UNITED TECHNOLOGIES MOSTEK

SOFTWARE PRODUCTS

M/OS-80
FLEXIBLE DISK OPERATING SYSTEM
OPERATIONS MANUAL
MK77984-4

M/OS-80
FLEXIBLE DISK OPERATING SYSTEM
OPERATIONS MANUAL
MK77984-4

T A B L E O F C O N T E N T S

Appendices

L I S T   O F   F I G U R E S

Preface

This manual describes the Mostek operating system (M/OS-80 Version 3). M/OS-80 is Mostek's answer to the growing need for a general-purpose operating system that can take advantage of the wide variety of pre-written applications packages available for microprocessor-based computers. Although newly enhanced and updated, this new offering has several years of field experience and has proven itself in a number of diverse areas.

Many of the concepts and techniques used may be familiar to the experienced user; however, expertise is not required to use this system. Using M/OS-80 does require some simple understanding of running computer programs and how files are referenced: i.e., what a file looks like, how to erase old files, and how to keep proper duplicate copies (backups) of program and data files.

This manual consists of 4 sections and two appendices:

1. - System Startup, Conventions, and Operations
2. - Console Commands and Utility Summary
3. - M/OS-80 Internal Operations
4. - M/OS-80 System Calls

Appendix A - Designers' Development Tool
Appendix B - System Setup

The first section contains that basic information one needs to run the system. Discussions of basic concepts, system operation, disk handling, operator instructions, file conventions, and command syntax are found here. In addition, there is an explanation of system errors and recovery procedures.

The second section is an alphabetical reference to the commands available to the operator. Starting with a table of contents and keyword index, it is intended to be referenced only for commands the operator actually needs, although it pays to skim this section to be aware of what is available.

The third section is for the programmer, especially the machine-level programmer. Here the individual system calls that are used by all applications programs are listed and explained. Included in this section is a chart that describes the inter-system compatibility of M/OS-80 and various other operating systems.

The fourth section gives detailed descriptions of the M/OS-80 system calls available. It describes in detail which registers are used and which need be set up before calling them.

The Designers' Development Tool appendix outlines the firmware program provided with M/OS-80 MK77984-4 systems. DDT is used to diagnose hardware, software and general systems problems in addition to providing a means by which to develop software for the system. Other versions of M/OS-80 (MK77984-5, MK77984-6) do not include this firmware.

The final appendix, System Setup, is an outline and procedure manual that can be used to configure a M/OS-80 system. Included in this section are descriptions of the various part numbers, version numbers and alternate configurations available.

1. M/OS-80 OPERATIONS

1.1. INTRODUCTION TO M/OS-80

M/OS-80 is a disk operating system designed to make user computer interaction as simple and self-explanatory as possible. By using simple names and options, a user can call up a program, tell the program what files to use, what to do with them, and where to put the results. The system allows disk and other devices to look identical for simple sequential operations while still providing for full random use of disk files. A set of standard commands is provided and the user can add commands if desired. For program development, a set of sub-systems is available to allow user programming: these include FORTRAN, BASIC, PASCAL and Assembler. For normal use, many packaged applications are available from many sources.

1.1.1. CP/M Compatibility

M/OS-80 is "upwards"-compatible with the popular 8080 operating system, CP/M (tm) M/OS-80 will run virtually all programs designed to run under CP/M version 1.4 through version 2.2. Programs written for other similar systems, such as Cromemco's CDOS(Tm), are also generally compatible. For a more-complete understanding of the compatibility issue, consult the documentation concerning supervisor calls in the section entitled: M/OS-80 Function Calls.

As a rule, however, if a program is designed to provide system-level support for specific hardware it is not likely to work. Programs that fall into this category are those of a highly-specific nature usually dealing with disk structure or file-directory manipulation. Examples of these are SYSGEN or MOVCPM. Neither of these CP/M programs will function as they do in a CP/M system. There are programs designed to replace these in M/OS-80 which perform the same or, in some cases, additional functions. Programs written specifically for M/OS-80 will not work on CP/M systems if the system calls not supported by CP/M are used or if the enhanced features of the CP/M calls are used.

When attempting to decide if a program will or will not work, first determine if the program was written to operate on a specific version of CP/M or some other operating system. If it is designed to run under CP/M version 1.4 or version 2.2 it should work. If it does not, please notify MOSTEK micro systems or your local MOSTEK Field Applications Engineer with the details of the problem.

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

1.1.2. FLP-80DOS Compatibility

Programs written for the FLP80-DOS operating system will only work if they are written in BASIC or FORTRAN and make no operating system-dependent CALLs. As a general rule, .BIN files are not compatible. There is a utility program called XFLP provided to move data files, BASIC source (ASCII) files, FORTRAN source (ASCII) files, and other non-.BIN files. Assembly language source files can be moved but they must be modified to conform to the new operating system and then be re-assembled. FORTRAN programs must also be re-compiled.

1.2. BASIC CONCEPTS

- Operating system:
  The program (M/OS-80) that handles all physical Input/Output and sets the conditions for user programs.

- Boot:
  The system program designed to bring the system up from a reset or power-up condition to a fully operational state.

- System or Boot Disk:
  A disk that contains the system boot program to be run by the disk hardware when the computer is first started.

- Operator Interface:
  The program (OPI) that has control when the computer is waiting for directions concerning what to do next.

- Command:
  Any program that runs directly on the computer under the operating system. Commonly-used commands are built into the operator interface. Any other program of file type .COM can be a command. User programs can either be commands or run as sub-programs under language commands. An example of this would be a user application written in BASIC.

1.2.1. Warnings

Precautions the user should take to minimize the chance of damage to a disk or data are as follows:

1) Whenever changing diskettes, log-in the new diskette either by doing a "log-in" (Control-C) or by following the directions of a program that requests disk changing.

2) Run "XSTAT" to verify the directory occasionally.

3) BE CAREFUL WITH DISKS. Diskettes are magnetic media and need special care. Observe the storage and handling warnings supplied with the diskettes.

4) NEVER leave a diskette loaded in a disk drive when POWER IS TURNED OFF or ON. Occasionally, random information will get written to the disk while the power supply is switching on or off.

5) Assume that a disk will be damaged occasionally or that data will be lost by human or machine error. For this reason, maintain adequate archives of back-up programs and data. The cost of diskettes and 15 minutes for copying is greatly overshadowed by the time and expense taken to recover weeks of lost data or programs.

1.3. SYSTEM STARTUP / COLD BOOTSTRAP

Starting M/OS-80 from "power-off", or after program failure, consists of pressing the RESET button on the front panel. The MOSTEK computer system will initialize the system console and bootstraps the disk. Operation of this operating system is not guaranteed on any other than MOSTEK computer systems.

1) Place bootstrap disk in system drive "A" which is the right-most drive in a Matrix or the only drive in a Matrix-SDT.

2) Press the reset switch.

3) M/OS-80 should boot up shortly and show:

```
Mostek M/OS-80 Version xx.xx
Serial Number xxxxxx
Configuration number xxxxxx
A.
```

The system is now running.

If a file named STARTUP.COM is present on the system disk, that program will be loaded and executed at this time. If a batch-command file named STARTUP.CMD is present on the disk, the BATCH command program will be loaded, executed, and will use STARTUP.CMD as data.

The version number shown indicates which revision of the operating system is currently in use. The first four digits describe the major software revision level. The second number is the assigned MOSTEK serial number. The last number is assigned at system generation by MOSGEN. (See the MOSGEN Operation Manual to further details.)

It is suggested that a master boot disk, which is write-protected, be kept in a safe place in case the working boot disk is damaged. To prepare a backup Master System Disk, perform the startup functions as described above and copy the master to a blank diskette as follows:

1) Load drive B: with a new diskette. Make sure the write-protect tab is in place. If there is doubt as to the contents of the blank diskette, use FORMAT to erase and re-initialize it.

2) Enter the command:

```
COPY /V B:=A:
```

When asked to mount the disks to be copied to and from, type RETURN.

3) When completed, the COPY program will have moved all files and system programs over to the new disk and then verified that operation. At this time, remove the copy from drive B and use that as the master disk. Store the master in a safe place after removing the write-protect sticker. The disk system hardware will <u>not</u> <u>write</u> to any disk which has this notch uncovered.

1.4. OPERATOR COMMAND FUNCTIONS

Once the system is running, all user interaction will be with the Operator Interface (OPI). This RAM-resident operating system routine performs all required functions by using internal subroutines and/or system command programs loaded in from disk into the user area.

OPI's prompt, shown by the system while running, is the current disk identifier (one letter, A through O) followed by a period:

```
e.g., "A."
```

Operating System Commands may be typed any time this prompt is displayed on the cursor line.

The current disk is indicated by the disk identifier as discussed above. The current or default disk is the disk drive from which files and programs will be taken when no specific disk identifier is given as part of a filename. The current disk may be changed by using the command:

```
d:(return)          [As per section 1.6  ]
```

While typing a command, the input mode is active and certain editing control characters are usable. These characters, as described in section 1.5, allow correction of errors while typing command lines.

The OPI program itself must be available for certain functions. OPI can either be memory-resident or left disk-resident and be loaded only between user functions. If the system cannot find OPI.COM when it is needed, it will ask the operator to specify which disk it is on and then load from the proper disk if not currently loaded.

<u>NOTE:</u>

The M/OS-80 system disk supplied may have all system utilities "system-protected". That is, the attribute byte is set so that users cannot inadvertently erase or damage the vital system files. This also causes all such listings to be "hidden" on DIR listings. The space that system files take up is also not included in any DIR listing not listing such files. To see system-attribute files type:

```
DIR *.* S
```

1.5. CONSOLE INPUT FUNCTIONS

While typing a command, the input mode is active and certain special control characters are usable. A control character is shown by:

^L (Up-arrow followed by the letter).
To type a control character, press CONTROL first and hold it like a shift key, then type the letter.

ACTION DESIRED	CONTROL KEY
Delete the current character and backspace cursor.	- Backspace (^H), - RUBout - DELete
Delete the current character and echo.	- ^X
Cancel the current line.	- ^U
Begin new line without terminating current line.	- ^E
Retype line after many corrections.	- ^R
Terminate a line.	- Return (or ^M)
Return to Operator Interface ("Log-in")	- ^C

Fig. 1-1: Control Characters and Their Functions

* Use "backspace-cursor" on display terminals, and "backspace-and-echo" on printing terminals which have no backspace.
There are some additional special "control(^)" keys. These control printed output to the printer and console:

^P - Sends all console output also to printer. Each occurrence reverses current action.

^S - Stop all processing until another key is struck. This will work whenever device input or output is attempted. Can be used to pause I/O on long output listings.

If output is sent to the printer with one of these functions and the printer is off, all processing will stop until another ^P is pressed or the printer is made ready.

1.6. DISK SELECTION AND LOG-IN

*** READ CAREFULLY ***

Disk-drive Identification:

The M/OS-80 system names the disk drives by single characters: "A", "B", "C"... for all known drives; up to "O" for a maximum of fifteen (15). The current drive is named to simplify use of the system. Whenever a command is given or executed, the current disk drive is implied unless otherwise specified (in the filename).

Current Disk Selection: The command to select a current disk is the disk specifier:

```
        d:   (disk letter, colon)

        e.g., "A:" selects drive A
              "C:" selects drive C.
```

When a disk unit is first accessed, the diskette mounted is logged in, so that the system knows what free space is on the diskette. <u>Whenever a diskette is changed, the new one MUST be logged in by typing ^C (CONTROL-C). This is essential. If not performed, the old disk will still be logged in and the system will improperly write in the new disk directory.</u> (This precaution is advised, regardless of the safeguards built into the system to trigger automatic log-in.)

M/OS-80 is different from CP/M in the area of login. When M/OS-80 is passed a control-C from the operator, it <u>logically</u> logs off all disks by clearing the bit maps of all disks from memory. M/OS-80 does not at that time perform any physical disk I/O as is the case with CP/M. Only when the next requested disk I/O occurs does it get the required directory information and log the new disk in. This permits the user to have empty drives when control-C is struck. M/OS-80 also does not require the system area to be filled in on any but the system disk. That system disk may be removed after boot provided that the program OPI.COM is available on some disk in the system.

1.7. FILE NAMING CONVENTIONS

When referring to devices, symbolic names are used, as shown below: (square brackets refer to optional values)

DEVICE NAMES

3-character name, colon [optional* number 0 through 7]
i.e., XXX:[#]   eg., CON: or PUN:1

DEVICE	NAME	RANGE	REMARKS
Console	CON:	0..7	Control terminal
Reader	RDR:	0..3	
Punch	PUN:	0..3	
Printer	LST:	0..1	Either will work
	PRT:	0..1	
Dummy	DUM:	-	Bit bucket or EOF

Fig. 1-2: Device Names

* For the optional trailing number to have any effect, the IOBYTE device function must be implemented in the system in use.

1.7.1. Disk Filenames

For disk files, a more complex name that explicitly specifies the disk unit and the file is required. A general or ambiguous name is possible for cases which need to match a set of similar or related names, such as finding all files of the same extension or all files beginning with a particular letter.

1.7.2. Specific File References

(File name - FNAME):

Optional Disk Drive Specifier	(A:,B:,C:...)
File Name	(1..8 characters)
Optional File Type	(period, 1..3 char.)

Format: [d:]x[xxxxxxx][.x[xx]]
Example: FILE or A:FILE.ASM

A filename or type can contain any printable character except: . , ; : = ? * /

Although lower-case characters may be used by special user programs, all system functions convert lower case to upper case. If a specific disk drive is not specified, the current drive is implied.

1.7.3. General File Reference (Gname)

A general filename may be used where a number of files are the target of some action. This non-specific filename is made by substituting a match character (* or?) for any character in a filename:

An "*" (asterisk) will match anything to the right of it (in the same field), and "?" (question mark) will match any single character in its position.

Example: A general filename can be used to erase (ERA) more than one file in a group.

```
*.*       matches  EVERYTHING

FIL?.?X   matches  FIL.AX           not FIL1A.X
                   FILA.QX
                   FIL1.XX, etc.

?80.A*    matches  F80.ASM          not F80.M
                   L80.A
                   Q80.ABS
```

1.7.4. File Type

The optional one to three character file type is used to denote the contents of a file. The file type is, in effect, a descriptive suffix to the file name. The choice of the characters is usually up to the user. Some programs assume the filename given has a specific type. This allows one file name to describe the same file in various forms.

For example:

If some assemblers are given the name DATAPGM, they will use:

DATAPGM.MAC for the input file,
DATAPGM.REL for the object output file, and
DATAPGM.PRN for the list output file.

.$$$	- Temporary
.ABS	- Non-executable core image
.ASM	- *8080 assembler source
.ASP	- *AS/Pascal source
.BAK	- *Editor back-up
.BAS	- *BASIC source
.C	- *"C" source
.CMD	- *Batch command file
.COM	- Executable command program (Binary form)
.DOC	- *Documentation text
.FOR	- *FORTRAN source
.HEX	- *Intel hex format (ASCII)
.LIB	- *Source library
.MAC	- *8080/Z80 macro assembler source
.PAS	- *Pascal source
.PRN	- *Print-out
.REL	- *Relocatable module
.RLB	- Relocatable library
.SYS	- System image
.TXT	- *Text file
.Z80	- *Z80 assembler source

(* are ASCII files)

Fig. 1-3: Suggested File Types

1.8. COMMAND SYNTAX

Requesting OS to perform an action requires typing a command to the Operator Interface which then processes it and calls the command. Any command that OPI does not recognize directly is assumed to be an executable module on disk. In these cases, OPI attempts to find a .COM file of the same name on disk. When the system decides that a command is a request for a disk-resident program, the following sequence takes place:

1) First, the disk specified by the name is searched. If no disk is specified, the current disk is used.

2) If the program is not found on the current disk, the master disk is searched as if it were a library. This is disk A until changed by a DCOM command.

If the command is recognized as built-in, or the program is found, the program's executable module is called into memory and the remainder of the command line is passed to the loaded program as control information.

A command starts with the command name. If a command program is specified, then a disk specifier is allowed (A: through O:) but no file type is allowed. The rest of the line depends upon the command but the following forms are standard:

- Any options are preceded by a slash (/), which ends the name.

- Where assignment functions are taking place, "Output (or destination) filename = Source filename" is the form used.

- Commas, equal signs or blanks may separate names. Blanks are generally ignored except to denote separation of names.

- All letters are translated into upper case.

In the command descriptions the symbols used are:

fname	- Specific file reference, i.e., no match characters.
gname	- General file reference,i.e.,? or * allowed.
[-]	- Optional value.
<->	- Description of value.
WORD	- (Underlined) exactly as shown.
{-}	- Choice of values.
(ret)	- RETURN

Unless otherwise noted, all commands are terminated by a RETURN.

1.9. SYSTEM ERRORS

If a disk failure or error occurs, M/OS-80 will print a full error message giving:

```
Failure Code    (2 HEX DIGITS)
Failure Message (Text, if standard message)
Drive           (A..O)
Logical block   (6 HEX DIGITS)
Device Status   (2 HEX DIGITS)
```

A typical message could look like:

```
--------------------------------------------------------
| Disk error 82, Drive B:  Block 000034, Status 01     |
| ---> Disk will not home <---                         |
| Enter R(etry), I(gnore), E(rror return), ^C(ancel) * |
--------------------------------------------------------
```

The cursor will wait at * for an operator response.

Valid operator responses are:

^C	-	Return to OPI and login, thus terminating any active task.
R	-	Retry disk I/O operation.
I	-	Ignore error and return to program with I/O OK message.
E	-	Accept error and return to program with I/O Error flagged.
Other	-	Reject and continue waiting for correct operator response.

NOTE - The operator will be notified only after the disk driver has already done ten internal retries.

Certain errors have standard messages as described in the following table.

81	- Disk not ready.
82	- Disk will not home.
83	- Seek position.
84	- Read error.
85	- Write error.
86	- Read-after-write Error.
87	- Illegal block number. *
88	- Illegal command. *
89	- Disk is write-protected. **

Fig. 1-4: Standard Disk Error Messages

Errors 81 through 86 may be remedied by the operator once corrective action has been performed. For example, after closing a diskette drive door or putting the disk in right-side up, the operator may type R for RETRY so that the system can attempt the disk I/O operation again.

* These errors are due to a non-changing situation. Retry will make no difference, especially since this can only occur due to program or system error.

** This error is due either to a true hardware write-protect tab, or changing a disk at an improper time. If the hardware tab is the problem, it may be replaced and Retry attempted. Otherwise, only ^C (Control C) will be effective.

The M/OS-80 Floppy Driver for the DCF Prom returns a status code which is shown in the disk error message as described above. The following bit table is provided to decode that status byte .

BIT	Error if One (set)
7	Invalid drive, track or sector.
6	Disk unit not ready.
5	Track seek error.
4	Sector not found.
3	CRC error.
2	Data lost.
1	Disk is write-protected.
0	Attempt to read a deleted sector.

Fig. 1-5: Disk Error Bit Decode Table

All other M/OS-80 floppy disk drivers return a status code described below. Disk status bits return detailed reasons for disk errors. The bit placement is as follows:

```
                    BITS
        ---------------------------------
        | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
        ---------------------------------
```

Seek/Home Disk Status

Bit 7	-	Drive is not ready.
Bit 6	-	Disk is write protected.
Bit 5	-	Head is loaded.
Bit 4	-	Seek error.
Bit 3	-	CRC error.
Bit 2	-	Head is positioned at Track 0.
Bit 1	-	Index mark detected from drive.
Bit 0	-	Busy (Command in progress).

Read/Write Disk Status

Bit 7	-	Drive is not ready.
Bit 6	-	Not used (0).
Bit 5	-	Record type 1=delete data mark 0=data mark
Bit 4	-	Record not found (desired track, sector, or side were not found).
Bit 3	-	CRC error.
Bit 2	-	Lost data.
Bit 1	-	Data request.
Bit 0	-	Busy (command in progress).

1.10. NON-SPECIFIC PROBLEMS

1.10.1. Using PROM-Based (DDT/DCF PROM) Systems

Other situations that may occur in M/OS-80 which may appear to be errors are outlined below with the recommended solution. Note, these errors are for DDT/DCF PROM-based systems only.

1) System will not boot. No messages on console. No disk activity.

   System disk in drive "A" (DK0: under FLP-80DOS)?
   System disk inserted correctly?
   Console terminal connected?
   System functional otherwise?

   Remove all disks from system and hit reset.
   Hit return on the console.
   If DDT monitor prompt (.) does not appear, recheck hardware.
   If DDT monitor prompt appears, mount system disk making sure of correct orientation (label side away from drive door latch). Hit reset. Type RETURN on CRT. If no disk activity (LED on drive "A" does not light), re-check disk hardware. Consult MOSTEK.

2) System will not boot. ".DSK ERR" message appears.

   DDT monitor/boot PROM program has detected an invalid disk condition. Check:

   System disk in drive "A" (formerly DK0:)? "A" is <u>not</u> the left drive as in some CP/M systems. In MOSTEK computers drive "A" is always the <u>RIGHT</u> drive. Your installation may have strapped the drives differently.

   System diskette oriented correctly? (Label side up or right depending on orientation of drives.)

   All cables completely attached? Matrix disk cables leading to RMDFSS enclosure may have worked loose.

   Check for complete system power. Some individual units are connected separately to the line power and have individual power switches. Make sure all units are on.

3) System will not boot. "OS.BIN[255] NOT FOUND"message appears.

   The PROM monitor/boot program could not find the system boot program on disk mounted in drive "A". For some reason the system area of the disk you are using is either blank, (has never had WRTSYS move the system boot) or has been damaged such that the system could not accurately read track 0.

4) The system has the DCF PROMs that came with the MDX-FLP board but not the DDT PROM. The system will not boot.

   Both the PROMS are necessary for the system to boot. The DCF PROM supplied with the DDT PROM is not the same as the DCF PROM supplied with the FLP board.

5) Some characters on the ADDS CRT terminal print as asterisk (*).

   To ensure that parity errors are not detected, be sure to set the <u>internal</u> switch on the ADDS to <u>ignore</u> parity. The disable parity switch is found inside the ADDS CRT, not on the back.

1.10.2. Common Non-Specific Problems

1) System takes 15 seconds to boot.

   This is normal. M/OS-80 is considerably more complex than CP/M and takes longer to boot. This boot process is only repeated during power-up and will not occur after the initial boot is completed.

2) System boots but nothing is shown in the DIR listing. Unable to put additional files on system disk due to lack of room even though there is less than 240K showing on DIR listing.

   This is normal. The system files needed by M/OS-80 are marked by the OS with a special "S" (System) attribute which prevents them from being seen on a normal DIR listing. These files also have the "P" attribute set so that they cannot be erased. To see the total listing of files on a system disk, type:

   ```
   DIR *.* S
   ```

3) Cannot seem to erase all files on a disk initialized using FLP-80DOS PIP FORMAT.

   This is normal. Since the directory areas of the two disks are completely different, residual information such as the OS.BIN and OS.CRS left on the FLP80-DOS diskette appear as directory data (although scrambled) to M/OS-80. Some of those characters indicate to M/OS-80 that there are protected directory entries and they cannot be deleted. To correctly initialize a disk use M/OS-80 FORMAT.

7) Some of the directory entries on an old CP/M disk show several hundred K of file space although those files are known to be considerably smaller.

   Some CP/M implementations did not clear a byte marked as RESERVED in the directory FCB. Since M/OS-80 uses those bytes to indicate larger directories for higher-density disks, those erroneous bytes can cause some unusual-looking directory entries. Since M/OS-80 recognizes this problem, the file will be treated as if it were the correct size. To repair the directory entry, copy the file to itself. A correct directory will be created in the process.

