET system has a storage capacity of over 20 megabytes (million bytes). The storage space is divided into four 5 megabyte units (logical drives) named E:, F:, G: and H:.

HEXADECIMAL: A method of counting using a base of 16. The numerals include 0 - 9 as in decimal counting (base 10) and append A, B, C, D, E and F. This method is frequently used in microcomputer systems and is indicated in this manual by "hex" or a "H" following the number. The table below illustrates the relationship between binary, decimal, and hexadecimal numbers.

Binary	Decimal	Hexadecimal
0000	0	0
0001	1	1
0010	2	2
0011	3	3
0100	4	4
0101	5	5
0110	6	6
0111	7	7
1000	8	8
1001	9	9
1010	10	A
1011	11	B
1100	12	C
1101	13	D
1110	14	E
1111	15	F

A single hexadecimal digit indicates the status of four binary digits. Thus the contents of a byte of data (8 binary bits) can be expressed with two hex digits.

MASTER PROCESSOR: A single printed circuit board in the Z/NET system that contains a Z80A microprocessor and 64K bytes of RAM. Of this memory space, about 62K is available (i.e., not allocated

to system buffers or firmware) for the MP/M operating system. This is the only board in the system that contains the MP/M O/S and it resides in the upper portion of the memory space. The space below (about 32K bytes) is available for memory segmentation (using the GENSYS utility) for program execution (primarily for PRL-type programs). All input and output between the system (either master or slave), terminals, mass storage devices (floppy or hard disks) and list device (printer) is handled by the master.

MEMORY SEGMENTS: Under MP/M, the master system memory space can be divided into segments with the GENSYS utility. Each segment can run a different program providing there's enough space (the O/S determines the amount of space required before assigning a program to a memory segment). Programs with a file type of PRL (program relocatable) can run in any of the segments. Programs with a COM file type will only run in the segment that begins at memory location 0000. This segment has the status of a transient program area (TPA).

OPERATING SYSTEM: Frequently abbreviated as O/S, this is a control program that coordinates all the functions of the computer system. It monitors program execution, communication between the operator and computer (via the terminal) and data input and output (I/O) between the mass storage units (floppy diskette and/or hard disk) and the computer. Other features commonly found in operating systems provide programs for file management (creating, erasing, transferring, and writing to and reading from files) and for altering the system to configure it to a specific hardware environment. The operating system in the Z/NET system is an enhanced MP/M, the Digital Research Multi-Programming Monitor Control Program. Another operating system mentioned in this manual is Digital Research's CP/M, a single user O/S.

PARSE: This is an expression borrowed from linguistics that means to break down an expression into components that can be analyzed. This references the processing of a command line from the terminal after it has been entered. Parsing it refers to the process of breaking it down into the command and command tail.

PERIPHERALS: All components external to the computer cabinet are called peripherals. In the Z/NET system, this applies to the terminals, floppy disk drives, hard disk drive, and printer. Other devices that can be considered peripherals are modems (devices for transmitting data over phone lines) and magnetic tape drives.

PHANTOM LINE: The phantom line is a signal that disables system memory so that another component can use the specified address. For instance, when the firmware in a PROM is addressed by the processor, the RAM with the same addresses is disabled with the phantom line to prevent two responses. The phantom line in Z/NET systems is also used when the master exchanges data with a slave.

PROCESSOR: In its basic definition, the processor (also called the central processing unit or CPU) is that component that runs the system. It reads programs (in the form of sequential instructions) and data from memory, massages the data, writes the result back to memory, outputs it to a peripheral or whatever. The meaning has been expanded a bit by Micromation to indicate both CPU and RAM on a single printer circuit board as in slave and master processors.

PROGRAMS: A program is a logical sequence of instructions that can be interpreted by the computer to perform a specific task. There are several types of programs mentioned in this manual. The MP/M utilities are one type of programs. They are provided for file and disk(ette) management. Other types of programs are application software written to perform tasks such as word processing, general ledger, accounts payable, etc. In the Z/NET system, all programs are in the form of files. (All files are not necessarily programs, however; see FILES.)

PROM: A Programmable Read Only Memory device is used for the permanent storage of a program necessary for system operation. Although there are several varieties of PROMs, the most commom application is for firmware. In this case, the address spaces allocated to the PROM are part of the total system memory space.

RAM (Random Access Memory): Up to 65,536 (usually abbreviated to 64K where K = 1024) individually accessible bytes for the temporary storage of programs and data. RAM (also system memory or memory) is typically volatile; the values stored in it disappears when the machine is turned off. Memory storage must be temporary so that the contents can change as required when different programs are run (see "PROM" for a description of memory components that cannot change). The Z/NET system features several sets of RAM, one in the master and one for each satellite processor. Each contains the full 64K allowed with minor deductions allocated to resident firmware.