1.11. HARDWARE CONFIGURATION

M/OS-80 can be run on a variety of MOSTEK configurations. A summary of the required parameters of those configurations is described below:

Memory	- 64K RAM - MDX DRAM-32A X 2 or SDB-80 + RAM80.
Console I/O	- MDX-EPROM/UART
	or SDB-80 UART-driven console
	or MDX-SIO (Channel A)
	or MDX-SIO2 (Channel A)
Disk I/O	- MDX-FLP
	or FLP80 floppy disk controller.
	or MDX-FLP2 floppy disk controller.
Firmware	- DDT-DCF PROM set (UART or SIO Console).
	or PHANTOM Boot PROM
Printer	- MDX-PIO or SDB-80 parallel port
	or MDX-SIO (Channel B)
	or MDX-SIO2 (Channel B)
Processor	- MDX-CPU1, CPU2, or SDB-80.

Fig. 1-6: Required Hardware Configuration

The EPROM supplied with the MDX-FLP1 or MDX-FLP2 (DCF) is not sufficient to boot the system nor does it provide a monitor/debugger. The DCF/DDT PROM set supplied with Matrix Development systems is required for MK77984-4.

If the system is to be configured as a UART or SIO Console, PIO or SIO Printer with 64K of RAM then part number 77984-4 should be used.

If the user desires to use combinations of hardware and software drivers not mentioned above, the use of MOSTEK's MOSGEN package is recommended. This package permits the reconfiguration of M/OS-80 to accommodate user-written drivers and non-standard hardware. (Refer to the MOSGEN Operations Manual for further details.) Consult Appendix B "System Setup" for further explanation pertaining to hardware configuration.

For more detail on M/OS-80 internals, consult Section 3.0 - M/OS-80 Internal Operations.

2. CONSOLE COMMANDS AND UTILITIES

This section describes each of the built-in commands and program utilities which are provided in the M/OS-80 package.

2.1. SUMMARY OF BUILT-IN COMMANDS

COPY	Disk-to-disk copy
DISKRD	Disk read-only test
DSKDUMP	Absolute disk dump
DUMP	File dump
EDIT	File editor
ERASE	Conditional file erase
FORMAT	Initialize a diskette
LABEL	Set or alter disk labels
LOAD	Load Intel Hex or .COM files
PPG	Prom Programming utility
PRINT	Printer file list
SORTDIR	Prepare sorted directory file
SPLIT	Split large text files into smaller pieces
SPOOL	Start print spool task
STRIP	Strip parity, nulls & rubouts from text file
SXFER	Single disk file transfer
WRTSYS	Write system boot to disk
XDIR	Extended directory
XFER	File copy or concatenate
XFLP	Move ASCII files from FLP80-DOS diskettes
XSTAT	Directory sizing & validator

Fig. 2-1: Utilities (.COM files)

```
COPY      [<destination>:=<source>:]
DISKRD    [d:]
DSKDUMP   <file>
DUMP      [<file>]
EDIT      [<file>]
ERASE     <gname>
FORMAT
LABEL     [<disk:<command>=<value>]
LOAD      <file>
PPG       <file>
PRINT     <file> [/<options>] [<title>]
SORTDIR   <outfile>=<gname>
SPLIT     <file> [<max size (IN k)>]
SPOOL     <file>
STRIP     <file>
SXFER     <gname>
WRTSYS    d:[file]=d:[file]
XDIR      [d:]
XFER      [ [/<opts>] <file>=<file>[,<file>,<file>...] ]
XFLP      <FLP80-DOS filename>
XSTAT     [d:]
```

Fig. 2-2: Command Syntax Summary

@	Batch monitor.
ATRIB	Set file attributes.
DBAT	Set batch default disk.
DCOM	Set default command disk.
DIR	Directory listing.
ERA	Erase file.
EXEC	Execute specific address
GTOD	Set or read time of day.
OPIRES	Make OPI resident.
OPINRES	Make OPI non-resident.
OREV	Report the revision, system version, and OPI.
PAUSE	Wait until key is pressed.
REN	Rename a file.
SAVE	Save memory image on a file.
TYPE	Print a file on the console.

Fig. 2-3: Built-in Function Summary

@	[<file> <Parm 1> <Parm 2> ....]
ATRIB	<general file reference> <attributes>
DBAT	d:
DCOM	d:
DIR	<gname> [S]
ERA	<gname>
EXEC	[address in hex]
GTOD	[date]
OPIRES	
OPINRES	
OREV	
PAUSE	[/B]
REN	<gname>=<gname>
SAVE	fname <number of 256-byte pages>
TYPE	<gname>

Fig. 2-4: Built-in Function Syntax

M/OS-80 UTILITY	CP/M EQUIVALENT
COPY B:=A:	PIP B:=A:*.* (+SYSGEN)
DISKRD	(None)
DISKDUMP	(None)
DUMP	DUMP
EDIT	EDIT
ERASE	(None)
FORMAT	(None)
LABEL	Partially by:STAT
LOAD	LOAD
PPG	(None)
PRINT	(None) - See TYPE
SORTDIR	(None)
SPLIT	(None)
SPOOL	(None)
SXFER	LOAD/SAVE
WRTSYS	SYSGEN
XDIR	(none) - See DIR
XFER /a <name>=<name>	PIP <name>=<name> [a]
XFLP	(None)
XSTAT	STAT

Fig. 2-5: CP/M - M/OS-80 Utility Program Cross Reference.

@ name.CMD	SUBMIT name.SUB (See notes)
ATRIB	(None)
DBAT	(None)
DCOM	(None)
DIR	DIR
ERA	ERA
EXEC	(None)
GTOD	(None)
OPINRES	(None)
OPIRES	(None)
OREV	(None)
PAUSE	(None)
REN	REN
SAVE <file> #	SAVE # <file>
TYPE	TYPE

Fig. 2-6: CP/M - M/OS-80 Function Cross Reference

2.1.1. CP/M Conventions Not Generally Transparent

XSUB - Batch file submission that passes data to application program from .SUB file. Not supported under M/OS-80.

STAT - Provision to change "I/O" byte and assign multiple Consoles, List devices, Punches etc. Not implemented in M/OS-80 I/O drivers.

USER - Provision to tag each directory with a user number and change that number with the "USER" command. Not implemented under M/OS-80.

MOVCPM - Program to create new configuration of CP/M. Supported by MOSGEN system.

2.1.2. Notes On SUBMIT File Conversion

Generally, SUBMIT (.SUB) files from CP/M programs can be easily converted, provided that they are not designed to run with the newer CP/M 2.2 XSUB utility. Parameters passed to the SUBMIT program are preceded in the command file text with a $. In M/OS-80 batch (.CMD) files, the parameters are preceded with a ^ symbol. Changing these symbols should correct most compatibility problems. See the notes on PIP and SAVE below.

2.1.3. Notes On PIP

M/OS-80's XFER program performs all PIP functions and several more. However, there is a minor change in the syntax. M/OS-80 expects the parameters passed to it to be seen just after the ! prompt or just after the XFER command and preceded by a "/". PIP expects parameters to follow all other parameters enclosed in []. An example is shown below:

```
    XFER /V A:=B:FILE.NAM       ::=     PIP A:=B:FILE.NAM [V]
                               -or-
    XFER /AF A:NEW=B:OLD        ::=     PIP A:NEW=B:OLD [AF]
```

2.1.4. Notes On SAVE

The SAVE command performs the same function in M/OS-80 as it does in CP/M. The only difference is in the syntax. M/OS-80's SAVE expects the number of blocks to save as the last argument in the command line while CP/M's SAVE expects it as the first argument.

```
    M/OS-80 = SAVE NAME.COM 5

    CP/M = SAVE 5 NAME.COM
```

2.2. @ - BATCH MONITOR

```
SYNTAX:  @
         or
         @ [/AB..] <fname> [<parm1> <parm2> ... <parm9>]

OPTIONS: /A..H - Specify disk for batch file.
```

DESCRIPTION:

The Batch monitor provides a means where a number of console commands may be stored in a file and executed sequentially by the Operator Interface. Any command that is typed into OPI may be put into a file built by a text editor. If the sequence of commands is to be used only once, then a temporary file can be built by BATCH.

Note - for convenience, the actual command used is "@" (at-sign).

BATCH has two modes:

Local Mode: Where the commands are typed in, immediately before use, and only used once.

File Mode: Where commands are taken from an ASCII file, with type .CMD. Parameters may be used in the file and an actual parameter from the command line is substituted for each formal parameter of the same number.

To STOP a batch sequence, type RETURN on the keyboard during the initiation of a command. Batch streams started during the boot process (STARTUP.CMD) cannot be stopped.

Local mode:

The local mode is indicated by not specifying a filename. BATCH will prompt with a (!) and allow the operator to enter the commands. A blank line will terminate the command entry.

```
e.g., @ (return)           (indicate local)
      ! DIR *.*  (return)
      ! TYPE HELLO  (return)
      !  (return)          (blank line)
```

File mode:
In file mode, BATCH takes its commands from a file with the name specified by the first name on its command line and with type .CMD. If the file does not exist on the current disk, the DBAT disk is searched for the file.

Parameters are inserted wherever ^# appears. (Two separate characters, up-arrow, followed by digit of 0..9) Parameter 0 (^0) is the command filename. 1..9 are specified in the command line. If an ^ (up-arrow) is needed in a command, use ^^ (only one is passed on).

When the command line is given, normally each word after the filename is a parameter. Complex parameters, including spaces, may be enclosed in single quotes ('). If not enough parameters are given, then the missing parameters are replaced with null strings. If parameters are given where none are requested, they are ignored.

Options:
The letter option /d (d=A..H) allows the user to specify the disk on which the batch work file will be written. This is normally the "A" disk. The DBAT command may be used to change this default assignment thus allowing for systems with an "A" disk which is write-protected or full. The disk assigned to hold the batch command file must not be removed until the batch is completed.

Batch command files may contain a command that calls another Batch file and return to the previous Batch file after the lower level file is completed. Otherwise, processing will stop when the second file is exhausted. The nesting level is limited to a maximum of 128 commands pending at any one time.

On lower level batches, the disk-select (/d) option is ignored.

FILES:
The @ command is built-in to OPI, and requires OPI be reloaded. Batch files must have the type: .CMD. A separate work batch file is created with name "$$$$.CMD" on the DBAT: disk.

ALSO SEE:

DBAT	- Assign location of Batch work file.
STARTUP.CMD	- Auto-start batch file (at boot time).

MESSAGES:

BATCH FILE ERROR can be caused by:

- Could not create batch file.
  (Disk full, or write-protected.)
- Requested Batch file could not be found.

PARAMETER ERROR: "^" is followed by other than ^,0..9.

DIRECTORY FULL: While creating work file.

DISK FULL: While writing work file.

WARNINGS:

BATCH logs in all disks at its start (^C).
Maximum of 128 commands may be active. Do not count non-evaluated second-level batches.

2.3. ATRIB - SET FILE ATTRIBUTES

SYNTAX:

```
ATRIB <general file reference> <attributes>
ATRIB gname {+PRWSU}      - Set file attributes
ATRIB gname 0             - Clear file attributes
```

DESCRIPTION:

M/OS-80 files may be assigned attributes or type information to provide additional protection and flexibility. Possible attributes are:

```
P - Permanent
R - Read-protected
W - Write-protected
S - System
U - User
+ - Add new attributes
```

Permanent files cannot be renamed or erased.

To SET attributes, give all appropriate letters: e.g., "PRW" would set all matched files to permanent, read, and write-protected.

To ADD more attributes to those already present, use "+", (e.g., "ATRIB *.COM +R) would add read-protection to all ".COM" files.

To REMOVE all attributes, use a general name, with attribute of "0" (zero). (I.E. "ATRIB *.COM 0")

Read or write-protected files are respectively prohibited from being read or written to. However, both types can be erased, and read-protected files can be executed. Read-protected files can only be executed (if the type is also .COM). They cannot be dumped or typed.

Attributes "S" and "U" are for file identification. Commands DIR and XDIR skip files of type "S", unless the "S" option is specified.

FILES:

ATRIB is a built-in command.

MESSAGES:

??? - If an unknown letter is specified for an attribute.

2.4. COPY - DISK-TO-DISK IMAGE COPY

SYNTAX:

```
COPY <output disk>:=<source disk>:
COPY [/V] <output disk>:=<source disk>:
```

DESCRIPTION:

COPY moves all sectors from source disk to output disk. This includes the system, directory, and data areas. Unformatted portions of uninitialized disks cannot be successfully copied but the system can be instructed to ignore the resultant errors, thus permitting recovery of a partially damaged diskette.

The program requests mounting of both disks to check for compatible disk directory areas and density. Disks of different density or number of sides cannot be copied due to the different placement of directory information. Use XFER /V B:=A:*.* to copy all files from drive A to drive B and use WRTSYS B:=A: to copy the system area if necessary.

If both disk drives specified are the same, then a single disk copy operation is assumed with the appropriate "Change to disk xxxx" messages.

Ie. COPY /V A:=A:

If the /V option is requested, a second pass is made which verifies the two disks to be the same, sector for sector.

Copy will copy any type of IBM 3740 <u>formatted</u> disks, not just M/OS-80 disks.

FILES:

COPY.COM.

MESSAGES:

>> Compare error <<	- Bad block written, or could not read original.
Load SOURCE disk	- Program requesting input disk.
Load DESTINATION disk	- Program requesting output disk.
Incompatible disks	- Disks are not of the same size and type.
COPY completed	- Copy pass done.
Beginning VERIFY	- Starting verify pass.

2.5. DBAT - SET BATCH DEFAULT DISK

SYNTAX:

```
     DBAT d:
     Directs batch operations to use disk 'd'
     for:  a) Work file ($$$$.cmd),
           b) .CMD file searches.
```

DESCRIPTION:

This command controls the batch function (@) in OPI, instructing it where to look for files.

FILES:

DBAT is built-in to OPI.

ALSO SEE:

@ - Batch processor.

MESSAGES:

??? - If no disk specified.

WARNINGS:

If a non-existent disk is entered, the current disk will be used.

2.6. DCOM - SET DEFAULT COMMAND DISK

SYNTAX:

```
DCOM d:
```

Directs OPI to use disk 'd' as master disk for command program searches.

```
DCOM
```

Directs OPI not to search for commands if they do not exist on the current disk.

DESCRIPTION:

When a command (.COM) program is run from OPI, the current disk is searched first, then the library disk is searched. DCOM is used to designate which disk will be considered the "library" or master disk, (the last disk to be searched). DCOM with no parameter inhibits the library search.

OPI, if non-resident, will also be loaded off of the library disk. If the library search is inhibited, then OPI will always be loaded off of the current disk.

FILES:

DCOM is built-in to OPI.

WARNINGS:

If a non-existent disk is entered, the current disk will be used.

NOTE:

Only requests for .COM files are recognized by OPI for library searches.

2.7. DIR - DIRECTORY LISTING

SYNTAX:

```
DIR
or
DIR  <gname> [S]
```

DESCRIPTION:

If no filename is given, *.* is assumed, which will display all files, of all types except files of type "S" which will not be listed unless a name is given and a second parameter of "S" is given.

The listing format is:

```
Name   Type Size/Extents  Attributes (2 across)
```

If disk is labeled the disk name is printed:

```
Disk label is :.........
```

Following the label, (if present), is the summary line:

```
    Total files (total extents) ......   Total Kbytes
```

DIR Lists disk files with size (in Kbytes) and number of directory entries used, with a summary of total space used printed at the end.

The summary is always printed. If the listing is cancelled prior to completion, the summary will include only those files actually listed.

Pressing SPACE will pause the listing. RETURN will cancel the listing.

Extents:

Each 16 Kbytes, 1 extent, of each file takes one directory entry, of which there is a maximum of 64 (normally).

A labeled disk may have more than 64 entries. Run LABEL to see if the disk has been labeled for more than 64 directory entries.

Attributes:

Shown as:

P - Permanent	
R - Read-protected	W - Write-protected
S - System	U - User

FILES:

DIR is a built-in command

ALSO SEE:

LABEL	- To see size of directory.
XDIR	- To obtain a multi-column sorted listing.
ATTRIB	- To get more detailed information on attributes.

WARNINGS:

"DIR .COM" will act as if no name was typed. Use "DIR ?.COM" or "DIR ?.?" to get files with no name, only a type. If this fails, use DSKDUMP to fix the directory.

The amount of space listed in the summary does not include the amount of space (if any) taken up by the system-attribute files not listed.

2.8. DISKRD - DISK READ-ONLY TEST

SYNTAX:

```
    DISKRD [<disk>:]
```

DESCRIPTION:

Reads all specified sectors of specified disk unit or current unit if none specified.

Disk driver error messages will print block address of error.

FILES:

The file DISKRD.COM is the program.

MESSAGES:

The map printed by the program is:

```
Disk "d"
"h"    "t"    /........../........../......
 |      |     \         /
 |      |     10 sectors per group
 |      -----Track being tested
 ------------Head being tested
```

Periods indicate successful sector read. If an error occurs, respond to the disk-error message block.

If the "E" option is given as a response to the disk drive error message, an "*" is printed. If the "I" option is chosen, a "." is printed.

2.9. DSKDUMP - DUMP DISK BLOCK IN HEXADECIMAL

SYNTAX:

For direct DISK operations:
DSKDUMP d: where: d is disk A..O

or

For FILE operations:
DSKDUMP fname

Program requests BLOCK:

Responses are:
("n" indicates a hex number of up to 6 digits)

N	- Increment and display next block.
(return)	- Increment and display next block.
^	- Decrement and display next block.
-	- Decrement and display next block.
n	- Selects block to display.
nI	- Indicates an interleaved read is to be done (only if not a file).
Mn,v...	- Modify buffer at offset value "n", with values specified by "v".
Q	- Terminates program.
^C	- Terminates program.
S	- Show contents of buffer.
W	- Write buffer back where read from.
Wn	- Write buffer to indicated block.
Xn	- Select eXtent indicated.
Yfilename	- Select a new file to edit.
Yx:	- Select the disk x: for direct edit.
Y*	- Select "SYS.DIR" (directory).
Ln	- Locate block which contains hex address n.

DESCRIPTION:

DSKDUMP allows reading or modifying sectors of a file or disk sectors directly. Each block (sector) requested is read into a 128-byte buffer which is then displayed. The buffer may be modified, re-displayed and written back to the disk to the same or another sector.

The prompt "BLOCK:" requests a command, which may be a disk block in hex or a single-letter command.

For a normal single-density/sided (IBM format) disk, blocks may be 0 thru 7B8(hex). Entering a block number causes that block of the file to be read into the working buffer and displayed on the screen.

For non-file direct-disk I/O, the value may be suffixed with "I" to indicate an interleaved read is to be done, e.g., 35I would get block 35(hex) interleaved, or the first block of the directory on a standard IBM format disk. The block read will be displayed. To stop the display, press any key.

If a command letter is given instead of the hex value, various operations may be performed. The commands are:

"W"	- Write buffer to disk.
"M"	- Modify buffer area.
"S"	- Show buffer area.
"Q"	- Quit the program.
"X"	- Go to new Extent.
"Y"	- Select dataset for direct edit.
"L"	- Locate a relative address in file.

No change is written to the disk until a "W" is done. On entry, DSKDUMP will login the disk. On ^C or Q (exit) after editing a file, the file will be closed.

The command formats are:

```
    Write:  W  - Writes buffer back where it was read from.
            W# - Writes buffer back to given block.

              e.g. W23 writes to block 23
                   W35I writes to block 35 interleaved
```

Modify: M#,<value>[,<value>...]

The first hex value is offset in the block to start modifying. Each value is put in successive locations. A value may be either a hex value or a character string ("ABCDE" would enter 5 bytes). All lowercase letters are converted to upper case.

```
e.g.
M20,23,"ABC",10   enters  into  20H..24H   the
bytes: 23H,41H,42H,43H,10H.
```

Show: S

S displays the contents of the buffer. It can be used to review a block after modifying it and before writing it out.

Quit: Q

Q ends the DSKDUMP program without writing to disk by exiting to the operating system.

Extent: Xn

Allows changing of the 16K-byte extent DSKDUMP is working on. Xn only has meaning when a file is being examined. Extents are numbered starting at 0.

Select: Yfilename
or Y*
or Yx:

Select allows reselection of file being displayed without leaving DSKDUMP program.

Yfilename will access the file (filename) directly on the disk for modification or display.

Y* will access the directory portion of the disk directly for modification or display. Y* is very useful to retrieve those files that were accidentally erased. **Caution must be used whenever the directory is being altered.** User must be very familiar with the directory format before attempting to alter disk directory.

Yx re-logs another drive for user. The value for x can be any disk A-x where x is the last drive in the system.

Locate:

Lx (Where x is a valid hex address occurring somewhere in the file currently in use.) Locate is used to find a specified address relative to the start of the file in use. When an address is entered, DSKDUMP will display the block and the offset into that block for the specified address. In the following example, underlined text is entered by the user.

Example:

```
DSKDUMP TEST.COM
Mostek M/OS-80 Direct Disk Dump (DSKDUMP 00.09)
Block: ?L123
OFFSET IN BLOCK=23

  Block:00001
  00000:00 00 00 ....etc.
```

The address 123 hex is located in block 1 + 23 bytes offset.

FILES: DSKDUMP.COM is the program.

MESSAGES:

```
*** File not found *** - No input file.
**** I/O Block Error ****- Either past EOF or I/O error.
```

2.10. DUMP - DISPLAY FILE IN HEX

SYNTAX:

DUMP <fname> [skip count]
Where skip count is the number of blocks to skip before beginning the dump of the file.

DESCRIPTION:

The DUMP utility is designed to give an ASCII and hexadecimal dump of any file. DUMP displays a relative offset every 16 bytes. The offset is relative 0 to the beginning of the file. The format consists of the offset address, sixteen bytes of data and the ASCII representation of each byte displayed to the right of the screen. ASCII code is shown where the character is printable or "." where the character is unprintable. Only the low 7 bits are checked for the ASCII display.

Press the space bar to pause the display.
Press any key (other than the space bar) to stop the dump.

FILES:

DUMP.COM is the program.

WARNINGS:

Some terminals decode certain ASCII character sequences as control codes and may attempt to perform the function while DUMP is displaying the ASCII equivalent of the HEX codes DUMPed, causing unpredictable results on the CRT screen.

Like any program, DUMP may over-run the through-put capability of the CRT terminal, thus causing a "torn" or otherwise scrambled display.

2.11. EDIT - ASCII TEXT EDITOR

SYNTAX:

```
EDIT  start EDIT with no file name.
EDIT <FILENAME.EXT> edit existing file FILENAME.EXT
EDIT <FILENAME.EXT> creates new file  if  FILENAME.EXT
     does not exist.
```

COMMAND SUMMARY: (See description for details)

Edit Functions:

An	Advance n lines.
Bn	Backup n lines.
Cn/s1/s2/	Change n occurrences of s1 to s2.
Dn	Delete n lines.
En	Exchange n lines.
Fn	Turn on or turn off print flag.
G file	Get file and insert into current file.
I	Insert lines of data.
Ln	Go to line number n.
Mn	Macro: Place command string into alter nate command buffer 1 or 2.
Pn file	Put n lines out to file.
Q	Quit, save all editing and return to DOS.
Sn/s1/	Search for n occurrences of s1.
T	Insert lines at top of file before first line.
Vn	View n lines on the console.
Xn	Execute alternate command buffer n (1 or 2).

Control Functions:

```
RUBOUT     -    Delete last character entered.
Backspace -    Delete last character entered.
^I (Control I)
           -    Move over to next "tab" stop.
```

2.11.1. Introduction

EDIT is designed to assists the user in creation and modification of assembly language source programs and text documentation by permitting random access editing of ASCII files. EDIT is designed for use with MOSTEK's M/OS-80 Operating System

EDIT permits editing of files on a line or character basis. Whole lines and character strings embedded within lines can be easily accessed, changed, deleted, or added to an existing or new file. The size of the file to be edited is limited only by diskette capacity. All I/O operations to the diskette are transparent to the user.

EDIT keeps one level of back-up upon completion of an editing session by performing all write operations to a work file (extension .$$$). Upon successful completion of the save, EDIT will rename the original file to FILENAME.BAK erasing any previous matching file with the extension of .BAK. EDIT will then rename the .$$$ file to the original FILENAME.EXT. In this way, the previous contents of the original file are kept in the .BAK file if needed later.

2.11.2. Description

In the following sections, (CR) indicates that the carriage return key is to be pressed. All user input is underlined. User input which must be entered exactly will be shown as UPPER CASE letters. User inputs which are in either UPPER or lower case will be shown in lower case.

Command Line:

EDIT prompts for a command with an asterisk (*). The user may then enter commands via the console keyboard. Correction of user input using rubout, backspace, and line delete, is supported by the operating system. Several commands may be entered on one line. Blanks and commas are ignored on command line input. A command line is terminated by a carriage return and may have up to 80 characters in it, including the carriage return.

All commands consist of one character followed by an optional operand. The operand may be separated from the command by zero or more blanks or commas for readability. The operand may be a decimal number in the range 0-9999 which specifies the number of lines upon which the command is to operate. Alternatively, the operand may be two decimal numbers separated by a dash (-). In this case, the command takes effect on lines numbered from the first number in the operand through and including the second number. If the number operand is not entered, EDIT assumes a value of one. (Except for the 'F' command which defaults to 0. See detailed description for more information.)

EXAMPLE:

```
        V
        V5
        V 5
        V,5
        V42-45
        V 42-45
        V,42-45
```

2.11.3. Starting EDIT:

EDIT filename(CR)
: Filename is the name of the diskette file to be edited on the currently-selected disk. The file may **not** have an extension of COM, BAK, or $$$ and must conform to M/OS-80 file-naming conventions.

EDIT responds with the following message:

```
        M/OS-80 TEXT EDITOR V1.2
```

If the user does not enter the filename when starting, EDIT requests filename to edit during initialization.

```
        FILENAME TO BE EDITED (TO ABORT, HIT CNTL-C)?
```

The user then types in the name of the file to be edited. If the user types ^C (CONTROL C), EDIT will return back to M/OS-80 without any file activity. If the file does **not** exist, EDIT requests:

```
        CREATE A NEW FILE? (Y/N)
```

If a new file is to be created, the user must respond with "Y". "N" indicates that a new file is **not** to be created. If the user types "N", the system will again ask for the name of the file to be edited. This is useful for those who type the wrong name to edit and don't want to go through the QUIT and have an empty file left on disk.

Example:

1. EDIT MYFILE(CR)
   User selects to run EDIT to edit the file named 'MYFILE' on the currently-selected disk.

2. EDIT(CR)

```
M/OS-80 TEXT EDITOR V1.2

FILENAME TO BE EDITED (TO ABORT, HIT CNTL-C)?MYFILE(CR)
```

3. EDIT NEWFILE(CR)
   If the file does NOT exist on the diskette, then the Editor outputs the following message on the console:

```
   CREATE A NEW FILE? (Y/N) Y
   ***NEW FILE
```

   Editor indicates that a new file is being created.

2.11.4. Detailed Command Descriptions

An - ADVANCE

Format:
    An or an (n= decimal number 0-9999)

ADVANCE is used to move the line pointer forward (toward the end of the file) a specified number of lines. If the operand n is not entered or it is zero, the pointer will be position to the next line in the file. If the 'F' flag is not set, the line which is accessed is printed on the console after this command.

Example:

```
    A(CR)
         The next line with its line number is printed on
         the console.
    A5(CR)
         User advances 5 lines from current pointer and
         prints line number and the line.
```

Note:
If the user attempts to advance the line pointer beyond the end of the file, an end-of-file indicator message will be printed on the user console and the line pointer will be on the last line of the file.

```
    (I.E.)
    A9999(CR)
          -User advances over a large number of lines.
    XXXX LAST LINE OF FILE
    ***EOF
```

Bn - BACKUP

Format:
    Bn or bn (n= decimal number 0-9999)

BACKUP is used to move the line pointer backward (toward the beginning of the file) a specified number of lines. If the operand n is zero or it is not entered, the pointer is positioned to the previous line in the file. If the 'F' flag is not set the line which is accessed is printed on the console.

```
Example:
B(CR)
     -User backs up over one line in the file.
XXXX A LINE OF INFORMATION
     -Editor prints the line number and the line.
B4(CR)
     -User  backs  up 4 lines from current position  in
     the file.
XXXX-4 SOME LINE.
     -Editor prints the line number and the line.
```

If the user attempts to back the line pointer past the start of the file, a top-of-file indicator will be printed on the console. The line pointer will be on line number 0001.

```
(I.E.):
B9999(CR)
***TOF
0001 FIRST LINE OF FILE
     Editor  prints  top-of-file  indicator  and  first
     line of the file.
```

Cn - CHANGE STRING

Format:

```
     Cn/string1/string2/
     cn/string1/string2/
```

Where n indicates the number of occurrences to change (0 - 9999), string1 represents the characters to be changed, string2 represents the substitute or new characters, and / represents a delimiter character which does not appear in either string.

This command changes the next n occurrences of character string1 to character string2 starting with the current line. Any character which does not appear in either string1 or string2 may be used as a delimiter. All three delimiters must be identical, with the exception that the last delimiter may be a carriage return. If the operand is zero or if it is not entered, then only one occurrence of string1 will be changed. In this case, only the current line will be searched in order to locate string1. If string1 is not found in the current line, then the Editor issues a warning prompt ('?') and a new command prompt (*). The line pointer will stay on the same line.

If the operand n is greater than 1, the search for occurrences of string1 occurs in a sequential manner starting with the current line. Each line which is changed is printed on the user console. After all changes are done, the line pointer will be on the last line that was changed. If the nth occurrence of string1 is not found before the end of the file is encountered, the last line of the file is printed by EDIT, as well as an end of file indicator (***EOF). The line pointer will be on the last line of the file. If string2 has no characters in it, character string1 will be deleted each time it is encountered by the change command.

Example:

```
     1 .V(CR)
          -User views current line.
     XXXX THIS IS A RECORD
          -Editor prints line number and line.
     2. C /THIS/THAT/(CR)
          -User enters change command.
     0009 THAT IS A RECORD
           -Editor prints the updated line.
     3. C/IS/WAS(CR)
          -Note that a carriage return for the last delimit-
          er is allowed.
     0009 THAT WAS A RECORD
     4. C /WAS //(CR)
          -This is the method used to delete characters in a
          line.
     0009 THAT A RECORD
     5. C 2 /T/V/(CR)
          Note that blanks can be inserted between the
          command and operand and string definition to make
          the command more readable.
     0009 VHAV A RECORD
     6. C4/VHAV/THAT/(CR)
          This is a multiple change request which will
          search forward in the file starting with the cur-
          rent line.
     0009 THAT A RECORD
          Editor prints out every line that is changed.
     0043 LAST LINE OF FILE
     ***EOF
          Editor reached the end of the file before any
          more changes could be done.  An end-of-file indi-
          cator message is printed.  The line pointer is on
          the last line in the file.
```

Dn - DELETE

Format:

```
     Dn
     dn
     Dn-m
     dn-m
          (n= decimal number 0-9999)
```

DELETE is used to remove the specified lines from the file. If the operand is not entered or is zero, only the current line is deleted. Note that line numbers are assigned dynamically as editing progresses. This means that lines in a file essentially get renumbered each time one or more lines are added to or deleted from the file.

Example:

1. D(CR)
   User deletes the current line from the file.
   The line pointer is on the next line in the file.
2. D4(CR)
   User selects to delete 4 lines starting with the current line from the file.
3. D4-15(CR)
   User deletes lines numbered 0004 through and including 0015 from the current file.

En - EXCHANGE

Format:

```
     En
     en
     En-m
     en-m
               (n= decimal number 0-9999)
```

EXCHANGE command replaces the specified lines with new lines to be inserted via the DATA MODE. It is exactly equivalent to the command sequence:

```
     Dn             -Delete lines
     B              -Backup one line
     I              -Go to DATA MODE
```

Fn - PRINT FLAG

Format:

```
     Fn
     fn
          Where  n=0  will  inhibit printing after  all  but
          the V-VIEW command,  and n= anything other than  0
          will  allow  printing after all change  or  access
          commands.
```

EDIT normally displays any lines which are accessed or changed. Thus, the following commands print out a line: An, Bn, Cn, Ln, Sn, Vn. In order to reduce the print time on a slower device (such as a teletype), this command can be used to inhibit printing on all of the commands except V-VIEW.

G file - GET FILE COMMAND

Format:

```
     G FILENAME.EXT
     g FILENAME.EXT
          -Where  FILENAME.EXT is the name of a file on  the
          selected disk.
```

GET is used to obtain lines from a given file and insert them in sequence FOLLOWING the current line. All lines in the file are read. The file which is read is not altered in any way. The GET command can be used with the P - PUT command to move blocks of text around within a file being edited (See P - PUT command for example).

I - INSERT COMMAND

Format:

```
     I
     i
```

INSERT is used to add data to the file being edited or to build new files. The inserted lines always FOLLOW the current line. After the command is entered, the Editor responds with the message:

```
     ***DATA MODE
```

The user then enters data ending with carriage returns. EDIT prompts with the line number for each line to be inserted. To terminate the insertions, the user enters a single carriage return. Note that blank lines must be entered as 'space carriage return' since a single carriage return terminates the DATA MODE. After the user terminates the DATA MODE, EDIT prompts for a new command (*). Lines can be inserted before the first line of a file by using the T - INSERT AT TOP command.

Example:

```
     I(CR)
          User selects data mode to insert lines into the
          file being edited.
     ***DATA MODE
          EDIT responds with message.
     0004 THIS IS AN INSERTED LINE.(CR)
          The line number being entered is printed by EDIT.
          The user then enters the line of data.
     0005 (CR)
          User terminates DATA MODE with a carriage return.
```

Note that correction of entered data can be accomplished while it is being typed just as in tM/OS-80 command line. Inserted lines can be up to 128 characters long.

Ln - GO TO LINE NUMBER n

Format:

```
     Ln
     ln
          (n= decimal number 0-9999)
```

The LINE command positions the line pointer to line number n. If the operand is zero or it is not entered, then line number 0001 is accessed. Any line number can be accessed from any position in the file. The line which is accessed is printed on the console unless the 'F' flag is set to 0. If the line number cannot be found (because it is larger than the last line number in the file), the pointer will be positioned at the last line in the file and an end-of-file indicator message will be printed.

```
Example:
1.  L10(CR)
     User accesses line number 0010.
0010 THIS IS A LINE OF DATA
     Line number 0010 is printed with its line number.

2.  L2001(CR)
     User selects line number 2001.
0943 LAST LINE OF FILE
     Editor prints last line of file.
***EOF
     Editor prints an end-of-file indication.
3.  L1(CR)
     User selects line number 1.
***TOF
0001 FIRST LINE OF FILE
EDIT  responds with top of file indicator,  the  first
line of the file, and its line number.
```

Mn- MACRO

Format:

```
M1(CR)
m1(CR)
M2(CR)
m2(CR)
```

MACRO allows a command string to be entered into one of two alternate command buffers (labeled '1' and '2'). The alternate command buffers will accept character strings of 80 characters or less. EDIT responds with the following prompt:

Example:

```
M1(CR)
1*S /OLD/ D1 B1 (CR)
     The user enters into alternate command  buffer 1
     the commands which:
                  1. Search for the 1st occurrence of the
                     string 'OLD', starting with the next
                     record.
                  2. Delete that record.
                  3. Backup one record.
```

Pn file - PUT

Format:

```
Pn FILENAME.EXT
pn FILENAME.EXT
Pn-m FILENAME.EXT
pn-m FILENAME.EXT
     Where n and m are decimal numbers and filename is
     the name of a file on the selected disk.
```

The PUT command is used to output one or more lines to a file on disk. This can be used to break up a given source module. It can also be used with the G - GET command to move blocks of text around in a file being edited. If the operand is not entered or it is zero, then only one line will be output. Lines of text which are output by the PUT command are not deleted. They may be deleted via the D (DELETE) command after the PUT command is used. The filename specified must not be the same as the current file being edited. **If the file already exists on disk, it is erased before any lines are output to it.** After the PUT command is used, the file output to the disk remains on the disk.

Example:

P25-30 TEMP(CR)
: User outputs lines 25 through 30 to a file called TEMP. The space between the number and the filename is not required.

D25-30(CR)
: User deletes lines 25 through 30 from file being edited.

L1(CR)
: User accesses line number one.

G TEMP(CR)
: User reads lines from file TEMP and places them after line number one. This effectively moves lines 25-30 to just after line 1. The space between the command and the filename is not required.

Q - QUIT

Format:

```
Q
q
```

The QUIT command returns control to the Operating System. The original file will be backed up using the same filename with an extension '.BAK'. All of the editing will be saved in the file under the original file name. QUIT can be done at any time during the course of editing which leaves the original file untouched.

Sn /string/ - SEARCH FOR STRING

Format:

```
Sn /string/
sn /string/
```

Where n is the number of occurrences to be found, string represents any set of characters which is to be searched for, and / represents a delimiter character which does not appear in the string.

SEARCH scans the file, starting with the NEXT line, for n occurrences of the character string between the delimiters. Each line containing the string will be printed on the console. The pointer is positioned on the line containing the nth occurrence of the string. If the nth occurrence of the string cannot be found before the end of the edited file, then EDIT issues an end-of-file indicator (***EOF). This command always searches forward (toward increasing line numbers) in the file.

Any character which does not exist in the search argument string may be used as a delimiter. The second delimiter may be a carriage return. If the operand n is zero or it is not entered, then only the first occurrence of the string will be sought.

Example:

1. S/ORD/(CR)

   User selects to search forward in the file, beginning with the next line, for the string 'ORD'. Only the first occurrence of the string is sought. The blank between the command and the string is not required.

   0021 SOME ORDERLY DATA

   EDIT prints the line number and line when the string is found. The line pointer is on line 0021.

2. S10/9AH/(CR)

   User selects to search for and view the next 10 occurrences of the string '9AH'.

   ***EOF

   EDIT encountered the end of the file and found no occurrences of the string. The end-of-file indicator is printed.

T - INSERT AT TOP

Format:

```
     T
     t
```

The TOP command inserts data lines at the top (start) of the file BEFORE the first line in the file. See the I-INSERT command for proper usage.

Vn - VIEW

Format:

```
     Vn
     vn
     Vn-m
     vn-m
          (n&m= decimal number 0-9999)
```

VIEW prints the specified lines on the console device. The line pointer is updated to the last line printed. If the operand n is zero or is not entered, then only the current line is printed.

Example:

```
     1. V(CR)
          User views current line on the console.
     0009 THIS IS A LINE
          EDIT prints line number and line.

     2. V3(CR)
          User views current line plus two more.
     0009 THIS IS A LINE
     0010 THIS IS NEXT LINE
     0011 THIS IS ANOTHER LINE
          EDIT prints 3 lines on the console.  The line
          pointer now points to line 0011.
     3. V3-4(CR)
          User selects to view lines 3 through 4.
     0003 SOME LINE OF DATA
     0004 NEXT LINE OF DATA
          EDIT prints lines 3 through 4.
```

Xn - EXECUTE

Format:

```
     X1
     x1
     X2
     x2
```

EXECUTE exercises the commands stored in the alternate command buffer numbered 1 or 2. After an alternate command buffer has been executed. Control is return to EDIT which prints a prompt for a command (*). The alternate command buffer is not destroyed during the operation. IF n is equal to 0 or is not entered, then alternate command buffer 1 is selected.

Example:

```
M1(CR)
*1S /OLD/ D1 B1 (CR)
X1(CR)

0010 FIRST OCCURRENCE OF OLD.
     'OLD' is located and the record is deleted.
0009 LINE NUMBER 9.
     Backup command prints its record.
```

NOTE: 1. The pseudo-macro command capability is executed by the 'M' and 'X' commands. The user puts his macro command string into alternate buffer 1 or 2 and executes that macro string via the 'X' command.

2. The default command in M1 is 'V23'

2.11.5. Editing Large Files

Editing of large files is no different than editing small files. All commands are fully functional. However, diskette access may be required for certain operations and a slight delay may be apparent before EDIT responds.

2.11.6. Editor Messages

If the user enters an unrecognizable file name, a syntax error will be indicated and EDIT will return to DOS. If the user enters an unrecognizable command, then the Editor will print a question mark and another command prompt:

Example:

```
R20(CR)
?*
```

All I/O errors to and from disk result in appropriate error messages. The original file should be backed up on another disk before using EDIT.

(Continued)

EDIT will prompts with several other messages as editing progresses:

***NEW FILE - New file is being created rather than editing of an already existing file.

***DATA MODE - Lines of data are to be entered rather than Editor commands.

***TOF - Top of file (beginning of file) has been encountered.

***EOF - End of file has been encountered.

***END OF EDITING - EDIT has successfully completed. Control is then returned to the DOS Operating System.

***END OF WINDOW. USE 'ADVANCE' TO SEE NEXT LINE - Occurs only with the VIEW command. Follow the directions.

2.11.7. Sample Editing Session

The user is urged to follow the steps given here to become acquainted with EDIT.

```
    EDIT NEWFILE(CR)
         User selects to run EDIT to create a new
    file.
    M/OS-80 TEXT EDITOR V1.2

    CREATING A NEW FILE( N = NO, OTHERS = YES )?(CR)
    **NEW FILE
         Editor indicates that a new file is being created.
    **DATA MODE
         Editor prompts for data lines to be input from the
         console device. User begins keying in a program.
    0001 ; A SIMPLE SAMPLE PROGRAM(CR)
    0002 LD A,(LAB1)(CR)
    0003 LD E,0(CR)
    0004 CALL SUB1 ;SOME COMMENT(CR)
    0005 LOOP LD (HL),0 ;STUFF ZEROS(CR)
    0006 INC HL(CR)
    0007 DNZ LOOP-$ ;LOOP FOR ALL(CR)
    0008 END(CR)
    0009 (CR)
    User terminates DATA MODE.
B99V20(CR)
    User backs up to beginning of file and views all lines.
```

(Continued)

```
***EOF
     Editor indicates end of file encountered.
L7(CR)
0007  DNZ LOOP-$ ;LOOP FOR ALL
     User views line 7 and observes an error.
C /DN/DJN/(CR)
     User modifies the line.
0007  DJNZ LOOP-$ ;LOOP FOR ALL
     Editor prints the changed line
Q(CR)
     User terminates the editing session.
```

2.11.8. Disk Full Handling

If the disk fills up during an EDIT session, the EDIT program will print ther error message: "DISK FULL". To attempt a recovery of this condition, the user will need to erase an un-needed file or ABORT the session thus losing all editing input for this session. The original file will be left intact.

```
***** DISK FULL *****
RECOVERY OPTIONS?   (D = DELETE A FILE
                     L = DIRECTORY LISTING
                     C = CONTINUE CURRENT COMMAND
                     A = ABORT EDITING          )?
```

The user can use these four command to:

1. List current disk directories (L).
2. Delete a unused file on the current disk (D).
3. Continue current command (C).
4. Abort this editing session (A).

2.11.9. Editor Definitions

SOURCE - ASCII characters comprising a computer program, or other textual (non-hexadecimal) data.

FILE - Diskette file which contains the ASCII SOURCE.

LINE - Single source statement which ends with a carriage return.

LINE POINTER - Position in the source where the next action of EDIT will be initiated.

CURRENT LINE - Line in the source pointed to by the LINE POINTER.

LINE NUMBER - Decimal number of a line, beginning at one (0001) for the first line in a file and increasing sequentially for each line. The maximum line number allowed is 9999 (decimal). Line numbers are assigned dynamically as editing of the file progresses. This means that when lines are added to or deleted from a file, all lines are automatically renumbered.

INSERT - Insertion of one or more lines in a file immediately following the current line. Inserted lines are assigned sequentially increasing line numbers.

DELETE - Removal of one or more lines from a file.

2.12. ERA - ERASE FILES

SYNTAX:

```
     ERA <gname>

     ERA *.COM   - erases all command files.
     ERA *.BAS   - erases all BASIC programs.
```

DESCRIPTION:

Erase all files whose name matches.

'ERA *.*' will clear the disk directory after confirming the request with a single character 'Y' or 'y'. ERA will not erase any files which have been given a 'P' attribute.

Erased files can be recovered using DSKDUMP if it is done <u>immediately</u> after the ERA command completes. To recover an erased file, enter DSKDUMP using SYS.DIR as the filename. Next, page through the directory until you see the FCB(s) for the file erroneously erased. Change the first byte, (which should be E5 for erased files,) to 00 and re-write the block. Remember to find all extents for files with multiple extents and perform this same operation. Do not attempt to do this if any files have been changed, added, or moved because alteration of the directory in this manner would cause a serious cross-allocation of file blocks.

FILES:

ERA is a built-in command.

ALSO SEE:

ERASE - Conditional erase.
ATRIB - Permanent (erase-protected) files.

MESSAGES:

Erase Cancelled - output after ERA *.* and 'Are you sure?' N.

2.13. ERASE - ERASE SELECTED FILES

SYNTAX:

ERASE <gname> - Erases files whose name matches after operator acknowledgment.

Responses:

Y	- Erase the file whose name is displayed.
Control-C	- Exit ERASE.
Other	- Leave file and proceed to next file.

DESCRIPTION:

ERASE will remove all files whose name matches the general name given. Before each file is erased, the name is displayed for the operator to ask if the file should be removed:

- A single character is expected.
- 'Y' will cause the file to be erased.
- Control-C will exit the ERASE program.
- Any other character will cause the file to be left alone.

Erased files can be recovered using DSKDUMP if it is done <u>immediately</u> after the ERASE command completes. To recover an erased file, enter DSKDUMP using SYS.DIR as the filename. Next, page through the directory until you see the FCB(s) for the file erroneously erased. Change the first byte, (which should be E5 for erased files,) to 00 and re-write the block. Remember to find all extents for files with multiple extents and perform this same operation. Do not attempt to do this if any files have been changed, added, or moved because alteration of the directory in this manner would cause a serious cross-allocation of file blocks.

FILES:

ERASE.COM is the program.

MESSAGES:

"Cannot erase" - Problem with disk, since file was found on search, yet the system could not find it for the erase.

"File not Found" - If a file is erase-protected (attribute "P") then file will not be seen by erase program and thereby not erased.

2.14. EXEC - EXECUTE AT ADDRESS

SYNTAX:

```
     EXEC address
     -Where address is a valid hex address (0 - FFFFH)
```

DESCRIPTION:

EXEC allows user to directly execute any valid hex address directly from M/OS-80. EXEC is especially useful to execute a program, (which the user has possibly altered) without re-loading the file from disk. EXEC could also be used to execute a user program which did not begin execution or reside at the TPA address of 100H. EXEC is used in the PROM-based system to enter Mostek's DDT PROM debugger program by typing **EXEC E11D.**

Example:

```
EXEC 100
```

User executes the program residing at 100H.

```
EXEC E11D
```

User executes the DDT program (PROM based system ONLY).