RECORDING DENSITY FORMATS: Data is stored on diskettes in one of two recording densities; single or double. The latter allows the storage of twice as much information as the former. To establish an industry standard, the IBM 3740 single density recording format was chosen. Micromation systems recognize this format for reading from and writing to diskettes (not all microcomputer manufacturers do, however). This feature allows the use of diskettes made by independent software manufacturers who also observe this standard. (The software must also be CP/M compatible though.) There are several double density standards hence no standard at all; compatibility between systems cannot be assured vis-a-vis this format. Appendix C describes the nature of single and double density recording formats.

SLAVE PROCESSOR: Every Z/NET system contains from 1 to 4 slave (also referred to as satellite) processors. This printed circuit board contains a Z80A microprocessor, 64K bytes of RAM and a 2K CP/M emulator PROM. When a program with a COM file type is invoked from the console, it is assigned to one of the slaves. (PRL type program are only executed in the master.) The slaves are dynamically allocated; there are no permanent console to slave assignments. For instance, the operator at console 1 may be using slave 3 for one program and slave 0 for another. This also means that an operator can use all the available slaves for different programs (using the program detach capability) at the same time.

TRANSIENT PROGRAM AREA: Originally in CP/M, the total system memory space was divided into two basic sections. The operating system was loaded into the uppermost section leaving the remainder below it for program execution. Since programs have only temporary use of this portion (once they're done, another program can be loaded there), this section was called the transient program area. It is further defined as that portion of memory that starts at memory address 0100 hex. In Z/NET there are several TPAs, one in the master and one each for every slave. In MP/M, these are used for the execution of programs with a COM file type. (In CP/M all programs have a COM file type.)

UTILITIES: Utilities are the programs provided with the Z/NET system to perform tasks relating to file and diskette management; e.g., a directory (listing) of files on the disk, transferring files from one disk to another, erasing files, renaming files, etc.

APPENDIX B

FILE NAMING AND RESERVED FILE TYPES

File Naming

Although it may not be a frequent occurrence, it may be necessary to create a file and name it. Programs are created in a variety of ways. For instance, the MP/M editor ED can be used to write programs or letters (a word processing system would be better suited to the this task though). Regardless of the method, each file must have a name. Furthermore, it is recommended that the name indicate the nature of the file so that all operators seeing the file in the directory can determine what it does (for programs) or what it contains (for text files).

The file name in MP/M is composed of two parts: the file name and a file type. The former can be 1 - 8 characters long and the latter 1 - 3 characters. The file type is optional, and, if included, separated from the file name with a period. There's a lot of latitude regarding the characters that can be used to name a file. The only restriction is that the following may not be used.

< > . , ; : = ? * [ ]

These characters have significance to MP/M. For instance, the ? and * are used for ambiguous file references; the : is used by STAT and other programs to designate a disk reference; the brackets are used in PIP for special commands; etc. All the other letters, numbers, punctuation marks, and special characters are available for file naming.

The computer doesn't care what you name a file. It won't accept two files of the same name and file type (there are safeguards to prevent such an occurrence), but this is the only restriction. Note that two files can have the same name but different file types and, in fact, this is common. For instance, assembling the 8080 assembly language program DOIT.ASM can render two more files with the DOIT name, DOIT.PRN and DOIT.HEX. Additionally, the EDitor creates a back-up of the original with type BAK when editing an existing file.

Reserved File Types

Although there's a lot of latitude for making up file names, the same is only partially true for file types. The MP/M operating system has several reserved files types that have meaning to the it. For instance, recall that COM and PRL programs are invoked by entering just the file name. When the system receives this input, it parses the first "word" of the command and looks through the directory for a COM or PRL file with that name. In addition,

these files are composed of a special code (in the Intel Hex format) which is also indicated by the file type. The list below designates the reserved files types; subsequent descriptions define their meaning.

ASM	HEX	PRN
BAS	INT	COM
PRL	SPR	PLI
REL	BAK	SYM
$$$	RSP	SYS

ASM: This file type is used to designate a file containing 8080 assembly language for subsequent assembly with the ASM.COM utility.

BAK: The MP/M EDitor, as well as some other word processors, rename the original file with a BAK file type after it's been edited. (BAK is short for back-up.)

BAS: The source file for a BASIC compiler must have a BAS file type. This is similar to the ASM file type described above except a BASIC compiler is used instead of ASM.COM

COM: Those programs that can be loaded and executed directly from the console have this file type. It indicates that the file is in the Intel Hex format and has been LOADed. It connotes that the program is written to start at memory location 0100 hex.