FILES:

EXEC is built-in to the operating system.

WARNINGS:

Since EXEC DIRECTLY executes the address given it does NOT scan the command line or set any buffers. It does NOT have the same function as the implied running of a .COM file from the operating system.

2.15. FORMAT - INITIALIZE M/OS-80 DISKETTES

SYNTAX: FORMAT

DESCRIPTION:

FORMAT is used to write 3740 sector marks and clear all existing data on floppy disks. It should be used on all new diskettes to ensure that all directory areas are cleared and that all areas can be read later.

> Note: Be sure to cover the diskette write-protect notch (if present) before attempting to initialize any diskette.

Once the program has started the following dialog takes place:

```
Mostek MDX Formator (FORMAT 00.04)
Drive to be formatted (A...x) [^C to exit]?
```

The operator then types a letter (A thru x where x is the last drive in the system) to indicate which drive will contain the disk to be formatted.

The system responds:

```
Format disk for (S)ingle or (D)ouble density?
```

The operator types S for single density (IBM 3740) format or D for double density (MFM) format.

```
To format disk in drive (n)
      Load disk and
           -- Press F to format, Press ^C to exit --
```

If the controller is set up for 5.25" drives, the following question is asked:

```
Enter tracks/side (in decimal) >
```

If the operator selects double density and the controller does not support double density, the following error message is printed on the console:

```
*** Cannot Format double density with this controller.
```

(Continued)

The program then asks:

```
    Enter sector interleave (in decimal) >
```

The operator should respond by hitting carriage return if the disk is to be used on a foreign (i.e. non MOSTEK) system. If the disk is to be used exclusively on MOSTEK supplied hardware, we suggest a sector interleave of 3 for 8" single density, and 5 for 8" double density. For 5.25" disks, the default is designed for optimum throughput.

The system then types the following message:

```
Formatting Single-sided Single-density
          8" disk in drive n, port xx, unit y.
      Double-      Double         5.25
```

Where n is the drive name, xx is the floppy controller board's address and y is the unit address on the controller for the drive.

Once the system has identified the drive as to density and number of tracks and sides, the following dialog takes place:

```
    Whole (w) or Partial (p)?
```

At this point the system attempts to determine if the entire diskette is to be formatted or just a part. To format an entire disk, type w. To format a segment of the disk, type p.

If w is typed, the system proceeds to format the disk. If p is typed, the following message is displayed:

```
    Enter Starting track (in decimal) >
```

At this time the operator is requested to enter the track location to start formatting the diskette.

Once the Starting Track has been established, the system displays the following message:

```
       [tracks left= nn], press return for all
    Enter Number of tracks to format (in decimal) >
```

At this time the operator is requested to enter the number of tracks to format from the point specified in the starting track prompt. If the operator hits return at this point all remaining tracks from the starting track to the end of the disk will be formatted.

(Continued)

Example: If the operator decides to format track 0 and 1 on a disk the value for starting track would be 0 and the value for number of tracks to format would be 2.

During the formatting process, the track being formatted is printed on the console device as follows:

```
(000) 123456789 (010) 123456789 (020) 123456789 .....
```

Once all tracks have been formatted, the following message is printed on the console:

```
-- Diskette completed --

Drive to be formatted (A..x) [^C to exit]?
```

If desired, the process can be repeated on the same or another disk. To repeat, enter drive number. To end the program and return to the operating system, hit control-C.

FILES:
    FORMAT.COM

2.16. GTOD - GET/SET TIME OF DAY

SYNTAX:

```
     GTOD
     Report current date and time. Time is only shown if
     not zero.

         GTOD mm/dd/yy hh.mm.ss
              Sets date and time as specified.

         GTOD mm/dd/yy
              Sets date as specified.
```

DESCRIPTION:

GTOD is used to set the date and time for printing and general informational purposes. The date items are separated by slashes (/), and the time items by periods (.). Date and time are separated by a space.

The date is required, but time is optional. If the time is not entered, it will be set to 0.0.0.

FILES:

GTOD.COM is the program.

MESSAGES:

Hit return to continue - given to allow synchronization with a clock.

2.17. LABEL - READ AND SET DISK LABEL

NAME:

LABEL - Set and Determine Disk Label.

SYNTAX:

```
LABEL
```
To display label configurations of all disks.

```
LABEL ?
```
To get a summary of commands.

```
LABEL <disk>:<command>=<value>
```
To set label.

Command & value	Description	Default
D=<name>	Set disk name for double density diskettes. Sets C=16.	None
DD=<name>	As above but sets S=128	None
N=<name>	Set disk name (1..8 chars) For use on single density diskettes only.	none
C=<cluster>	Set disk cluster size (In blocks)	8
S=<size>	Set directory size (number of entries)	64
B=<begin>	Set directory beginning location (on disk)	52
P=<password>	Set the password (1..3 chars)	none

DESCRIPTION:

Normal M/OS-80 floppy disks have 64 files with file allocation in 8-block (1K-byte clusters). For many small files, double-density disks or business applications, a disk label provides additional information about the disk to the system. The label contains the name of the disk, the revised cluster size and the size of the directory. Since the label is the first entry in the directory, **it should only be written on a clean disk.**

The command "LABEL" will display all known information about all disk units. A DIR listing must be requested for each diskette before information gathered by the label program is accurate.

Format of the display is as follows:

```
Disk Label   Dir size   blk/cluster   dir start   current state
```

Disk label	- The name given to the disk.
Dir size	- The size of directory given.
blk/cluster	- The number of blocks per cluster.
dir start	- Starting block (in decimal) for the directory.
current state	- The current state of disk. The following table describes the bit definitions.

Bit Definition for the Current-State Byte.

Bit 0	-	This Directory is a BIG directory.
Bit 1	-	This Unit MUST have a BIG directory.
Bit 2	-	Reserved
Bit 3	-	Reserved
Bit 4	-	This Disk Has a Label.
Bit 5	-	Reserved
Bit 6	-	This is a Double-Sided Diskette.
Bit 7	-	This is a Sub-Directory.

The command "LABEL d:xx=<value>" will set the label on the specified disk to the given value. For a full list of label parameters, type "LABEL ?" and the program will list all options. For example, LABEL B:N=TEST1 will name the disk in B to "TEST1".

LABEL logs in the disk after changing the label to update the OS with the new label. Various disk parameters can be set or reset; however, changing file values on a disk containing files may cause improper access to previous data and should be done only with extreme care.

Certain values are most effective if set following these guidelines:

Cluster size: 8 minimum, other values: 16,32,64, 128 max.

Directory size: Steps of 4, from 32 to 252, with 4 entries per block and an integer number of clusters.

Note: To set the name on a dual-density diskette, use the D or DD command. Using the N command will create a directory which will be unusable.

FILES:
LABEL.COM is the program.

MESSAGES:

Must have directory entry 1 clear -

When setting a new label, the first directory entry must be clear for the label. LABEL should only be used on freshly-initialized disks.

If a dual-density diskette is improperly formatted (using the N= command, a DIR listing may produce the following message:

```
>>Error - # of clusters exceeds 254 of a small directory
```

If this message occurs, reformat the disk and relabel using the D or DD commands.

WARNINGS:

LABEL's disk report does not force each disk to be logged in or report if disk is logged in; hence, disk name may be incorrect if disk was just changed.

Various disk parameters can be set or reset; however, changing file values on a disk containing files may cause improper access to previous data. In particular, changing the start of the directory or the file cluster size will cause problems and must be avoided.

2.18. LOAD - LOAD PROGRAM INTO MEMORY

SYNTAX:

LOAD <filename>

DESCRIPTION:

LOAD is used to bring .COM (binary image) and .HEX (INTEL Hex) files into the transient program area. No attempt is made to execute the file brought in, as in the implied run command. Control is returned to M/OS-80 and the operator once the operation is completed. OPI must be resident when using this program (See OPIRES). To make room for the loaded program, LOAD relocates itself at the top of the transient program area prior to actual execution.

If the filename to be loaded is entered without extension, LOAD will search the disk directory for a file with a .COM extension, and next for a .HEX extension. If neither are found, LOAD will report "File not Found" and reenter the operating system. Files with other than .HEX or .COM extension cannot be loaded. If any other extension is used, an invalid extension error will be reported.

When a .COM file is loaded, the loading range will be displayed. These addresses are the memory locations where the file was placed in memory. If a .HEX file is loaded, the execution-begin address and range of the loaded program is displayed. If an attempt is made to load a program into a memory location occupied by the LOAD program or M/OS-80, the highest address available to the user will be displayed with an error message. LOAD will not load a program which has addresses below 80H.

ALSO SEE:

DDT Appendix for further information on debugging programs once loaded.

FILES:

LOAD.COM is the program.

```
EXAMPLE 1:     A.LOAD TEST.COM
               Loading File TEST.COM
               Loading Range 0100 to 07FF
               A.(System prompt)
```

```
     EXAMPLE 2:      A.LOAD TEST.HEX
                       Loading File TEST.COM
                       Execution Address is 0187
                       Loading Range 0130 to 0400
                     A.(System prompt)
```

MESSAGES:

Invalid Extension -
Only .COM and .HEX files can be loaded.

Attempted to Overlay onto LOAD Program -
The file being loaded attempted to overlay LOAD program's RAM area. LOAD runs just below M/OS-80 at the top of the transient program area (TPA).

Attempted to Load in Memory below 80H -
Memory below 80H must be preserved for system use.

Invalid Record Encountered -
An invalid record in a Hex file was encountered. INTEL Hex format rules were violated.

Checksum Error -
A checksum error in a Hex file was encountered.

No End-of-File -
The Hex file did not contain an End-of-File record.

Attempted to Read Past End-of-File -
An End-of-File record was encountered in the middle of a file.

Highest Memory Available to User XXXX -
When a program attempts to load into memory beyond the available transient program area, this message advises of that upper limit.

WARNING:
OPI must be resident (See:OPIRES).

2.19. OPI - OPERATOR COMMAND PROCESSOR

SYNTAX:

OPI is the program that processes the commands:

DIR	HALT	SAVE
ERA	@	OPIRES
REN	DBAT	OPINRES
TYPE	OREV	PAUSE
DCOM	ATRIB	EXEC

OPI starts all user programs and is returned to after a user program completes.

DESCRIPTION:

For general use, OPI may be made RAM-resident with the resultant loss of about 3k of user space. See OPIRES-OPINRES.

FILES:

OPI.COM is the program.

MESSAGES:

```
Program  not  found - Could not find the program  requested.
Load error          - Program was too big to load.
???                 - Built-in function error.
```

WARNINGS:

OPI must be available or the system WILL NOT RUN !!

2.20. OPINRES - MAKE OPI NON-RAM-RESIDENT

SYNTAX:

OPINRES - Forces Operator Interface to become non-RAM-resident.

DESCRIPTION:

OPI is normally loaded as a program between command program executions, thus requiring it to be on a currently-loaded disk at all times. OPI may be made RAM-resident for ease of use or made non-resident to save space.

The commands @ (batch) and OREV require OPI to be RAM-resident. In addition, the SAVE command requires that OPIRES be executed prior to the SAVE. OPIRES must be done before the step that creates the user area that SAVE wants to write to disk. If OPI is not RAM-resident at that time, SAVE is disabled.

FILES:

OPINRES is built into OPI.

ALSO SEE:

OPI, OPIRES

MESSAGES:

??? - If OPI is already non-resident.

2.21. OPIRES - MAKE OPI RAM-RESIDENT

SYNTAX:

```
    OPIRES  - Forces Operator Interface to become  RAM-resi-
              dent.
```

DESCRIPTION:

OPI is normally loaded as a program between command program executions, thus requiring it to be on a currently-loaded disk at all times. OPI may be made RAM-resident for ease of use or made non-resident to save space.

The commands @ (batch) and OREV require OPI to be RAM-resident. In addition, the SAVE command requires that OPIRES be executed prior to the SAVE. OPIRES must be done before the step that creates the user area that SAVE wants to write to disk. If OPI is not RAM-resident at that time, SAVE is disabled.

FILES:

OPIRES is built into OPI.

ALSO SEE:

OPI, OPINRES

MESSAGES:

??? - If OPI is already resident.

2.22. OREV - REPORT SYSTEM IDENTIFICATION NUMBERS

SYNTAX:

```
    OREV
              Operator Interface (OPI xx.xx-xx.xx)
              Mostek M/OS-80 Version (xx.xx)
              Serial Number xxxxxx
              Configuration number xxxxxx
```

DESCRIPTION:

Reports the version number of the system, the serial number, the user configuration number and operator interface being run using the following format:

```
              Mostek M/OS-80 Version  AA.BB
                                      /  /
        1. Major OS Version # ------    /
                                       /
        2. OS Revision #--------------

                                   Serial Number xxxxxx
                                                /
        3. Mostek Assigned User Serial #---------

                               Configuration number xxxxxx
                                                /
        4. User Assigned Configuration #---------
```

FILES:

OREV is built-in to OPI, but requires OPI to be reloaded.

2.23. PAUSE - SUSPEND OPERATION

SYNTAX:

PAUSE
Put the system into a wait state until key pressed.

PAUSE/B
Put the system into a wait state and ring the console bell until key pressed.

DESCRIPTION:

Used to sound the bell and/or wait for operator checking during a batch operation. To stop the batch, press RETURN TWICE - once to exit PAUSE and once to stop the batch processor.

FILES:

PAUSE is built into OPI.

ALSO SEE:

@ (Batch)

MESSAGES:

"Strike any key to continue."

2.24. PPG - PROM-PROGRAMMING UTILITY

SYNTAX:

```
PPG
or
PPG <filename>
```

DESCRIPTION:

PPG is a software package designed to run with the PPG 8/16 hardware supplied by MOSTEK. It is NOT designed to run on any other hardware. For a complete description, refer to the **PPG Utility Operating Procedure** supplied with the PPG 8/16 hardware. The PPG software provided, gives the user the ability to program 2708's, 2716's, or 2758's using M/OS-80. A general description of the functions available follow.

The following commands are available with PPG:

L -	Load a specified PROM.
P -	Program data to PROM.
M -	Display or Update memory.
R -	Read data from a PROM.
V -	Verify a PROM against memory.
Q -	Quit or exit from the program.

These commands give you the ability to copy already programmed PROMS, program PROMS with user programs, verify questionable PROMS against the original file or a master PROM, or manually change single bytes of a PROM (by reading them into memory, making changes using the M command then programming a new blank PROM). Complete information for all commands are given in the PPG Utility Operations Procedure manual.

FILES:

PPG.COM

2.25. PRINT - PRINT TEXT FILE

SYNOPSIS:

```
PRINT filename [/<options>] [title]
```

Optional specifiers are control options & title.

B - Bend	- If width is specified, then bend excess characters to next line (otherwise chop off rest of line).
C - Center	- Center heading within margins given.
D - Date	- Output date at right side of heading.
D+ - Date/time	- Output date and time on heading.
F - Form-feed	- Use form-feed to force top-of-page (instead of using LF's).
Ln - Lines	- Lines per page (Default = 66).
Mn - Margin	- Will set a left margin indent of n characters.
N - None	- No formatting (i.e., no paging / same as LOWO). Overrides all options except /U.
T - Terminal	- Send output to terminal (instead of to list device).
U - User	- Use user-defined options (over-rides all other options).
Wn - Width	- Width of page (Default = 0:Don't Care if line too long for printer).

Options are run together after the slash, i.e. /W72B will cause 72-column-wide printing with bending.

DESCRIPTION:

Specified text file is sent to the printer with given controls.

The whole filename must be specified, including type. If page width is specified, each page is limited to given width, by truncating or bending (B).

If paging is specified, a title line and 1 blank line precedes the text and 6 blank lines follow on each page. Pages may also be triggered by form-feed (^L) or the alternate ^K.

Tabs are expanded at 8 character stops (1,9,17...) and all other control characters are printed as ^l.

User-set defaults:

The user may build standard options into the PRINT.COM file by filling them in starting at 104H & specifying /U as the option, as follows:

```
A.DSKDUMP PRINT.COM (return)
Block: 0 (return)
Block: M4,'(enter print options here)'
Block: W
Block: Q

A.PRINT TEXT.FIL /U (return)
```

Using the user options totally overrides any other options.

2.26. REN - RENAME FILE

SYNTAX:

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

DESCRIPTION:

This function renames all files matching the old name with the new name. If a specific is used, a search is made for the existence of the new name. If any are found, no renaming will take place. However, once renaming has started, no further tests are made; thus, duplicate directory entries may be formed when the matched characters in new name are placed in the oldname.

```
e.g.,  REN *.Z80=*.ASM        (ok)
       REN SALLY.ASM=*.ASM    (bad)
```

If no specific drive is mentioned, the default drive is used. If a specific drive is mentioned in the new filename, then that drive is assumed on the old filename regardless of drive specified or default drive. If no drive is specified on the new filename and a drive other than the default drive is mentioned on the old filename, a "File not found" error will occur.

FILES:

REN is a built-in command.

MESSAGES:

File already exists	- If rename could cause a name that is already in the directory.
File not found	-File to be renamed cannot be found.

2.27. SAVE - SAVE USER AREA

SYNTAX:

```
SAVE fname  <Number of 256-byte pages>
```

Save memory from the user area, N pages (256 bytes each).

REQUIRES:

OPIRES before the user area contents are created.

DESCRIPTION:

The user area starts at 100(HEX) and ends at the bottom of M/OS-80. Since most system functions preserve the user area, programs can be saved on disk if done immediately. If the saved file is given the extension .COM and it is an executable module, it could become a user command-program.

The high 2 digits of the highest hex address to be saved are the number of pages when converted to decimal.

```
e.g.,  SAVE TEST.COM 17   Saves  100H  to  11FFH
                          where   17   (decimal)
                          is 11(hex).
```

SAVE overlays any prior file of the same name, and will continue past a disk error.

FILES:

SAVE is a built-in command.

ALSO SEE:

OPIRES - SAVE will only work when OPI is resident before the save is requested.

MESSAGES:

```
??? - If OPI is non-resident,
    - or size is missing
    - or pages >256 (>64k)
    - or cannot create output file
    - or disk full during write
```

Bad disk block over-written - Allows SAVE to be used to cover a bad spot on the disk. Save gives an error message and continues.

NOTE:

The syntax for SAVE is different than the incorrect syntax used in CP/M systems. The M/OS-80 SAVE command conforms to the CP/M COMMAND line syntax of:

```
OPERATOR <filename> <parameter>
```

2.28. SORTDIR - CREATE SORTED DIRECTORY FILE

SYNTAX:

```
    SORTDIR <outfile>=<gname> [/M]

        Mostek M/OS-80 Disk Sorted Directory List (SORTDIR 01.01)
        DISKNAME:  enter  disk  1 name (Use :  (colon) on  drive
                                        letter ie. B:Somename)

        searching disk: d

        DISKNAME: enter disk 2 name    (Use :  (colon) on  drive
                                        letter ie. B:Somename)
        searching disk: d

        DISKNAME: (ret)
        Filenames will be sorted.
        nn file - done
```

DESCRIPTION:

SORTDIR will search the directory of a number of disks (1 or more) and build a file of the filenames found, sorted in alphabetical order. Each name will be tagged by the name of the disk on which it was found. The /M option is used to search for specific files on multiple disks. If the user need to find all of the disks files with the .COM extent, the user would type "SORTDIR <outfile>=*.COM /M". Sortdir will ask the name of each disk name before each disk search. If the name given starts with a drive name, (i.e. A:) that drive is searched.

SORTDIR is used for two purposes:

1) Building documentation of a multi-disk set.
2) Used as the starting point for a @ (Batch) CMD file.

The output file format is:
(,)filename(spaces).(period) filetype (tab) (<) diskname (>)

For a one-disk list, remove the disk name by:
   1) Use a disk name of "-" (dash).
   2) Replace "(tab)<->" by null, using the editor.

To allow multi-disk sorts of different types:
If the first 2 characters of a disk name is: (letter) (colon), then that disk will be switched to next.

MESSAGES:

A sample of the program run is:

```
B.SORTDIR B:ZIP=*.DOC
Mostek M/OS-80 Disk Sorted Directory List (SORTDIR 01.01)
DISKNAME: M/OS-80 Manual
searching disk: b
DISKNAME: (ret)
filenames will be sorted
12 file - done

ZIP contains:

,OSCMD1  .DOC  :<M/OS-80 Manual>
,OSCMD2  .DOC  :<M/OS-80 Manual>
,OSCMD3  .DOC  :<M/OS-80 Manual>
,OSCMD4  .DOC  :<M/OS-80 Manual>
,OSCMD5  .DOC  :<M/OS-80 Manual>
,OSCMD6  .DOC  :<M/OS-80 Manual>
,OSCMD7  .DOC  :<M/OS-80 Manual>
,OSCMD8  .DOC  :<M/OS-80 Manual>
,OSHW1   .DOC  :<M/OS-80 Manual>
,OSINTRO .DOC  :<M/OS-80 Manual>
,OSOPER1 .DOC  :<M/OS-80 Manual>
,OSOPER2 .DOC  :<M/OS-80 Manual>
```

WARNINGS:

Since disks will probably be switched in a multi-disk sort, it is advisable to specify a specific output disk, i.e., A:OUT instead of just OUT, with A being the current disk. The file indicated by <outfile> must reside on the current disk.

2.29. SPLIT - FRAGMENT TEXT FILE

SYNTAX:

```
SPLIT filename  [nn]
```

Where nn is the maximum size of each segment in K bytes. If no size is given, 16K segments are created.

DESCRIPTION:

Splits the text into smaller files starting a new file wherever: a "(return)(line feed)#(return)(line feed)" is found or when the segment is bigger than nn kilobytes.

Each subfile is named the same as the original with a suffixed serial digit(s): 0, 1, 2...9, 10, 11, 12... FILE.ASM will generate FILE1.ASM, FILE2.ASM...

The original file is not disturbed.

The serial number of the segment is added to the end of the filename. The name must contain a maximum of 7 characters to allow up to 9 splits, and a maximum of 6 characters if there can be more than 9 splits.

Creates the new files with parity low, no rubouts or nulls, and all control characters other than return, line feed, and tab are printed as ^l (two characters in the output file).

FILES:

SPLIT.COM is the program.

WARNINGS:

The recognized control characters do not include escape (^[) or formfeed (^L).

If files that have been created from SPLIT exceed the given segment by one or more bytes (due to the text), the directory entry for that file will show to be 1K greater due to the allocation of an additional cluster.

2.30. SPOOL - CONTROL PRINT SPOOLING

SYNTAX:

```
     SPOOL            - Gives instructions on running SPOOL.
     SPOOL d:filename - Starts the specified file printing.
     SPOOL *          - Stops any spool print in progress.
```

DESCRIPTION:

The print spooler is a system feature that allows the printer to output a file to the system list device while the system continues with other functions.

The file must remain on the disk while it is being printed and must contain all its own page handling. Any ASCII file may be spool-printed, but direct printer activity cannot occur while a spool-print is active.

MESSAGES:

File not found - The file must either be on the A: disk or the disk must be specified.

WARNINGS:

SPOOLER takes only partial control of the printer. Other programs that use the printer may also send data to the printer driver resulting in a mix of characters printed. Control P is, however, de-activated.

2.31. STARTUP - EXECUTE INITIALIZATION PROGRAM

STARTUP.CMD - Auto-start batch file.

DESCRIPTION:

If the file "STARTUP.COM" exists on the boot disk, it will automatically be run as a program with a blank command line. If the file "STARTUP.CMD" exists on the boot disk, it will be automatically started as a batch stream using Batch (@) unless STARTUP.COM is present.

STAND-ALONE SYSTEMS

Since startup is intended for use in implementing stand-alone systems, the user control-C is totally disabled when either the startup program (STARTUP.COM) or command file (STARTUP.CMD) is started, thus locking the user into the running program. Note that the normal "stop batch" capability, by pressing return, is inactive and ^C is also inactive.

When return is made to the Operator Interface, control-C is re-enabled so normal login sequences can be effected.

ALSO SEE:

@ - BATCH

WARNINGS:

- For STARTUP.CMD, the "A" disk must be write-enabled.

- If STARTUP.COM is run, then STARTUP.CMD will not be.

2.32. SXFER - SINGLE-DISK FILE TRANSFER

SYNTAX:

```
SXFER  <gname> [/M]

Mostek single file transfer (SXFER 00.03)
Insert * SOURCE * disk, press return
```

SXFER will find all files that match the general name given. If /M option is requested, the operator is prompted to verify that each file of the general set is to be transferred or passed over.

```
Insert * OUTPUT * disk, press return
```

SXFER will report files output as they are written.

```
****** COPY COMPLETED *******
```

DESCRIPTION:

SXFER is a file-transfer utility especially designed for single-disk systems like the Mostek Matrix-80/SDT. This program is different from XFER or COPY since files are moved one by one, based on the filename. The source and output disks may be of different size or density.

The program is set up to use as much memory as is available and will load partial files to fill memory on each pass.

There are three stages to the program:

1) Find all files that match the general name given. If the MAYBE option (/M) is given, the operator will be asked which of the matched files are actually to be moved.

2) Load source disk and load into RAM memory as many files as will fit.

3)Load output disk and write to disk all that was loaded in the previous load pass.

Passes 2 and 3 will alternate until all specified files have been copied to the output disk.

FILES:

SXFER.COM is the program.

MESSAGES:

The SXFER program has a large number of messages which fall into a number of similiar categories:

- Status reports, indicating an action taking place.

- Normal error, indicating disk full or similiar condition.

- Absurd error, indicating a condition that shouldn't happen.

All messages use the following insertions:

ffff	- a file name
tttt	- a type of transfer (New, Partial, Full, xxxx)
nnnn	- a record or file count
eeee	- a system error code

Status reports:

There are nn files - An indication of total files to be transferred.

```
- ffff - tttt load of nnnn records
- ffff - tttt dump of nnnn records
```

Indicates that file ffff was loaded for input, either tttt is a partial or a full load and nnnn records were loaded.

Normal errors:

```
*** Input position error [file ffff at nnnn]
```

Could not position file for input, either a null file (if nnnn is zero) or an incomplete file (missing blocks): the former can be ignored, the latter cannot be copied properly.

```
** COULD NOT CREATE FILE - DIRECTORY FULL **
** Sorry - this is a fatal condition      **

** File write error (eeee) **
** Sorry - this is a fatal condition **
```

Indicates the output disk is full or had a write error, and no more files can be written.

** Could not open [ffff] - File from file list could not be found on input disk ???

** Could not open [ffff]

*** Output position error [file ffff at nnnn]
On output, means a partially copied file could not be found when the second part was to be written.

** Unable to close ffff (eee)
The file could not be found when it was time to close it.

WARNINGS:
It is advisable to be careful about switching disks since it is possible to put the wrong disk in at the wrong time and wipe out a good file.

SXFER performs no verification and no temporary file cycle, so if an old file on the output disk is to be kept it must be renamed prior to the SXFER copy operation.

2.33. TYPE - TRANSFER FILE TO CONSOLE

SYNTAX: TYPE <gname>

DESCRIPTION:

The first file matching the general name will be printed, in ASCII, at the console. Tabs will be expanded into spaces. During the display, RETURN will cancel the listing. Space will pause the listing until another space key is pressed.

Non-printable characters are expanded to ^l format, except for return, line feed, and tab. Returns without following line feeds have a line feed inserted.

FILES:

TYPE is a built-in command.

MESSAGES:

File not found - No matching file could be found.

2.34. WRTSYS - WRITE FILE TO SYSTEM AREA

```
SYNTAX:  WRTSYS[/S]
                    {<output disk reference>}:={<input disk >}:
                    {<filename>              } {<filename>   }
```

Either operand can specify a file or a disk reference. /S indicates a single disk system, and requests changing disks.

DESCRIPTION:

WRTSYS is a program used to write a file to an area of the disk designated as the "system area". Usually, this is tracks 0 and 1 on eight-inch floppy disks but the actual tracks allocated to the system area can vary from disk to disk. No other program is allowed to write to this area (with the exception of DSKDUMP). The system area is where the boot PROMs expect to find the system to be read, loaded and executed. In some cases, the program in the system area is another boot program which, in turn, reads the real system from a file called SYSTEM.COM which is in the files area of the disk.

WRTSYS can move a boot file (which is a program designed to boot the DOS) to the disk system area, or can create a file-image of the system area or copy the system area of one disk to another disk's system area.

In any case, WRTSYS logs in both disks and properly determines the size of the system region. It then reads or writes as much of this region as necessary. All operations are read-after-write verified.

Examples:

WRTSYS A:=B:	Read system area on disk B: and write it to system area of A:
WRTSYS ZXOS16.SYS=A:	Copy System Boot from disk A to file ZXOS16.SYS.
WRTSYS A:=OS16.SYS	Copy System Boot from file OS16.SYS to system area of disk A (when setting up a new boot disk).

WRTSYS requires '.SYS' as the file type when a filename is given, i.e., WRTSYS B:=NAME.SYS.

WRTSYS allows writing from a file on one disk to the system area of another on the same drive, by specifying '/S' on the command line (the standard procedure for MOSTEK MATRIX-80/SDT systems). WRTSYS will pause after reading in the source file, and before writing, to allow changing disks.

FILES:

The file WRTSYS.COM is the program.

MESSAGES:

ILLEGAL DEVICE SPECIFICATION - Input or output request is invalid.

WRTSYS done and verified -Function completed normally.

Put new disk in "A" drive, press return
- Single disk WRTSYS disk swap.

Input/output errors:

Open error	- Input file not found.
Create error	- Could not create output file.
Read error	- Error on reading input source.
Write error	- Error on writing output.
Re-reading error	- Error on reading for verify.
Verify error	- Data written fails to verify.
Not Enough Memory to Verify	- Insufficient memory available to verify. WRTSYS opens a new buffer and completely reads in the tracks of the destination disk. If there is not enough memory to put both the source and the destination tracks in memory, this error will be produced. The copy is probably O.K.
Not Enough Memory For Disk Buffers.	- Insufficient memory available to fit tracks into memory for the copy. System too small to run program.

WARNINGS:

WRTSYS requires at least twice the size of the system region for verification plus room for program and current running M/OS-80 system.

2.35. XDIR - DISPLAY SORTED DIRECTORY

SYNTAX:

```
XDIR            - Display all non-system files.
XDIR <gname>    - Display all non-system files matched.
XDIR <gname> S  - Display all files matched.
```

Format:

```
<Filename>  <Kbytes in file>    (3 or 4 across)
```

Final message:

```
nnn entry(s) listed,  nnnK disk space used.
```

DESCRIPTION:

XDIR will display a sorted directory-file listing. It gives a multi-column display that accounts for the terminal display problem by pausing when the screen is filled.

XDIR will normally NOT show system-type files, but will do so if a second parameter of 'S' is present.

The program may be tailored for different-sized terminals by changing the lines and columns counters:

```
DSKDUMP XDIR.COM
block: 0 (ret)
block: M3,rr,cc(ret)
block: W
block: Q
```

Where: rr is the number of rows to be used by the XDIR display, usually screen length-2. The value is in hex. cc is the number of columns of names to display. Standard is: 16,4

FILES:

XDIR.COM is the program

WARNINGS:

XDIR properly collects incomplete file extents, but bases its size calculation on total records as if all extents were full.

It does not show file attributes, as there is no room on the screen.

2.36. XFER - TRANSFER FILES

SYNTAX:

```
     XFER  (return)
     or
     XFER [/<options>]  <disk>=<gname>
          - Move  all  matching  file(s)  from  one  disk  to
          another.
     XFER [/<options>]  <fname>=<fname>
          - Move a single file to a new file.
     XFER [/<options>]  <fname>=<fname>,<fname>...
          - Append multiple files to make a new file.
     XFER/V  B:=A:*.*
          - To copy a whole disk:   (from drive A to B)
```

Options:

```
/A - Do an ASCII file transfer.
/C - COMPARE without moving.
/E - EXIT on compare failure.
/F - FILTER out illegal ASCII characters(implies S&A).
/I - IGNORE ASCII eof (^Z).
/M - MAYBE transfer a file (if operator says yes).
/N - Copy only new file to disk.
/O - Copy only if files already exist on disk.
/P - Proceed on Error.
/S - STRIP all rubouts and nulls, and parity (implies A).
/T - Expand TABS (implies A).
/V - VERIFY after moving.
/X - Print HEX address of comparison failure.
/Z - Don't print transfer size statistics.
```

DESCRIPTION:

XFER is the general file-transfer utility. It allows moving files from disks or devices to other (or the same) disks or devices. Files are moved by first copying to a temporary file, then when completed and verified, removing any older file of the same name, and renaming the new file.

Files are assumed to be binary unless the "A" (ASCII) option is specified.

The MAYBE option (/M) allows a general filename to be individually selected by asking the operator if each file is to be copied. The answer may be either:

```
N - No, skip it.
Y - Yes, copy it.
R - Remaining. Copy it and all following files.
```

All options must precede file specifications, start with a slash (/), and end with a space.

When ASCII data is transferred, parity (bit 7-MSB) is automatically stripped and the end-of-file is indicated by ^Z (SUB). Stripping is important when ASCII files are appended so that the whole data block (128 bytes) is not transferred.

The file-compare option ("C") is a compare without copy. It is used to compare two files for identical contents.

If the "T" (TAB) option is specified, ASCII format is assumed and tabs are expanded. This option is used primarily to create printer listings.

The IGNORE and FILTER options may be used together to restore a destroyed text file. It will strip all non-printable characters except return, line feed, tab, form feed or alternate form feed (^K), and restore a proper end-of-file marker.

<u>Local mode:</u>

If no file is indicated, the program responds with an exclamation point (!) to prompt for multiple commands. Each command is processed as it is entered. A null line will indicate no more commands are to be entered. Any options should be placed at the beginning of the line on which they are required. Since XFER does a login before each command, disks may be changed between local commands.

Examples:

```
XFER/T  PRT:=XFER.Z80     Print file.

XFER/A   PGM.ASM=PART1.Z80,PART2.Z80   Append 2 source files

XFER    B:=A:DUMP.COM                Copy between disks

                    or

XFER    ret
!/TA    PRT:=XF
!     JUNK=RDR:
!  (return)
```

Disk-to-disk transfers and general filenames may be used.

```
                    New disk in B
                    Old disk in A
e.g., xfer/v b:=*.asm will move all assembler files

xfer/v b:=a:*.* will move ALL files
```

The disk specifier must be on the left side, and the general name can only be on the right side. General renames cannot be done while transferring. To do this, use the REN command after transferring. If transferring and verifying a general filename, any verify failure will stop XFER and wait for operator response.

Files may be appended together to make one larger file. Appending of files is indicated by putting more than one source filename separated by commas (file1, file2, ...).

When appending text or other ASCII files, it is ESSENTIAL that the "A" option is indicated so that the end-of-file is indicated by ^Z (SUB), and the whole data block (128 bytes) is not transferred. General filenames cannot be used while appending.

Broken extents are built when creating a file using "Random" record I/O. References to records beyond the last allocated cluster cause the operating system to create enough extents to cover the missing area. These extents are not filled in with allocated clusters until the records skipped over in those extents are referenced by reading or writing. This concept seemingly creates very large files on the disk which may have no more than a few actually allocated clusters thus very little real data.

The transfer program (XFER) <u>will</u> copy files with 'broken extents' correctly.

FILES:
XFER.COM is the program.

MESSAGES:

- "<filename> OPEN ERROR" indicates the source file cannot be found or the output disk is full, or is an unreported write-protected disk.

- "<filename> WRITE ERROR" or "<filename> CLOSE ERROR" indicates the disk is full.

- No ambiguous files matched - No files matched general files name.

- Illegal output filename - General name as an output name.

- Failed compare - compare or verify pass failed.

- Length compare error - one file is the prefix of the other, but they are of different lengths.

- File specification error - bad command line (no =, etc.).

- Cannot append to an ambiguous file.

- Cannot erase old file - Transfer was completed and verified but the old file is write-protected and cannot be removed.

WARNINGS:

Use of a temporary output file allows assurance that a file will not be removed until its replacement is verified; however, it also takes twice as much disk space.

If Verify is not requested when performing a copy, and the receiving disk fills up before the entire file is copied, then the copy will not be completed although <u>no</u> <u>error</u> <u>message</u> <u>will</u> <u>be</u> <u>shown.</u>

2.37. XFLP - COPY FILE FROM FLP80-DOS TO M/OS-80

SYNTAX:

```
XFLP <filename>
```

DESCRIPTION:

XFLP is designed to permit those applications and data files written on Mostek SINGLE SIDED FLP80-DOS diskettes to be copied over to M/OS-80 disks. XFLP expects the FLP80-DOS disk to be mounted in the "B" drive and the M/OS-80 disk to be mounted in drive "A". Only a single file of specific filename may be transferred at one time.

When attempting to copy files up from FLP80-DOS, make sure to get a directory listing of the FLP-80DOS disk prior to entering M/OS-80.

NOTE:

Only ASCII files or programs or binary files that are of a data nature may be successfully transferred. ".BIN" modules may be copied but their ability to execute under M/OS-80 is doubtful unless they are specifically written for use in a CP/M or M/OS-80 system. .BIN files are renamed to .COM when transferred. Higher level language programs in their ASCII form must be re-compiled under the M/OS-80 (CP/M) version of the compiler before they can work.

MESSAGES:

Syntax error in command line	Self Explanatory
FLP-80DOS file not found	Self Explanatory
No M/OS-80 directory space found	Self Explanatory
FLP-80DOS file access error	Error returned by disk controller.
FLP-80DOS directory access error	Error returned by disk controller.
Error in extending M/OS-80 file	Could not build additional directory spaces on CP/M disks.
End of M/OS-80 disk space	CP/M disk full.
FLP-80DOS Disk must be Single Sided	

2.38. XSTAT - REPORT DISK STATUS

SYNTAX:

```
     XSTAT
     or
     XSTAT <disk>:
```

DESCRIPTION:

This command prints:

The total, used and free space on disk.
The size of user RAM memory.
The total and free number of directory entries.
Disk and directory parameters.
Invalid conditions in the directory.

The whole report format is:

```
1) Disk label is /.........../ Version xx.xx
    or
   Disk is Unlabeled

2) Directory type:  Standard

3) User memory size:      nn Kbytes
4) Total disk space:     xxx Kbytes
5) Disk space used:      xxx Kbytes
6) Disk space remaining: xxx Kbytes
7) Cluster size:          xx
8) Sectors/track:         xx
9) Total Sectors:      xxxxx

No files on disk
    or
errors:    xxH Not allocated
           xxH Linked Cluster <orig-file>  <curr-file>
           <curr-file> Null File


10) Directory Entries Used:    xxx
11) Directory Entries Remaining: xxx
12) Disk Status Completed.
```

XSTAT prints the names of any NULL (empty) files.

The Total disk space may be used to determine the type of disk being used, which is useful on dual-(switchable) density disk systems. For example, a standard IBM 8" disk has normally 243 Kbytes.

XSTAT runs a validation on the disk directory to see if any cross-linked files have occurred. These can be caused by forgetting control-C (^C) when changing disks.

MESSAGES:

The errors that XSTAT reports refer to cluster numbers (## in hex) that are the internal segments of disk space that M/OS-80 allocates to each file.

Possible directory error messages are:

```
##H NOT ALLOCATED
```

Operator still has not pressed control-C, (so far no damage has occurred).

```
##H LINKED CLUSTER  <file1>  <file2>
```

The same disk cluster has been allocated to both files. The effect of this is that one file has been written over an older file. The more recent of the two is still unmolested - delete the older one. It is safest to delete both, since it may be impossible to tell which file is undamaged. ASCII files can be checked out by "TYPEing" or DUMPing them.

FILES:

XSTAT.COM is the program.

3. M/OS-80 INTERNAL OPERATIONS

M/OS-80 is an operating system for disk-based Z80 MOSTEK MDX or SD microcomputer systems. The function of an operating system is to define certain functional conventions and provide programmers with generalized input/output routines to be used in advanced general-purpose microcomputer systems development. These comprehensive sets of programs and routines allow a program to ignore details of device operation and merely specify a device symbolically. The system is supported at two levels: the resident operating system and the loadable extended file features.

The resident operating system consists of 3 major sections:

IOSYS	- Device I/O System.
DOS	- Disk Operating System.
OPI	- Operator Interface.

The operator interface (OPI) is described in detail in Section 2. The normal sequence of system operations uses OPI to invoke a user program. Once loaded, the user program passes all system requests to DOS. IOSYS is considered internal to DOS, but is retained for compatibility reasons.

M/OS-80 PROM-based system makes use of the MOSTEK Disk Controller Firmware (DCF) and Designer's Development Tool (DDT) PROMs. These PROMS are designed to boot the system and provide a minimum operator interface in case of system failure. Starting at E000 (hex) and extending to EFFF (hex), these two 2716 (2K) PROMs are constantly part of the system address space. It is extremely inadvisable to make direct reference to these PROMs in any software written for M/OS-80 because future releases of M/OS-80 may not use these PROMs.

3.1. SYSTEM ORGANIZATION

M/OS-80 is a RAM/Disk-resident operating system that is loaded from disk when a cold boot sequence is initiated. During system configuration, the memory location of M/OS-80 is fixed, and the bootstrap loader is automatically setup to load M/OS-80 properly.

The system resides in high memory above user programs. By defined use of low memory (0-100H) all user programs call M/OS-80 through a standard sequence which is transparent to actual memory size. Additional memory simply expands the available user area.

The system is broken down as follows:

RAM Memory areas:

Top ->	BIOS	- Device Driver System.
	DOS	- Disk Operating System.
	OPI	- Operator Interface Linkage.
100H->	TPA	- Transient Program Area.
0-FFH	Low Memory	- Reserved Space.

Allocation of Disk Areas:

Track 0-1	System Area
Tracks 2-n	Disk File Directory
Remaining:	User File Region

These system segments work together to handle all standard disk and character-device I/O for the user. The functions are described in detail later in this manual.

3.2. MEMORY STRUCTURE

3.2.1. IOSYS

The Input/Output System performs the various primitive I/O functions for the character-devices (console, printer, punch, reader) and for the disk devices. A program which does all of its own file management functions or does not use the disk might need only IOSYS.

3.2.2. DOS

The Disk Operating Subsystem performs all of the disk-oriented features of M/OS-80 including managing disk files (creating, opening, reading, and writing). In addition, M/OS-80 DOS is responsible for calling user programs, editing console input, allowing user control over I/O, and many other functions.

3.2.3. OPI

The Operator Interface is the set of routines that the operating system uses to communicate with the user. This sub-system permits the user to specify which of several system functions to perform. To execute one of these system functions, OPI invokes a user or system program based on the command line and passes parameters from the command onto the program. Those commands which invoke system-level RAM-resident programs are referred to as built-in commands. If a command is not recognized as being one of the built-in system functions, M/OS-80 tries to find and execute a disk-resident .COM module corresponding to the command. Any programs invoked are loaded into the user area as described in the "User Area" section below. OPI can be totally RAM-resident or may be reloaded between user programs, see the OPIRES and OPINRES commands. If OPI is non-resident, a small OPI linkage table is left resident in RAM.

3.2.4. Transient Program Area (TPA)

The RAM location where user programs are loaded and executed is referred to as the TPA (Transient Program Area) or user area. The TPA starts at 100(hex) and can extend as high as the bottom of the Operator Interface (OPI). If the OPI is non-RAM-resident, user programs can extend up to the bottom of the DOS (giving an additional 3K). All commands not built-into the DOS and user command programs are loaded and executed here. Built-in commands generally do not modify this area. **If OPI is non-resident, it will always use the TPA. Bootstrapping also uses the lower 3K of the user area.**

3.2.5. Low memory

From 0 to 0FF(hex), (the first 256 locations in memory) are reserved for special purposes by the system, although some selected locations are available to the user. (See Function Details.)

3.3. DISK STRUCTURE

The following sections discuss M/OS-80's disk structure. Shown below are a list of terms used and the definitions of those terms.

3.3.1. Disk Terms - Definitions

<u>Bootstrap Loader</u>
Program whose function is to load the M/OS-80 DOS from the system disk when the system is first powered up.

<u>M/OS-80 Resident Image</u>
M/OS-80 system itself. May reside on the system area or as a file (SYSTEM.COM). Loaded and executed by the Bootstrap Loader.

<u>Disk File Directory</u>
The section of the disk used by M/OS-80 to keep track of the files in the user area of the disk.

<u>File Block</u>
One 128-byte area of a disk. Usually one sector, although sectors can be larger than 128 bytes, in which case a sector contains more than one block.

<u>Cluster</u>
Eight or more (in multiples of eight) file blocks. Clusters are used to allocate space to files. Since a cluster consists of eight 128-byte blocks, a cluster is at least 1K long.

<u>User Files</u>
Portions of the disk user-file area are allocated by the system for user-created files. The user-file area extends below the directory area to the end of the disk. Files are allocated space in cluster-sized pieces. At least one cluster is allocated for each file (that has been written to) so the minimum file size is 1K bytes even though not all sectors have been written to. As each cluster is allocated to a file, the system marks that cluster as used in the "bit-map" and adds the cluster number to the file's directory. Each file has one directory entry for each 16K bytes with each extent holding 16 entries.

Bit Map

M/OS-80 keeps a map of all clusters of each disk logged in. As clusters are used (assigned to file directories) M/OS-80 marks that cluster as used in the bit map. The bit map is created whenever a disk is logged in (upon first use or after a control-C) by reading the entire directory and posting the allocated clusters from the directory entries for all un-deleted files. The system does not read the directory when control-c is hit but waits until the next requested access to the unlogged disk is made.

Disk map

Block		Cluster
0	Bootstrap	--
1..51	M/OS-80 Memory Image	--
52..67	Directory (2K=64 entries)	0..1
68...	Files	2..240

Directory blocks number 0..63 (in a standard 64-entry directory).

Fig. 3-1: Standard Disk Allocation Map

3.4. BOOT STRUCTURE

M/OS-80 has an extended boot structure when compared to CP/M. The firmware cold starts from the system region of the boot disk, by having the boot PROM load block 0. This block contains a bootstrap program that loads the system proper, from the system area of the disk.

First, the system initalizes all basic devices, especially the boot disk. At this point, the system is actually running; however, no user program is active. The newly booted software then chains to a program called "SYSTEM.COM" which contains the system to be actually run. This program module is brought in from disk, initializes its special devices, and then loads the Operator Interface from "OPI.COM". OPI is self-relocating and will move itself into high memory if it has been specified as RAM-resident at boot time.

The system region must contain a proper system image, SYSTEM.COM and OPI.COM must exist on the "A" disk, and these modules must match at revision level for the system to boot successfully.