HEX: In addition to the PRN file, a HEX file is created during assembly which contains the Intel Hex code only. This file can be LOADed to create a COM file.

INT: The BASIC compiler creates a file of type INT from a BAS file (it will have the same file name). INT is short for intermediate which intimates that it will only run under a BASIC run time monitor (RUN.COM for instance).

PLI: Source programs written for a PL/I assembler/compiler should have the PLI file type (similar to ASM and BAS file types).

PRL: Similar to a COM program, PRL files are different in that they are written so that they can execute in any page in memory. (PRL is short for page relocatable.)

PRN: As an ASM file is assembled a file with the same name but with the PRN file type it is created. It contains the hex code in the first columns and the source code in the rest of the line.

REL: The REL file type indicates a relocatable program module for insertion in another program.

RSP: This file type designates the MP/M Resident System Processes.

SPR: MP/M is composed of several programs to control terminal, disk and other peripheral I/O and other system processes. These have been given the SPR file type. Since they can vary in size, these programs are also page relocatable. (SPR is short for system page relocatable.)

SYM: The Digital Research Macro assembler creates a file of type SYM containing the symbols referenced. This is for use with SID, the program debugger similar to DDT.

SYS: The MP/M system files have this file type (MPM.SYS for instance).

$$$: The PIP and ED utilities create temporary files as they execute. If a DIRectory is performed during this time (e.g., from another console), the temporary file would have the same file name with the $$$ file type.

APPENDIX C

DISKETTE FORMATS

A discussion of formatting isn't complete without a few words about diskettes.

A floppy diskette is a magnetic media, similar to a cassette tape in this respect, used to store data for fast retrieval by the computer. Unlike a cassette, however, where data (in the form of sound) is recorded in a continuous stream, the area for data storage on the diskette is broken down into tracks and sectors. These are not physical attributes of the diskette (i.e., you can't see them when you look at the recording surface); they are written on the diskette when it is formatted. This type of formatting is referred to as soft-sectored. (An alternate form of formatting is called hard-sectored. This is briefly described below.) Figure C-1 illustrates the apparent and invisible components of the flexible (floppy) diskette.

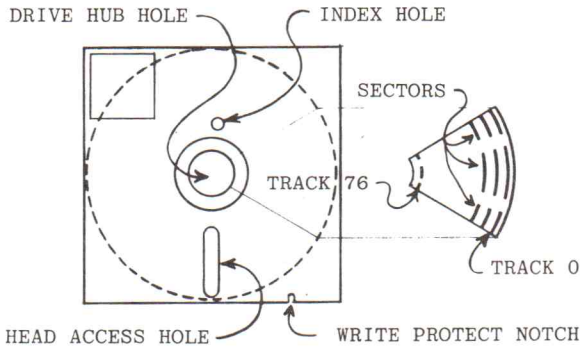


FIGURE C-1 FLEXIBLE DISKETTE

There are 77 concentric circles around the drive hub hole called tracks. Track Ø is the outermost; track 76 the innermost. Each track is subdivided into either 26 or 52 sectors. There are 26 sectors in single density; 52 in double density. Sector 1 appears under the drive read/write head when the index hole in the diskette is within the cut-out in the paper sleeve.

The process whereby the tracks and sectors are recorded on the diskette is called formatting. In this process, the drive read/write head travels in a line from the outside to the inside of the diskette in 77 steps while the diskette spins beneath. Every time the index hole is encountered, the head writes sector identification marks (which indicate the beginning of each sector and the sector number), the data field (which is blank until data is recorded) and a special code called the cyclic redundancy check (CRC). (There are more, complex components as well but they're not relevant to this discussion.

The CRC is used to ensure that the data read from the diskette is the same as that which was previously written. The operation is rather complex but, simply put, it is like multiplying a number (in this case the data stored in the sector) by a constant and writing the answer on the diskette. When the data is read at another time, the same operation is performed again except that the answer is compared with that written when the data was originally recorded. If the two are not the same an error message is printed.

There are two basic formats that can be written on a diskette called single and double density. In single density, there are 26 sectors per track and each sector contains 128 bytes of data. This is the IBM 3740 format and is an industry standard observed by many (but not all) microcomputer manufacturers including Micromation. This means that all application software touted as IBM single density compatible (and CP/M version 2 compatible) will run on the Z/NET system.

There are several standard double density formats used by computer companies (hence no real standard at all). Micromation has selected one with 52 sectors per track each with 128 bytes. Other computer manufacturers may use a different double density format. In this case, the diskettes cannot be read by the Micromation DOUBLER floppy disk controller. This is not a frequent problem since application software is provided in single density. However, if it is, the information can be transferred from one machine to another by first converting it to a single density diskette. (This is contingent, of course, on the other machine observing the standard single density format.)