Boot functionality can be changed by users in the field. Refer to the MOSGEN Operation Manual for changes in boot structure.

3.5. COMMAND STRUCTURE

A M/OS-80 command is a program that is designed to be loaded starting at 100H and stored on the disk as a memory image in a file with type .COM. Program (.COM) files may be either a MOSTEK-supplied system utility or a user program. Any filename is legal as a command; however, built-in names will not be recognized as user programs because the operator interface predefines them. Examples are :ATRIB, BYE, DIR, ERA, REN, SAVE, TYPE, etc.

The operator interface will pass the user command line to a command program. The invoked program need not use the command line or M/OS-80 itself, although most will use both.

For ease of programming, the first two arguments in the command line will be pre-formatted into standard file control blocks (FCBs) at 5CH and 6CH by the console processor. The rest of the command line, after the command name, will be placed at 80H.

Standard format for the command line is:

```
                          (Arguments)
      A.<Program name> [/][<operand>[/]<operand>]
     /      ^             \
    /       |              \
 M/OS-80   .COM filename   User (system) program
 prompt                    parameters
```

For example:

To use the XFER utility to copy all files from drive A to drive B:

```
      A.XFER /V B:=a:*.*
```

The /V is seen as the operand "V" and the B:=A:*.* is interpreted as another operand. The user program itself decodes that portion of the command line.

If a name which not does match a .COM file on the current or library disk is used as the program name, a "Program Not Found" message is printed.

3.6. PROGRAMMER FACILITIES

3.6.1 System Calling Conventions

M/OS-80 reserves low memory (0-0FFH) for internal uses. M/OS-80 resides in high memory with OPI and BIOS. The user is allocated all memory from 100H to below M/OS-80 for unspecified uses.

A program is always loaded and starts execution at 100H. Once started, it can do whatever it wants; however, if it destroys M/OS-80 or BIOS it will have no way to use the disk or recover the operating system without the operator intervening.

M/OS-80 is entered through two special locations in low memory:

JMP 0 is used for returning to OPI and the operator or the next step in a batch.

CALL 5 is the normal system request call.

Standard conventions, when M/OS-80 is called thru CALL 5, are discussed in section 4. (**See Function Calls.**)

The region from 0-100H has the following data of specific interest to a program:

0..2	System return entry.
3	IOBYTE for device re-assignment.
5..7	System call entry.
6..7	Pointer to top of user area.
08H..4FH	Reserved for interrupt vectors.
30H-32H	Reserved for system.
38H-3AH	Illegal address trap.
40H-5BH	Reserved for system.
5CH..6BH	User FCB 1.
6CH..7BH	User FCB 2.
80H..0FFH	User buffer.

3.6.2. User Memory Size and Stack Position

The two system entries (locations 0 and 5) are also used to supply the user program with special information. The system return entry is also the start of the BIOS entry vector and the system call entry is the first location in M/OS-80 which is one byte above the highest available user memory.

The following section of code will supply the user with the system's lower boundary (the user's upper boundary).

```
        LD   HL,(6)       ;GET CONTENTS OF LOCATION 6
        DEC  HL           ;HL:=TOP OF USER MEMORY
```

3.6.3. Command Line at Program Start

The user file control blocks (FCBs) UFCB1 and UFCB2 and user buffer (location 080H) are reserved for the user. They may, however, contain specific information from the operating system when a user program is started. These FCBs and buffers are used to pass the command line that OPI was given to start the program.

When a program is invoked, pre-processing is done on the command line by OPI, and the results are placed in special locations for the program.

1) The whole command line, translated to upper case, is placed in the user buffer starting at 81H and terminated with a null (0) byte. The buffer starts with the character after the command file name, which is either a space or a name terminator (/,=,etc.) The length is placed in 80H.

2) The first two filenames are pre-scanned and placed in user FCB 1 & 2 (at 5CH and 6CH respectively), ready for use to access disk files as per FCB conventions (described later).

   Note that user UFCB 2 is not a full FCB and will be destroyed if user UFCB 1 is used. Similarly, any disk I/O will wipe out the command line if the disk buffer is not saved elsewhere.

3) The disk I/O buffer is also pre-set to 80H.

3.7. CHARACTER DEVICES

M/OS-80 has a set of system calls for direct interaction with physical character devices. Those supported in the current version with links to actual hardware are noted with an asterisk (*).

	(Function)		(Call numbers)
*	Console	Input	1,6,10,11,80H
*	Console	Output	2,6,9
*	Terminal	Control	82H,8EH,9DH
	Reader	Input	3
	Punch	Output	4
*	List	Output	5
	Clock	Read and Set	8FH,90H,91H,92H

These are all considered ASCII character devices which return 7-bit data, high bit clear and use control-Z (1AH) to indicate end-of-file.

3.7.1. Character I/O Format

When output goes to Console, horizontal tabs (HT,9) are expanded with an eight-character-wide tab field (i.e., tab stops are 1, 9, 17, 25, 33...). Tabs sent to punch or list are left as the TAB (9) character.

Whenever character device output is occurring, the console input is periodically tested for ready status. Normally no action is taken other than loading any character on the keyboard into a single character buffer. There are a couple of special characters that can affect functioning:

> CONTROL-S (^ S) will pause any output until a continue is indicated by pressing any key. If Control-C (^C) is pressed, an abort will occur.

There are 3 controls for console/printer "daisy chaining" which cause all console output to be printed on the printer as well as the console:

> CONTROL-P (^P), when input, will toggle state of console/printer link but never be passed to a program.
>
> CONTROL-T (^T/ DC4/ 20/ 14H), on output, will turn on the link if transmitted from the console.
>
> CONTROL-W (^W/ ETB/ 23/ 17H), on output, will turn off the link if transmitted from the console.

For example, even though characters are being output one at a time, "AE(TAB)(^T)BC(^W)D" will produce "AE   BCD" on console and "BC" on printer. These functions may be disabled by using the console options function (9DH). (See Section 3.7.5)

3.7.2. Console Line Buffered Input/Output

For the console, line-buffered I/O is available by way of the M/OS-80 system call:

Function	Code (Decimal)/(Hex)
Output	9/09H
Input	10/0AH

Output buffering displays a sequence of text characters until a "$" terminator is found.

Input-buffered I/O fills a text buffer with characters from the keyboard until the buffer is full or a return is entered and allows character backspacing and other control characters:

CONTROL-E Physical new line only, not terminate input.

RETURN (Ctrl-M) Terminate input.

CONTROL-R Repeat what has been typed (minus any corrections).

CONTROL-U Delete current line.

CONTROL-X Delete with ECHO (for hard-copy terminals).

BACKSPACE (Ctrl-H), UNDERSCORE, RUBout or DELete - Delete previous character and back up cursor (for CRT terminals).

CONTROL-C Abort back to system.

CONTROL-P Toggle console/printer link.

All non-defined control characters will echo (^L) where "L" is the equivalent control letter.

To use buffered input, a buffer is passed with maximum size previously initialized. The following diagram illustrates the buffer:

```
-----------------------------------------
|Max chars/Actual chars/  ... Buffer / |
-----------------------------------------
      0        1             2.....n
```

The actual length is filled in by M/OS-80. A null (0) byte will be placed after the last byte (but not added in the count). If too many characters are typed, M/OS-80 will return with maximum = actual, but with no null byte at the end. A terminating return will not be in the buffer but it will be echoed to the console; however, no line feed will be sent.

3.7.3. Non-Echo Console Input

Many advanced programs require the ability to selectively decide what is echoed back to the terminal. There are two functions and a console option that provide for this:

(Action)	(Function)
Non-echoed input	80H
Raw input/output	6
Disable echoing	9DH(bit 4 & 6)

The non-echoed input is function 80H. It is the primary function for this requirement. This program option may be used where buffered input is being used, but echoing is disabled for password entry or other similiar functions.

3.7.4. Unformatted I/O

Raw or unformatted I/O is performed through function call 6. This call is used for both input and output based on the value of user register E. Again the character is un-echoed but with this function the OS performs no buffering or decoding of the characters as they are read thus permitting various control characters such as control-C or control-S to pass without alteration or other system action.

3.7.5. Console Options

The console-options control is to allow extended use of the standard console input functions. By changing the options, the operations performed by the system at end-of-line and while echoing can be tailored to a program's needs.

Changing the console options is done in two stages. A mask is passed to determine what values will be changed, and the new values of just those bits are passed.

Register (E) has a mask of the bits to be changed (bit=1 if the option is to be changed). Register (D) has the new values of options (bit=1 if the option is to be set, bit=0 if the option is to be reset).

Bit number	Bit value	Meaning
7	80H	Disable print output toggle (^W/ ^T).
6	40H	No echo on input (func. 1 & 10).
5	20H	Reserved
4	10H	Reserved
3	8	No echo of RETURN (func 1 & 10).
2	4	Enable ESCAPE as end of line.
1	2	Disk read after write (driver-optional).
0	1	Control-P flag proper.

3.7.6. Abort (Control-C) Control

The Control-C character is usually used to abort or abnormally exit a program and return to OPI. It is not always desirable to allow the user to exit a program improperly. Three states of Control-C processing may be specified, using function 82H:

(DE value)	(Action)
-1(0FFFFH)	Ignore control-C.
Address	Go to user exit on control-C.
0	Normal system action (abort).

Disabling control-C will cause it to be totally ignored. An address will become a routine which is jumped to on a control-C. The user must supply the return to the program through a jump instruction.

User-defined abort addresses are reset upon returning to OPI or when chaining; HOWEVER, when disabled, it is not re-enabled until done so by a user program. Disabling ^C also keeps Batch from being aborted by (CR) return. ^C is enabled only after hitting system reset or redefining the appropriate bits as shown above.

3.7.7 Printer Spooling

Printer spooling is a system feature to allow printing while other functions are continuing. M/OS-80 then prints from the specified file until the contents are all printed.

Starting a spool consists of taking a disk file to be printed, opening it like any other input file, calling function 85H, and passing the opening file control block. See **Disk Structure** for details on disk files.

(DE) Contains:	Action:
FCB address	Start printing file described
0	Stop any prior spool being printed
1	Get status of spool
	A= 0, not active; -1, active

3.7.8. Clock Functions

The system allows a clock feature, providing the clock hardware is available. The date function may be used regardless of hardware provided the GTOD function is used to set the correct time and date. The date function is Day, Month, Year (from 1900). The time function is seconds, minutes and 24-hour hours. All of the values are stored in BCD packed form, e.g., 12 minutes will show as 12H, not 0CH.

(Function)	(Action)	Registers
8FH	Set date	B=Day, D=Mon, E=Year-1900
90H	Read date	A=Day, B=Mon, C=Year-1900
91H	Set time	B=Sec, D=Min, E=Hour (24-hr time)
92H	Read time	A=Sec, B=Min, C=Hour

3.7.9. Terminal Control and Cursor Setting

M/OS-80 allows for standard control of a display terminal by including a terminal driver in the I/O system and having a system call (8EH) to set the cursor or allow for the functions. The functions listed may not be available on all terminals. Included in the effort to allow standard use of display terminals wherever possible, all M/OS-80 terminal drivers respond to Form Feed (control-L) to clear the screen. Since not all terminals do so, special conversions are in the drivers. The current version of M/OS-80 is sysgened with a 'dummy' terminal driver.

For those terminals that have cursor (arrow) key pads, a feature is provided that, when enabled, converts the key pad to the standard cursor-motion control keys. To enable the cursor pad M/OS-80 requires a program to call function 8EH, with option (0,13) on entry, and option (0,14) on exit. The cursor pad will remain enabled until disabled.

```
    Set CRT cursor:  D-column (1..80) , E-row (1..24)

    Special functions:
                 if E=0         then D denotes special function:
                         (For switch functions, on-odd, off-even)

    0-Clear              10-Normal lite          20-Page send
    1-Home               11-Keyboard on          21-Aux send
    2-Backspace          12-Keyboard off         22-Del char
    3-Right              13-Cursor pad on        23-Insert char
    4-Up                 14-Cursor pad off       24-Del line
    5-Down               15-Protect on           25-Insert line
    6-Clr eol            16-Protect off
    7-Clr eos            17-Blink on
    8-Highlight          18-Blink off
    9-Low light          19-Line send

    Cursor pad translates:
                              Up-^W

                  Left-^A                  Right-^D
                    or ^H
                              Down-^Z
                               or  ^J
```

This feature means that a program expecting a LEFT cursor function should have turned on the cursor pad and be looking for a control-A; similarly, up would be control-W. The alternate keys for left and down are respectively back-space and line feed.

3.7.10. Math Functions (Multiply/ Divide)

16-bit unsigned multiply and divide (not 2's complement) functions are provided for programmer convenience.

(Function)	Action
89H	DE := DE*HL

8AH	HL := HL/DE
	DE := HL mod DE (remainder)

3.7.11. Device Lockup

All abort (^C) and printer (^P, ^T, ^W) functions are external to the actual device driver routines so that *if* a device is *not* ready, the system will lock up; e.g., pressing (^P) with printer off will stop everything until the printer is readied.

If the device in question is set up with a ready status function, then the toggle can be reversed using another control-P (^P) or the sequence control-S, control-C (^S^C) can be used to exit the program and return back to OPI.

The sequence Control-S, Control-E (^S^E) can be used to skip the current input/output operation as if the device returned or accepted the character.

If the device has no ready status function, then the system will just hang until the device finishes its operation. For example, in this case ^S will have no effect if the paper tape reader is out of tape and the driver does not do a time-out.

3.7.12. IOBYTE

In order to add additional character-devices to the system, an "IOBYTE" patterned after INTEL's^{tm} is provided, but not normally implemented in the I/O system. It is normally in location 3, but should be accessed through the provided system calls. A single byte is used to specify which physical device is to connect to which logical character device:

```
BIT | 7  | 6  | 5  | 4  | 3  | 2  | 1  | 0  |
    (________) (__) (_______) (___________)
         |       |      |           |
    PRINTER   PUNCH  READER      CONSOLE
```

(NOTE: The INTEL^{tm} form only provides four consoles, but since M/OS-80 is oriented toward an eight-user system, it provides eight.)

INTEL is a trademark of INTEL Corporation.

3.7.13. M/OS-80 Character I/O Implementations

The current release of M/OS-80 provides the user with the ability to choose a particular set of control codes for his terminals special functions. (Refer to the MOSGEN Operations Manual for detailed description on linking these control codes into the operating system.) The system console is connected to an asyncronous serial port of the users choice. The currently configured system (before any user modification) uses either the MDX-SIO1, MDX-SIO2, or the MDX-EPROM/UART serial boards.

The system printer is now connected through the MDX-PIO. The option of changing the list device is totally up to the user. The standard Centronics interface is supported.

M/OS-80 has provisions for both a reader and a punch device; however, there are no firmware or software connections to these in the present release. There is also no current support for a hardware clock, although calls to M/OS-80 will support the time-of-day and date functions. Time and date are set by using the GTOD function and remain constant until reset. The user does, however, have the ability to generate a system with these drivers. (Refer for the MOSGEN Operations Manual for detailed descriptions on linking these drivers to the operating system.)

Virtually all system functions in M/OS-80 will not over-run any character I/O device running at 19.2 K Baud. If over-run problems are encountered with a particular terminal, it is recommended that the DTR/CTS 'modem' protocol be used for the console terminal. If the console terminal does not support this protocol, the baud rate of the terminal must be reduced or suitable waits inserted in the offending software. Operations running in DDT may over-run a terminal at 19.2 to 4800 baud during the memory display. **Neither M/OS-80 nor DDT supports X-ON/X-OFF protocol.** Remember that X-OFF is the character Control-S (^S) and that the operating system treats the automatic ^S as if it came from the keyboard.

Note that all MOSTEK serial devices send ASCII data in eight-bit form and ignore parity. Since some software routines set parity as a software flag, these non-standard characters could pass through to the serial device and be transmitted as valid eight-bit values. Make sure that the terminal that is used is set to IGNORE parity also to avoid problems. The ADDS series of CRTs display characters with incorrect parity as an asterisk (*).

3.8. DISK FILES

M/OS-80's power is derived from its management of the system's disk resources. The disk is divided into regions called "FILES". The user supplies a filename to the system which, in turn, builds a file control block (FCB) that the user then refers to whenever the file needs to be accessed.

3.8.1. File Structure

A file consists of a sequence of 128-byte blocks, uniquely described by a filename. The disk is managed in clusters of 1024 (1K) or more bytes each (eight or more blocks). As a file is written, the system allocates clusters for the file's data and stores pointers to the clusters being used in the file's directory entry on disk. When a file is read or written sequentially, the system automatically handles all functions related to the internal structure of the directory.

3.8.2. Extents

In a standard floppy directory, each file has one directory entry for each 16K bytes or 128 blocks. A file may have from 1 to 16 directory entries. Hard disk systems permit an extended directory structure which supports "sub-directories" and larger files. Each directory entry is called an "extent". A directory extent consists of 32 bytes in which are stored the file's attributes (protected, system etc.), the filename, the amount of space used, and which clusters of the disk are allocated to this file. For dataset creation, deletion, renaming or sequential access, the extents are treated as a cohesive whole but for random access, each extent acts as an independent file. and therefore the proper file extent must be specified by OPENs and CLOSEs so the appropriate directory entry is accessed. If the file structure is unlike that of M/OS-80, random block functions may be used (as when reading or writing to FLP-80DOS diskettes).

3.8.3. File Contents

The data in a file may be ASCII coded or eight-bit binary. A hard end-of-file exists in the last block of a file. If the file is ASCII text, the file does not have to be an even number of blocks. To handle uneven blocks, a system convention of ASCII character (SUB) (Control-Z (1AH)) is used to denote the logical end of file. If the file is exactly an even number of blocks, then a hard end-of-file will occur and no 1AH bytes are inserted into trailing blocks.

3.8.4. File Access

Steps:

```
--------------------------
1) Determine Filename
2) Setup File Control Block (FCB) for File
3) Open an Old File
        or
   Create a new file
4) Read and/or Write From the File
5) Close the File.
```

To access a file in M/OS-80 a system File Control Block must be created. The File Control Block (FCB), connects the logical description with the hardware or physical description. The FCB describes the name and disk drive to be used for the file. The CREATE (call 22) function is used to create the FCB for a new file.

The next step in the use of a M/OS-80 file is to OPEN (call 15) the file specified by the FCB just created. In the process of opening a file, the system fills in the FCB with an almost identical disk file directory entry for that file. In addition, the directory entry clusters are allocated for the file and other information pertaining to the size of the file.

Once a file has been opened, various calls can be used to write to various areas of the file. Most I/O functions to and from the disk require the use of an FCB to direct the file access process and to restrict that access to areas pertaining to that one file.

3.8.5. Filenames

```
A "FILENAME" specifies:
       1)  a disk unit,
       2)  a file name of 1 to 8 characters
and    3)  a type of 0 to 3 characters.
```

Any alphanumeric characters may be used except: star (*), question mark (?), slash (/), period (.), comma (,), colon (:), or space. All system functions convert lower-case characters to upper case, but the system allows any characters once the file control block has been created.

3.8.6. Building a File Control Block

The user can specify a file in a command line, in a program by a character string and convert it to an FCB, or directly format the name as a File Control Block (FCB). For ease of programming, the first two file names given in a command line are pre-formatted into standard user FCB's by the console processor (at 5CH & 6CH).

A system call (function 86H) is provided to build an FCB from a name-string terminated by a slash (/), equal sign (=), comma (,) or any non-printing character. Lower-case letters are translated to upper case and a "*" is replaced by "?"s to fill out the name for general file searches.

The final result is a FCB initialized as below:

0	disk specifier (0-current, 1-A, 2-B...)
1..8	filename (left justified)
9..11	filetype (left justified)
12..14	zeroed
32	zeroed

Sequential FCB: Length 33 bytes
Random FCB: Length 36 bytes

FCBDK	Disk descriptor	0	(0-current, 1-A,2-B,3-C,4-D)
FCBFN	File name	1..8	(left justified)
FCBFT	File type	9..11	(left justified)
FCBEX	File extent	12	(Initially 0 incremented by 1 for each 16 Kbytes of file)
	Reserved	13..14	
FCBRC	Record count	15	(Total number of 128-byte blocks)
FCBMP	Cluster Allocation Map	16..31	up to 16 Allocated Clusters (-1,1..254)
------ the preceding 32 bytes are saved in directory-----			
FCBNR	Next record	32	(Next record to read or write) (0..127).
FCBRR	Random record	33..35	(3 byte-record pointer)

Fig. 3-2: File Control Block Layout

3.8.7. FCB Use

All M/OS-80 file calls denote the file by passing the FCB address (in DE) to M/OS-80.

3.8.8. Disk Buffer

All disk operations are handled by using a common 128-byte disk buffer which is set initially at 80H, but may be moved using function call 26. The directory search functions (17/18) use the disk buffer as do all read/write operations.

3.8.9. Disk Selection

The "CURRENT" disk function allows a program physical device independence. Whenever a file is OPENed or CREATEd, the first byte of the FCB indicates either a specific disk (A=1, B=2...) or the current disk (0).

The "A" disk is the initial state of the M/OS-80 file manager and results after a M/OS-80 reset (call 13H). Other drives may be selected (call 14H), deselected (call 12H) or ejected (call 8CH). Since SA800 disk drives do not have the eject feature, M/OS-80 gives an operator message and waits for appropriate operator action.

The current status of the disk system may be queried. The current disk is returned by M/OS-80 call 25. The disks that are logged-in is returned by M/OS-80 call 24, with one bit set for each active drive. Least significant bit is drive "A".

If a disk is labeled, the current disk label may be read using M/OS-80 call 95H. Register DE is the address of an FCB which will be filled in with the label of the disk specified by the disk byte of the FCB. This label name was the most-recently-filled-in name. The disk should be tested to see if it is logged in to ensure proper return of label name.

3.8.10. Opening and Creating Files

Before any processing can be done on a file, and after the FCB has been created, it is necessary to OPEN or CREATE the file. If a file exists previously, then it may be opened; otherwise, it must be created. If an old file exists, but its contents are to be totally replaced with new contents, it is effectively a new file. For this new file situation, it is advisable to delete any prior file, and then create the file as if it never existed.

Another method used by several M/OS-80 utilities is to delete any file with the same name but with a .BAK extension, rename any file with the same name and extension to name.BAK and create a new file. This naming process provides a level of backup that will add additional system security.

3.8.11. Data Transfers

Each time the file is accessed, the system will read from or write to the next record in the file, switching to the next extent as required. If the file has been modified, then the previous extent will be closed, and the next one will be created if it does not exist. During normal sequential processing, file extents are transparent to the user.

Data Transfer Function Calls are:

(function)	(action)
20	Read sequentially
21	Write sequentially
98H	Read without advance
99H	Write without advance
33	Random read*
34	Random write*
35	Determine file size*
36	Determine random record*

* Require random FCB.

3.8.12. Sequential Disk I/O

The simple read/write sequential functions are used for most processing. The disk buffer is set to point to the data buffer in memory, and the function is called. The read/write without-advance provide a simple means for updating data files by not changing the next-record value, thus allowing reading and re-writing data in a random fashion. Random I/O of this type requires the program to do its own extent processing (see below).

Random Disk I/O

Random access to files is accomplished by providing a random FCB, where the last 3 bytes of the FCB are a 24-bit block number (see FCB block layout: bytes 33-35) which indexes into the file. For most purposes, only the first 2 bytes need be maintained, as that covers 65,536 records, or eight megabytes, the current system maximum. This random-record pointer may either be totally maintained by the program and incremented for simple sequential processing, or sequential and random processing may be interleaved. Interleaved sequential/random processing can be done by using the Determine Random Record function (36) to convert the sequential FCB's extent and next-record number into a proper random-record pointer.

Determine File Size Function

Appending to a file may be done by using Determine File Size function (35) to set the random-record pointer and FCB to point to the next record after end-of-file.

Extent Processing

For program-directed extent processing, the program must handle switching extents by closing the current extent, changing the extent number, opening or creating a next extent and adjusting the next record number.

During sequential processing, M/OS-80 OPENs the next extent before a read or after a write. The effect is that the data buffer may not still contain the data after a write, since it may have been used for a directory access when the next extent was created or OPENed. This is only a problem for direct/random access or if a write buffer is used after an extent change.

```
  M/OS-80 system call numbers are shown by <##> in hex

(Name string)                 (Command line)
|                              |
|                              |
V                              V
[FCB format ]                  [Operator Interface]
|   <86>                       |
|                              |
------> [ FCB ]  <--------------
           |
           |
--------------------------------------------------<--------
|           |  |                |             |           |
V           |  V                V             V           |
[Find File] |  [Delete File]--->[Create File] [Open File] |
|<17>       |  <19>                  \<22>       /  <15>  ↑
|           |                         \         /         |
V <------   V                          V       V          |
[Find]  |  [Rename File]                ---->0<-----------|
[Next]  |        <23>                       / \       [Switch]
[File]  |                                  /   \      [Extent]
|<18>   ↑                                /     \         ↑
V       |                          [Write]    [Read]      |
-------/                           [Block]    [Block]     |
                                      |<21>    /<20>      |
                                      |       /           |
                                      V <----/            |
                                   [Close File]-----------/
                                      <16>
```

Fig. 3-3: Sequence of DOS file Operations

3.8.13. Directory Processing

The directory may be directly accessed in two ways:
(1) SEARCH/SEARCH NEXT functions
(2) As the file "SYS.DIR"

Searching the directory consists of building an FCB containing a name and extent value (bytes 0..12). The extent value may be a question mark (?) if any extent is desired. First, the SEARCH call is used (M/OS-80 call 17) to initialize the search sequence, and to return the first matching file. Use the SEARCH NEXT call (M/OS-80 call 18) to return subsequent matching files. Each subsequent call will return either a found file (A register = 0..3) or a file-not-found indication (A=255,-1).

The file found is in the directory block contained in the data buffer. Each directory block can hold 4 entries, so the specific file entry is both pointed to by "HL" within the buffer and indicated by the file number in "A".

While a standard directory has 64 entries, a larger directory may have been specified, so if a SEARCH/SEARCH NEXT scheme is used, provisions must be made for more than 64 entries. Up to 255 entries may occur (0..254).

3.8.14. Directory as a File

Another way of handling the directory is to OPEN the file "SYS.DIR". While the directory is not actually a file, using SYS.DIR as an FCB name will cause the system to create an FCB that looks like the directory. SYS.DIR is a write-protected "file", and cannot be renamed or deleted.

3.8.15. File Numbers

Each time the directory is accessed by a program, either to OPEN, CLOSE, or CREATE a file, or on a file search, the system returns to the user the file number of the file in the disk directory. This number is the number of the file from the start of the directory block, and is 0..3.

To determine where the entry is in a buffer containing the directory block, use:

```
     entry address:=(buffer address+16*(file number mod 4))
```
If buffer is 80H, and "A" has the file number 1 then:

```
          80H+16*(1 mod 4) -> 80H+16*1 = 90H
```

Therefore 090H is the address of directory entry one in the buffer.

3.8.16. Program Chaining

This function allows one program to call another program (M/OS-80 call 88H). OPI uses PROGRAM CHAIN to start a user program.

It is the responsibility of both programs to cooperate on passing data. The user FCBs and buffer are ideal places for storing inter-program common data. The data buffer and abort exit are reset on chaining, just like normal program startup. The default system buffer location is 080H.

The user program abort address is reset when chaining unless disabled, in which case it will remain disabled after the chained program receives control. When chaining, a return code may be passed to the next program. This return code is defined by M/OS-80 call 93H, and will be 0 if not set by the previous program. The return code is in the "A" register upon entry to the chained program.

3.8.17. File Attributes

Files may have attributes set to protect the file from normal access. Attribute functions include:

```
  3 active attributes:
W-write protect,
P-permament,
R-read protect

  2 informational attributes:
U-user
S-system
```

M/OS-80 call 94H is used to set file attributes:

```
DE=fcb with general filename to match
B=new attributes
   bit order is: [PWRU/S00+]

Bit 7 = P - Permanent
Bit 6 = W - Write-protect
Bit 5 = R - Read-protect
Bit 4 = U - User number (not implemented)
Bit 3 = S - System file
Bit 2 = Reserved
Bit 1 = Reserved - Big disk
Bit 0 = + - Add flag.


if "+" is set then new attributes are OR'd with
old attributes.
```

Files that have W or R attributes set use the high 2 bits of the first byte of the FCB to save that information (bit 7=W, bit 6=R). These bits are set by the file OPEN and are tested by the READ/WRITE functions . The lower bits of the first byte are not changed because they specify the disk.

3.8.18. File/Directory Management

Files may be deleted (M/OS-80 call 19) or renamed (M/OS-80 call 23). No permanent files will be affected. Either function allows a general filename, although no rename will occur if the new name matches any current filename on the disk (as of the beginning of the RENAME).

The RENAME FCB is a special form: the first 16 bytes are the old filename and the second 16 bytes are the new name.

These functions return (in the "A" register) the file number of the last file renamed or deleted , or a -1 if no files were affected.

3.8.19. Direct Disk Functions

For special purposes, it is desirable to directly access the disk without M/OS-80 file management. There are three functions for this purpose:

Function	Action	Registers
8BH	Home disk	B=disk (0-current,1-A...)
83H	Read block	B=disk
84H	Write block	ADE=block

The disk is specified as the current disk (B=0) or as a specific disk (B=1...8). When reading or writing, straight logical blocks may be specified or standard interleaving may be used, by setting the most significant bit of the B-register. The logical block is specified as starting at 0, and placed in the DE register pair. A 24-bit number is allowed; however, for floppy disks, the "A" register may be zeroed.

3.8.20. Changing the Bottom of the System

The user is allowed to load special function modules and make them part of the system by declaring a new system bottom with function 97H. By relocating the bottom of the system, M/OS-80 allows the system to tell each program where the top of its memory is, and will protect the special module until it is unloaded.

The new bottom is set by passing the high byte of the address in register "E". If the address is higher then the original system bottom (at boot time), then the original value will be reset.

```
        (E) has high byte of new bottom
                 e.g.,
                         To set bottom to 5100H then pass 51H
                         To reset bottom, then pass OFFH
```

The first nine bytes after system bottom ie: 5100H (5100-5108H) are reserved for M/OS-80. The user must not use this area.

3.9. USER CONTROL BLOCK (UCB)

M/OS-80 handles all disk and console requests by retaining some information in a User Control Block (UCB) and determining the remaining information at access time. A system call (M/OS-80 call 81H) returns the address of the UCB pointer. If the pointer is re-directed to a different UCB when re-scheduling is required, M/OS-80 will resume processing another user. However, the user program must specify to the system that M/OS-80 is to retain the user's stack inside M/OS-80 and not switch to its own stack, as is normally done. This is done by setting the UCB.URSKQ := OFFH. The stack should allow at least 80 bytes more for M/OS-80's use. A further switch is provided to tell the user that the disk driver should not be interrupted and hence to inhibit re-scheduling.

Using M/OS-80 call 81H returns the address of the User register (URREG), so after the call, BC contains as below.

BC+0	UCB	(2)	User Control Block Pointer
+2	SCHSW	(1)	Reschedule Switch
+3	SDVT	(2)	System Device Table
+5	SDT	(2)	System Disk Table
+7	SYSBOT	(2)	Bottom of System Area
+9	OPILINK	(2)	OPI Linkage Table

(ED. The current version of M/OS-80 does not support multiple users.)

3.10. CDOS AND CP/M COMPATIBILITY

M/OS-80 is an independent product of MOSTEK. However, because of the availability of programs for CP/M, great care has been taken to assure that all normal programs for CP/M Version 1.4 will run on M/OS-80. In addition, virtually all programs written for CP/M version 2.2 will also work. Most programs written for Cromemco's CDOS will also work on M/OS-80 due to the structure shared by the two systems.

MOSTEK Corporation does not claim, state or imply any approval of MOSTEK or M/OS-80 by Digital Research or Cromemco, and does not make any claims or warranties as to the suitability of any product produced for CP/M or CDOS to be run with M/OS-80.

M/OS-80 and CDOS have significant extensions beyond CP/M, all of which are distinguished from CP/M BDOS calls by the call function beginning greater than or equal to 128 (80 Hex).

WARNING:

Programs that "know" about the internal features or functions of CP/M and bypass or alter the operation of CP/M or take advantage of undocumented eccentricities of CP/M cannot reasonably be expected to run under M/OS-80. This means programs that go beyond the normal CP/M call at 5 and the branch table pointed to by location 0 will most likely not work.

MOSTEK has performed acceptance testing on a wide variety of application programs including several compilers, interpreters, word processors, editors and data base management systems. For a updated list of those programs, contact your local MOSTEK Field Applications Engineer.

CP/M is a trademark of Digital Research, Pacific Grove, Ca.
CDOS is a trademark of Cromemco, Mountain View, Ca.

3.11. BIOS OPERATIONS

For those programmers who cannot use the normal calls, as provided by the operating system through location 5, the following information is provided. Many CP/M programs use this method, although somewhat crude, to communicate with the operating system.

3.11.1. The CP/M Jump Table

The I/O system starts with an internal branch table to provide CP/M compatibility. The table is reached via the jump at 0: the address at locations 1 and 2 point to the start of the table.(1)

```
         JP      RBOOT
(1) ->   JP      WBOOT   ; USER ABORT
         JP      CSTAT   ; CONSOLE STATUS        (RETURN IN A)
         JP      CIN     ; CONSOLE INPUT         (RETURN IN A)
         JP      COUT    ; CONSOLE OUTPUT        (FROM C)
         JP      LOUT    ; LIST OUTPUT           (FROM C)
         JP      POUT    ; PUNCH OUTPUT          (FROM C)
         JP      RIN     ; READER INPUT          (FROM A)
         JP      HOME    ; HOME CURRENT DISK
         JP      SELDSK  ; SELECT DISK           (FROM C)
         JP      SETTRK  ; SET TRACK             (FROM BC)
         JP      SETSEC  ; SET SECTOR            (FROM BC)
         JP      SETBUFF ; SET I/O BUFF ADDRESS  (FROM BC)
         JP      READ    ; READ DISK          (RETURN STAT IN A)
         JP      WRITE   ; WRITE DISK         (RETURN STAT IN A)
         JP      LRDY    ; LIST STATUS        (RETURN STAT IN A)
         JP      SECTRN
```

4. M/OS-80 SYSTEM CALLS

4.1. SYSTEM CALL SUMMARY - NUMERICAL (CP/M)

Code Hex	Dec	Function	Parmeters (in -> out)
0H	0	System Reset	
1H	1	Read Console (with echo)	-> A=char
2H	2	Write Console	E=char
3H	3	Read Reader	-> A=char
4H	4	Write Punch	E=char
5H	5	Write List	E=char
6H	6	Unformatted Console I/O	E=out char/ E=-1 -> A=input character.
7H	7	Read I/O byte	-> A=I/O byte
8H	8	Set I/O byte	E=I/O byte
9H	9	Print Console Buffer	DE=Buff adr
0AH	10	Read Console Buffer	DE=buff adr
0BH	11	Check Console Status	-> A=255, if ready
0CH	12	Deselect Current Disk	
0DH	13	Reset Disk System	
0EH	14	Select Current Disk	E=(A=0,B=1..)
0FH	15	Open File	DE=Address of FCB -> A=-1 or File number
10H	16	Close File	DE=Address of FCB -> A=-1 or File number
11H	17	Search Dir. for File	DE=Address of FCB -> A=-1 or File number
12H	18	Search Dir. for Next File	DE=Address of FCB -> A=-1 or File number
13H	19	Delete File	DE=Address of FCB -> A=-1 or File number
14H	20	Read Next Record	DE=Address of FCB -> A=0 ok, or 1,2
15H	21	Write Next Record	DE=Address of FCB -> A=0 ok, or 1,2,-1
16H	22	Create File	DE=Address of FCB -> A=-1 or File number

SYSTEM CALL SUMMARY - NUMERICAL (CP/M)

Code Hex	Dec	Function	Parmeters (in -> out)
17H	23	Rename File	DE=FCB(old,new) -> A=-1 or File num
18H	24	Login Vector	->A=bit vector (disk A=lsb)
19H	25	Current Disk	-> A=disk number
1AH	26	Set Buffer Address	DE=New buffer address
1BH	27	Allocation Vector	->BC=Address of map,DE = clusters, A=sectors/cluster

4.2. SYSTEM CALL SUMMARY - NUMERICAL

The following calls are not supported by CP/M 1.4.

Code Hex	Code Dec	Function	Parmeters (in -> out)
1CH	28	Write Protect Disk	
1DH	29	Get W/P Vector	-> A = W/P vector
1EH	30	Set File Name Attrib	DE=FCB w/ atrib set
21H	33	Random Read	DE=random FCB -> A=0 ok, or 1,3,4,6
22H	34	Random Write	DE=random FCB -> A=0 ok, or 1,3,4,5,6
23H	35	Determine File Size	DE=random FCB -> A=-1 or File num
24H	36	Determine Random Record	DE=random FCB
80H	128	Read Console (no echo)	-> A=char
81H	129	Get User Register	BC=^URREG,SW,SDVT,SDT,SYSBOT
82H	130	Set User Ctrl-C	DE=abort,0-normal,-1=dis-able
83H	131	Read Logical Block	DE=blk, b=disk (msb if IL)
84H	132	Write Logical Block	DE=blk, b=disk (msb if IL)
85H	133	Spool Control	DE=Spooler FCB
86H	134	Format Name to FCB	DE=FCB, HL=string
88H	136	Chain to Program	DE=FCB
89H	137	Multiply	DE*HL -> DE
8AH	138	Divide	(HL/DE)->HL,BC (HL mod DE)->DE
8BH	139	Home Driver	B=disk
8CH	140	Eject Diskette	E=disk
8DH	141	Get Version,Ser.no.	-> B=major rev./ C=minor rev. H=Sub-Maj./L=Sub-Minor DE=Serial Number
8EH	142	Set CRT Function	D=col ,E=row or D=spec,E=0
8FH	143	Set Date	B=day, D=mon, E=yr-1900
90H	144	Read Date	-> A=day, B=mon, C=yr-1900
91H	145	Set Time	B=sec, D=min, E=hr (24)
92H	146	Read Time	-> A=sec, B=min, C=hr (24)
93H	147	Set Program Return Code	A =code

SYSTEM CALL SUMMARY - NUMERICAL

The following calls are not supported by CP/M 1.4.

Code Hex	Dec	Function	Parmeters (in -> out)
94H	148	Set File Attributes	DE=fcb, B=atrib(PWRU/Sxx*)
95H	149	Read Disk Label	DE=fcb
96H	150	Turn Motors Off	
97H	151	Set System Bottom	E=Hi byte of address
98H	152	Read W/O Advance	DE=^FCB -> A=0 ok, or 1,2
99H	153	Write W/O Advance	DE=^FCB -> A=0 ok, or 1,2,-1
9AH	154	Test Block Allocated	
9CH	156	Directory Listing	DE=FCB to match
9DH	157	Set Console Options	E=options byte
9FH	159	Get Master Disk	-> A=master disk
A0H	160	Mount Disk	E=disk
A1H	161	Dis-mount Disk	E=disk

4.3. SYSTEM CALL SUMMARY - CATEGORIES

4.3.1. General System Functions

HEX	DECIMAL	FUNCTION
0	0	Return to System
81H	129	Get user register
89H	137	Multiply
8AH	138	Divide
8DH	141	Get Version number
97H	151	Set System Bottom

4.3.2. Console I/O Functions

1	1	Get Console (and echo)
6	6	Get/ Put Console (raw/ no echo)
0AH	10	Input Buffered Line from Console
0BH	11	Test Console Character Ready
80H	128	Get Console (no echo)
2	2	Put Console
9	9	Print Buffer to Console
82H	130	Set User Control-C Handling
8EH	142	Set Cursor or Terminal Function
9DH	157	Set Console Options
3	3	Get Reader
4	4	Put Punch
5	5	Put List
7	7	Get I/O byte
8	8	Set I/O byte

4.3.3. Date and Time

8FH	143	Set Date
90H	144	Read Date
91H	145	Set Time
92H	146	Read Time

4.3.4. Disk Functions

1AH	26	Set Disk Buffer
88H	136	Chain to Program
86H	134	Format FCB from String
93H	147	Set Program Return Code

4.3.5. Disk Select

0CH	12	Deselect Current Disk
0DH	13	Reset DOS Select A Drive
0EH	14	Select Disk
1CH	28	Write Protect Disk
1DH	29	Get Write/Protect Vector
8CH	140	Eject Disk
96H	150	Turn Motors Off
A0H	160	Mount Disk
A1H	161	Dis-mount Disk

4.3.6. Directory Maintenance

11H	17	Search Directory
12H	18	Search Next Entry
17H	23	Rename Files
1EH	30	Set File Name Attributes
94H	148	Set File Attributes
9CH	156	Directory Listing

4.3.7. Disk Status Functions

18H	24	Disk Login Vector
19H	25	Current Disk
1BH	27	Disk Cluster Allocation Map
95H	149	Read Disk Label
9FH	159	Get Master Disk

4.3.8. Files

0FH	15	Open File
10H	16	Close File
13H	19	Delete Files
16H	22	Create File

4.3.9. File Data Functions

14H	20	Read File Block
15H	21	Write File Block
21H	33	Random Read
22H	34	Random Write
23H	35	Determine File Size
24H	36	Determine Random Record
98H	152	Read Without Advance
99H	153	Write Without Advance
9AH	154	Test Current Record Allocated

4.3.10. Direct Disk Access

8BH	139	Home Disk Drive
83H	131	Read Logical Block
84H	132	Write Logical Block

4.4. FUNCTION CALLS

M/OS-80 System Calls are designed to permit the systems-level programmer to directly interface with the various primitive functions of the operating system. These calls permit direct interaction with the various hardware devices attached to the system. To use them, a thorough understanding of the operating system and the hardware devices is usually necessary, as well as a fairly high level of understanding of assembly language.

All calls require that the system function code be loaded into system register C. The application program requesting the operating system service then makes a CALL to location 5. The operating system will then perform the function(s) as requested and return (if necessary) to the application. The application will not receive control again until the operating system has completed the requested function.

4.4.1. Function Details

Additional parameters passed to and from the operating system are passed in the other registers. Details of that protocol are shown below:

Format

Title:	Function code in hex and decimal. Function Description.
Entry parameters:	What registers are passed to function.
Return Parameters:	What registers or information are passed from function.
Example:	Example shows function use.

For purposes of this manual, the Z80 register set is noted by enclosing the register or register pair in braces [] to indicate that the register(s) specified contain the passed information. Registers enclosed in parenthesis indicate that the registers indicated contain the <u>address</u> of the passed information.

For example, [HL] would indicate the register pair HL contains the parameter while (HL) would indicate that the register pair HL contains the address of the parameter.

4.4.2. 0 - System Return

Returns control to the system.

Entry parameters: None.

Return Parameters: None.(Does not return).

Example:

```
LD    C,0
CALL  5
  --OR--

JP    0          ;Return to OS.
```

4.4.3. 1 - Get Console Character (and echo)

Gets a character at console keyboard. If no character is ready, function waits until character is available.

Entry parameters: None.

Return Parameters: Register [A] is returned with character (ASCII).

Example:

```
LD    C,1
CALL  5
           ;   [A] = Character
```

4.4.4. 2 - Put Console

Sends a character to the console output device, expands tabs and controls printer daisy-chain.

Entry Parameters: [E] has character to output.

Return Parameters: None.

Example:

```
LD  C,2
LD  E,'A'       ; output 'A'
CALL 5
```

4.4.5. 3 - Get Reader

Gets character from "reader" device (if present). If none ready, then system waits until character ready.

Return Parameters: If Zero flag clear, then [A] has character input at end of file: Z=set, A=1AH (^-Z).

Example:

```
LD    C,3
CALL  5
CP    1AH
JP    Z,EOF
```

If the reader driver supports binary then:

```
LD    C,3
CALL  5
JP    Z,EOF
```

4.4.6. 4 - Put Punch

Sends a character to punch device, with no processing.

Entry parameters: Register [E] has character to output.

Example:

```
LD    C,4
LD    E,'A'
CALL  5
```

4.4.7. 5 - Put List

Sends a character to list device, with no processing.

Entry Parameters: Register [E] has character to output.

Example:

```
LD    C,5
LD    E,'A'
CALL  5
```

4.4.8. 6 - Put/Get Console Unconditionally.

Format (1): Gets a character from the console device without echo. System does not wait for a character to be input.

Format (2): Send character to console without processing.

```
Format (1):          (Input)
Entry Parameters:    [E] has OFFH (255).
Return Parameters:   [A] has character input if one is available or
                     zero if no character is waiting.

Format (2):          (Output)
Entry Parameters:    [E] has character to output not equal to OFFH

Example (Format 1):(INPUT)
                     LD    C,6
                     LD    E,OFFH
                     CALL 5
                                    ; A = character input

Example (Format 2):(OUTPUT)
                     LD    C,6
                     LD    E,'B'
                     CALL 5         ; output 'B' to console
```

4.4.9. 7 - Get I/O Byte

Returns the current value of the I/O Byte in [A].

```
Return Parameters: [A] has I/O Byte.

Example:             LD    C,7
                     CALL 5
                               ; A = I/O Byte
```

4.4.10. 8 - Set I/O Byte

Sets a new value for the I/O Byte.

```
Entry Parameters: [E] has I/O Byte value.

Example:             (To select console 3)
                     LD    C,8
                     LD    E,3
                     CALL 5
```

4.4.11. 9 - Print Buffer to Console.

Sends a character string to the console. The string is terminated by a '$' character. Standard console processing is done: tab expansion and printer daisy chain.

Entry Parameters: (DE) has address of string ended by '$'.

Example:

```
LD    C,9
LD    DE,STRING
CALL 5
      ...

STRING: DEFM 'THIS IS A STR $ '
```

4.4.12. 0AH/10 - Input Buffered Line from Console.

Reads a line of text from the console, allowing editing until termination by RETURN or LINE FEED. The call expects a buffer that starts with 2 length bytes, the first holding the maximum length; the second holding the returned true length. The buffer can have up to 255 characters. The character position after the last character in the string is zeroed. Input terminates either on a terminator or when buffer is filled. If the buffer is filled, then no final null is placed.

The terminator character is echoed with a RETURN, but no following LINE FEED is sent. The terminator may be an ESCAPE, and the RETURN echo may be suppressed.

See: Set Console Options (9DH).

```
                buffer format:
 -------------------------------------------
| full len | returned len | x | x | x | .....
 -------------------------------------------
```

Entry Parameters: (DE) has address of line buffer where format is as above.

Example:

```
        LD    C,10
        LD    DE,BUFF
        CALL 5
        ;
        LD    C,2            ; add line feed
        LD    E,10           ; 10 is ASCII line feed
        CALL 5
        LD    A,(BUFF+1)
                       ; get length of text entered
        LD    HL,BUFF+2
                       ; point to start of data

BUFF:   DEFB    80,0
        DEFS    80
```

4.4.13. OBH/11 - Test Console Character Ready.

Tests console ready to see if any characters are waiting. The waiting character may be a ^-S or control-P and hence not return character ready, yet cause the related action to be affected.

Return Parameters: If no character ready then [A] has zero (0). If character waiting then [A] has OFFH (255).

Example:

```
LD    C,11
CALL  5
OR    A
JP    Z,NOTRDY        ; not ready
                      ; ready
```

4.4.14. OCH/12 - Deselect Current Disk. Return CP/M Version.

Deselect current disk and clear all related disk-status and data buffers. Return CP/M Version in register pair HL. Value returned in HL is always 002FH in this version.

Entry parameters: None.

Return Parameters: [HL] has 002F for version number.

Example:

```
LD   C,12
CALL 5
```

4.4.15. ODH/13 - Reset DOS/Select A drive.

Reset all DOS disk functions. Returns disk buffer to 80H. Clears all disk buffers. Assures that DOS is initialized for any new disks to be loaded.

Entry parameters: None.

Return Parameters: None.

Example:

```
LD   C,13
CALL 5
```

4.4.16. OEH/14 - Select Current Disk.

Select current disk, which is used for disk operations, when no specific disk is specified. No disk activity takes place until the disk is actually accessed. If a disk higher than those in the system is requested, then no change in the current disk occurs.

Entry Parameters: [E] has disk number (0-A,1-B,...)

Return Parameters: None.

Example:

```
LD   E,1        ;Select Drive B
LD   C,14
CALL 5
```

4.4.17. OFH/15 - Open File.

The file described by the File Control Block (FCB) passed in DE is opened for access. The file must already exist to be opened. The FCB must be setup as described in section 3. (See **Creating a File Control Block**). A file must be either opened or created before it may be accessed.

Entry Parameters: (DE) has address of FCB of existing file.

Return Parameters: If no file found then [A] has OFFH (255); else [A] has directory block number of file found (0..3). There is one directory block number (starting at 0) for every four directory entries. The (DE) register pair points to the directory entry in RAM memory.

4.4.18. 10H/16 - Close File.

The file described by the File Control Block (FCB) pointed to by the DE register pair is closed and the disk directory updated. If the file cannot be found, an error code is returned. Files must be closed so that RAM FCB information can be posted to the disk directory.

Entry Parameters:	(DE) has address of FCB of existing opened file.
Return Parameters:	[A] has -1 (255) if no file found or [A] has file number of file found (0..3). See OPEN (above) for explanation of file numbers.

4.4.19. 11H/17 - Search Directory

The disk directory is searched for the first occurrence of the file name specified in the FCB addressed by the register pair DE.

The FCB passed can have match characters so that the SEARCH/SEARCH NEXT call can return successive files with similar names. The match character "?" (3FH) can be in the disk file attribute byte (FCB+0), the name (+1..+8), the file type (+9..+11), or the extent byte (+12). The name and type matching is simple ASCII string matching. If disk byte contains "?", then all ASCII characters will match regardless of letter. This will return a set of files with ambiguous names. If the extent byte contains "?", then all extents will be returned.

NOTE:	This call and the next call (Find Next Directory) will return directory entries whether they have been erased or not. Files are marked as being erased by replacing the attribute byte (FCB+0) with an 0E5H.
Entry Parameters:	(DE) has FCB with search argument as described above.
Return Parameters:	[A] has -1 (255) if no file found, or [A] has the file number of file the found (0..3). See OPEN for discussion of file numbers. (HL) has address of file directory entry in disk buffer.

4.4.20. 12H/18 - Search Next Entry.

Search Next is used to find the next directory entry for the FCB name described in the Search Directory call (11H) described above. Also see NOTE above.

Entry parameters:	None required. No system calls may come between this call and previous call 11H or 12H.
Return Parameters:	Same as 11H Search directory call except starts at next entry after last search next.

4.4.21. 13H/19 - Delete Files.

Deletes all files whose names match the FCB pointed to by the DE register pair. The FCB may contain match characters as described in Search Directory (11H) above.

Entry Parameters:	(DE) has address of FCB of files to delete.
Return Parameters:	[A] has file number of last file deleted, or -1 (255) if no files found.

4.4.22. 14H/20 - Read File Block.

Read the file block (128 bytes) indicated by the FCB next-record byte into the current disk buffer. After reading, advance the FCB to point to the next record. At the start of reading, if necessary, open a new extent. To read sequentially, the next-record byte must be initialized to zero before the requesting read.

Entry Parameters: (DE) has address of FCB of an opened file.

Return Parameters: [A] has:
- 0-Read completed.
- 1-End of file sensed.
- 2-Read attempted on unwritten cluster (Random-access files only).

4.4.23. 15H/21 - Write File Block.

Write the file block (128 bytes) indicated by the FCB next-record byte. After writing, advance the FCB to point to the next record and, if necessary, create a new extent. To write sequentially, the next-record byte must be initialized to zero before the start of writing.

Entry Parameters: (DE) has address of FCB of an open file.

Return Parameters: [A] has: 0-Write completed.
1-Extent error (error on writing).
2-Out of disk space.
-1-(255) Out of directory space on creating new extent.

4.4.24. 16H/22 - Create File.

The file described by the File Control Block (FCB) passed in the register pair (DE) is created and initialized for access. The file cannot already exist, and the directory is checked. The FCB must be setup as described in section 3. (**See Building a File Control Block**) A file must be either opened or created before it may be accessed.

Entry Parameters: (DE) contains address of an FCB of non-existing file.

Return Parameters: [A] has -1 (255) if no directory room or file number of file created. See OPEN for discussion of file numbers.

4.4.25. 17H/23 - Rename Files.

Renames all files whose names match the FCB given. The FCB may contain match characters, in which case search is done first to test for possible duplicate name creation.

Entry Parameters: (DE) has address of FCB of files to rename. The first FCB contains old name, and the FCB contains the new name. (Requires two contiguous FCBs)

Return Parameters: [A] has file number of last file deleted, or -1 (255) if no files found.

4.4.26. 18H/24 - Disk Login Vector.

The call returns a bit vector of disks currently active. Each bit indicates a logged-in disk with least significant bit (LSB) indicating disk A.

Entry parameters: None.

Return Parameters: [HL] has bits for active drives.

The Format For Disks are as follows:

```
L:  /HGFE/DCBA/
H:  /xONM/LKJI/
```

x Represents non-existent drive

NOTE: To maintain compatibility with early versions, registers A and L contain the same values.

4.4.27. 19H/25 - Get Current Disk.

The call returns the number of current disk, A=0, B=1,... etc.

Entry parameters: None.

Return Parameters: [A] has current disk.

4.4.28. 1AH/26 - Set Disk Buffer.

Set the RAM memory address of the buffer to be used for disk data operations and for directory search calls. Internal directory functions use special system buffers, not this user buffer. The initial location of this buffer is 080H

Entry Parameters:	(DE) has address of disk buffer.
Return Parameters:	None.

4.4.29. 1BH/27 - Disk Cluster Allocation Map.

Returns the starting address of the disk allocation map (bit map). Allocation maps are for system program use. For big directories, the argument returned contains the buffer pointer for the disk bit map. Bit maps return size information about the disk format that helps in displaying a disk directory.

Entry Parameters:	None.
Return Parameters:	(BC) returns address of RAM bit map (1 bit per cluster). [DE] returns number of clusters. [A] returns number of sectors per cluster. (HL) also returns address of RAM bit map.

4.4.30. 1CH/28 - Write-Protect Disk.

Sets the current disk to logical write-protected. Allows a program to totally inhibit write access to a disk. Holds until the disk is logged in again.

Entry Parameters: None.

Return Parameters: None.

4.4.31. 1DH/29 - Get Write-Protect Vector.

Returns the write-protect bit-vector. The bit order is the same as function 24. The vector bits are set either by the system directory mechanism when it finds the disk has been changed without a proper login, or by a program using function 28.

The vector format for disks are as follows:

```
L:  /HGFE/DCBA/
H:  /xONM/LKJI/
```

Return Parameters: [HL] has write-protect bits.

4.4.32. 1EH/30 - Set File Attributes.

Allows an alternate file attribute function to system function 94H. The high bits of the first 2 bytes of the file type (FCB+9,+10) are used to set/reset or the write-protect attribute and system attributes of the files whose name matches the FCB given.

Entry Parameters: (DE) has address of FCB with requested attribute set.

Return Parameters: [A] returns file number of files found or -1 (255) if no files found. See OPEN for discussion of file numbers.

4.4.33. 21H/33 - Random Read.

Provides a means for random file access without any additional file processing. A random FCB is required. (See section 3.2.3) The random record pointer in the FCB (FCB+33,+34,+35) is set to the record to be read and this function is called. No change is made to the record pointer; however, next record pointer and extent may be changed as required to position within the file.

For sequential reading, the random record pointer must be incremented by the user. To mix random and sequential reading and writing, function 36H should be used after sequential accesses to reset the random record pointer for future random reads.

Entry Parameters:	(DE) has address of opened random FCB set to record to be read.	
Return Parameters:	[A] has:	0- Read OK.
		1- Reading unwritten data.
		3- Cannot close current extent.
		4- Seek to unwritten extent.

4.4.34. 22H/34 - Random Write.

Provides a means for random file access without any additional file processing. A random FCB is required. (See section 3.2.3) The random record pointer in the FCB (FCB+33,+34,+35) is set to the record to be written and this function is called. No change is made to the record pointer; however, the next record pointer and extent may be changed as required to position within the file.

For sequential writing, the random record pointer must be incremented by the user. To mix random and sequential reading and writing, function 36H should be used after sequential accesses to reset the random record pointer for future random reads.

Entry Parameters:	(DE) has address of opened random FCB set to record to be read.
Return Parameters:	[A] returns: 0- Write OK. 3- Cannot close current extent. 4- Seek to unwritten extent. 5- Directory full. 6- Disk full.

4.4.35 23H/35 - Determine File Size.

Provides a means of appending to a file. A random FCB is required. The random record pointer in the FCB (FCB+33, +34, +35) is set to the record past the last record in the file. Writing may proceed from this point by just writing normally.

For text files, the last previous record should be read and writing started at the logical end-of-file marker (^-Z).

Entry Parameters: (DE) has address of random FCB for an open file.

Return Parameters: FCB random-record pointer set.

4.4.36 24H/36 - Determine Random Record.

Provides a means for mixing sequential and random file access. The record pointer is set based on the next record pointer and extent. A random FCB is required.

To mix random and sequential reading and writing, this function should be used after sequential accesses to reset the random record pointer for random accesses.

The function is also useful while reading or writing a file sequentially to determine the random record pointer for use in building an index file.

Entry Parameters: (DE) has address of random FCB for an open file.

Return Parameters: FCB random-record pointer set.

4.4.37 80H/128-Get console (no echo)

Inputs character from the console, without echoing it back to the CRT.

Entry parameters: None.

Return Parameters: [A] has character input.

Example:
```
LD C,128
CALL 5
```

4.4.38 81H/129 - Get User Register

Returns a pointer to the system user register. User registers are primarily used for system programs.

```
                (# of bytes)
   (BC)  -> URREG  (2)
            SCHSW  (1)
            SDVT   (2)
            SDT    (2)
            SYSBOT (2)
            OPILINK(2)
```

Entry Parameters: None.

Return Parameters: (BC) has address of URREG.

Example:
```
LD      C,81H
CALL    5
      ; BC=ADDRESS OF URREG
```

4.4.39 82H/130 - Set User ^-C Handling

This call sets the system's response to the user pressing ^-C. See: **Abort (control-C) Control** for additional information.

Entry Parameters: [DE] has:

```
0     - Restore normal system action.
-1    - Disables Control-C function. (OFFFFH).
Other - Set cntl-C routine address.
```

Example:
```
LD     C,82H
LD     DE,UCTRLC
CALL   5
...
UCTRLC: ; USER CONTROL-C ROUTINE
```

4.4.40. 83H/131 - Read Logical Block.

This function reads a logical block from the disk without regard to any file structure. Therefore, logical blocks can be used to read disks created with other operating systems. When the function is invoked, the DE register pair should contain the block number desired. Data is read from the disk I/O buffer. In summary, Read Logical Block allows direct disk access without standard file processing.

Entry Parameters: [B] has Drive number (0-Current, 1=A, 2=B...) For block interleaving, set high bit in reg. B. [ADE] has Block 0..xxxxxx (24-bit number).

Return Parameters: [A] Returns:
0-OK
1-I/O error
2-Illegal request
3-Illegal block number

4.4.41. 84H/132 - Write Logical Block.

This function writes a logical block to the disk without regard to any file structure. Therefore, logical blocks can be used to write to disks created with other operating systems. When the function is invoked, the DE register pair should contain the block number desired. Data is written from the disk I/O buffer. In summary, Write Logical Block allows direct disk access without standard file processing.

Entry Parameters: [B] has Drive number (0-Current, 1=A:, 2=B:...) For block interleaving, set bit 7 in [B]. [ADE] has Block 0..xxxxxx (24-bit number).

Return Parameters: [A] Returns:
0-OK
1-I/O error
2-Illegal request
3-Illegal block number

4.4.42. 85H/133 - Print Spooling Control.

Controls the printer spooler, thereby allowing simultaneous printing and other processing. The spooler can be either stopped or started, or queried under system control. Printer calls (function 5) cannot be used properly while the spooler is active.

Starting the spooler requires passing an opened FCB to this function.

To test if a system has a spooler installed, the status call may be called with [A] = 2. If no spooler exists, then [A] will not change, otherwise it will be either 0 or -1.

Entry Parameters: (DE) has: FCB of spool file (to start spooler)
[A] has: 0 - to stop spooler
1 - to get status

Return Parameters: After status call only:
[A] returns:
0 - not active
-1 - active

4.4.43. 86H/134 - Format FCB From String.

Formats a standard FCB from a text string. This is used to setup an FCB from scratch.

The filename will be terminated by the first occurrence of: space, non-printing character, equal sign (=), slash (/) or comma (,).

Entry Parameters: (DE) has address of where FCB is to be built.
(HL) has address of string with filename.

Return Parameters: None.

Example:

```
 LD C,86H
 LD DE,FCB
 LD HL,STR
 CALL 5
...
FCB:    DEFS 33
STR:    DEFM '<gname>'
                          ;<gname>=name.ext
        DEFB  0
```

4.4.44. 88H/136 - Chain to Program.

Allows one program to chain to another. The standard system region from 5CH to 0FFH is not modified. If control-C was disabled by the chaining program, then control-C will remain disabled; however, if the prior program set a control-C routine, it will be reset. If the program will not fit into available RAM, the system returns to the operator with the message "Load error". The "chained-to" program will receive whatever return code was set into the A register; if none was set, [A] will have zero.

Entry Parameters: (DE) has address of FCB containing filename of program to run.

Return Parameters: [A] returns -1 if file not found.

Example:

```
    LD C,88H
    LD DE,PGMN
    CALL 5
          ; ONLY RETURNS HERE ON ERROR
  ...
    PGMN:    DEFB     0
    ; NOTE FCB FORMAT OF NAME
             DEFM     "NEWPROGCOM"
```

4.4.45. 89H/137 - Multiply.

Provides a basic 16-bit unsigned multiply routine.

Entry Parameters: [DE] has multiplicand.
[HL] has multiplier.

Return Parameters: [DE] returns [DE] * [HL].

Example:

```
    LD     DE,20
    LD     HL,10
    LD     C,89H
    CALL    5
          ;  HERE DE=200
```

4.4.46. 8AH/138 - Divide.

Provides a basic 16-bit unsigned divide routine.

Entry parameters: [DE] has integer divisor.
[HL] has integer dividend.

Return Parameters: [HL], [BC] returned with [HL]/[DE].
[DE] returned with remainder ([HL] mod [DE]).

Example:

```
LD      DE,3
LD      HL,10
LD        C,8AH
CALL    5
                          ; HERE [DE]=1 AND [HL]=3
```

4.4.47. 8BH/139 - Home Disk Drive.

Forces disk heads to return to home position.

Entry Parameters: [B] has drive number (0=current, 1 = A:,2 = B: ...).

Return Parameters: None.

4.4.48. 8CH/140 - Eject Disk.

Request removal of disk indicated. Either physically ejects disk or requests operator to remove it, depending on disk hardware features.

Entry Parameters: [E]=disk (0=current, 1=A, 2=B...).

Return Parameters: None.

4.4.49. 8DH/141 - Get Version and Serial Numbers.

Returns the version and serial numbers of the system. Can be used by programs to check that required features are available.

Return Parameters: [B] has Major Version number.
[C] has Minor Version number.
[H] has Sub-Major Version number.
[L] has Sub-Minor Version number.
[DE] has Serial Number.

Example:

```
LD      C,8DH
CALL    5
; B=2, C=34 ,H=01, L=00
```

For current version 02.34 - 01.00

4.4.50. 8EH/142 - Set Terminal Function.

Provides a hardware-independent way of controlling an advanced CRT terminal. Section 3.2.2 has further details.

Entry Parameters: [D] has column and [E] has row to set cursor.
-or-
[E] has 0 and [D] has special function.

(For switch functions, on=odd, off=even) Special functions (in [D])

0-clear	1-home	2-backspace	3-forespace
4-up	5-down	6-clr eol	7-clr eos
8-highlight	9-low light	10-normal light	11-keyboard on
12-keyboard off	13-cur. pad on	14-cur. pad off	15-protect on
16-protect off	17-blink on	18-blink off	19-line send
20-page send	21-aux send	22-del char	23-insert char
24-del line	25-insert line		

Cursor pad translates:

```
            Up-^W
Left-^A            Right-^D
           Down-^Z
```

4.4.51. 8FH/143 - Set Date.

Allows a program to set the date stored in the system. All input is in BCD.

Entry Parameters:	[B] has Day
	[D] has Month
	[E] has Year (1900+)

4.4.52. 90H/144 - Read Date

Allows a program to read the date stored in the system. All output is in BCD.

Return Parameters:	[A] has Day
	[B] has Month
	[C] has Year (1900+)

4.4.53. 91H/145 - Set Time.

Allows a program to set the time stored in the system. All input is in BCD.

Entry Parameters:	[B] has Seconds
	[D] has Minutes
	[E] has Hours (24-hr time)

4.4.54. 92H/146 - Read Time.

Allows a program to set the time stored in the system. All output is in BCD.

Return Parameters:	[A] has Seconds
	[B] has Minutes
	[C] has Hours (24-hr time)

4.4.55. 93H/147 - Set Program Return Code.

Sets a return code that gets passed to the next program on a chain function by way of [A]. The chained-to program will find that [A] contains the return code upon entry.

Entry Parameters: [A] has return code.

Example:

```
LD C,93H
LD A,10
CALL 5
      ; NEXT LINK WILL PLACE 10 IN "A"
```

4.4.56. 94H/148 - Set File Attributes.

Allows setting file attributes to restrict access to a file. Section 3.2.3 details the attributes.

Bit:	Attribute:
7 -	(P) Permanent
6 -	(W) Write-protected
5 -	(R) Read-protected
4 -	(U) User file
3 -	(S) System file
2 -	(.) n/a
1 -	(.) n/a
0 -	(+) Add attributes, instead of replace

Entry Parameters: (DE) has address of FCB of files in which to set attributes.
[B] has new attributes with bits (PWRU/Sxx+).

4.4.57. 95H/149 - Read Disk Label.

Places the disk label for the current disk into the FCB provided. It also places the directory type flags into A.

```
    Directory type flag:
bit 7 - (S) Sub-directory
    6 - (D) Double-sided disk
    5 - (.) n/a
    4 - (.) n/a
    3 - (L) Disk has a label
    2 - (.) n/a
    1 - (R) Big directory required by hardware
    0 - (B) Big Directory
```

Entry Parameters: (DE) has address of FCB which will be filled in with the disk name.

Return Parameters: [A] returns directory type flags.

4.4.58. 96H/150 - Turn Motors Off.

Allows turning the motors off on the current disk. Hardware determines how motors are affected.

4.4.59. 97H/151 - Set System Bottom.

Allows a user program to load a special driver or common section of code and then adjust the operating system bottom-end to include the new code. The function requires that the new bottom be below the original system bottom and to be on a page (256-byte) boundary. Note that the first nine bytes of the new system bottom are reserved for use by the system.

An address of 0FFFFH will cause the system bottom to be reset to the original value.

Entry Parameters: [E] has high byte of the address of the new system bottom.

4.4.60. 98H/152 - **Read File Block Without Advance.**

Read the file block indicated by the FCB next-record byte. After reading, no change is made to the FCB. This function is designed to allow random reading. The next record byte must be set to the proper value and the proper extent must be opened before reading. If extents must be switched, then the old extent must be closed before opening the new one.

Entry Parameters:	(DE) has FCB of an open file.
Return Parameters:	[A] Returns: 0-OK. 1-End-of-File sensed. 2-Date being read from an unwritten extent.

4.4.61. 99H/153 - **Write File Block Without Advance.**

Write the file block indicated by the FCB next-record byte. After writing, no change is made to the FCB. This function is designed to allow random writing. The next-record byte must be set to the proper value and the proper extent must be opened or created before writing. If extents must be switched, then the old extent must be closed before opening the new one.

Entry Parameters:	(DE) has FCB of an open file.
Return Parameters:	[A] has: 0 - OK. 1 - Extent error (error on writing). 2 - Out of disk space. -1 - (255) Out of directory space on creating new extent.

4.4.62. 9AH/154 - Test File Block Allocated.

Test if the file block specified is allocated. The current block of the FCB passed in DE is tested.

Entry Parameters:	(DE) has address of the FCB of an open file.
Return Parameters:	[A] returns: 0-Block allocated -1-Block not allocated

4.4.63. 9CH/156 - Directory Listing.

Display a standard directory listing for files matched by the FCB passed.

THIS FUNCTION IS ACTIVE ONLY IF OPI IS RESIDENT (see OPIRES); otherwise the error "DIR function not active" is displayed.

Entry Parameters:	(DE) has address of the FCB of file match request.

4.4.64. 9DH/157 - Set Console Options.

Allows setting certain special characteristics for console activity. Affects functions 1,2,6,9,10. Reference paragraph 3.2.2.

Option bit
- 7 - Disable print output toggle (^W/^T)
- 6 - No echo on any input
- 5 - n/a
- 4 - n/a
- 3 - No echo of RETURN as terminator
- 2 - Enable ESCAPE as line terminator
- 1 - Disk read after write
- 0 - Control-P Flag

The function allows changing only certain options by having a change-mask of bits to be changed and a new value for those bits as separate values.

Entry Parameters:	[E] has mask of bits to change. [D] has new values of changed bits.

4.4.65. 9FH/159 - Get Master Disk.

Returns the current values for the command library master (DCOM) and the batch file master (DBAT). The indicated commands are used to set these values (DCOM or DBAT).

The returned values are A=0, B=1, etc.

Return Parameter:	[A] has master disk (DCOM). [B] has batch master (DBAT)

4.4.66. A0H/160 - Mount Disk.

Requests the specified disk be mounted.

Entry Parameters:	[E] has disk number (A=0...).
Return Parameters:	[A] returns: 0-OK. -1-Disk already mounted. -2-I/O error on mount.

4.4.67. A1H/161 - Dismount Disk.

Requests the specified disk be dismounted. The disk will always be logically dismounted for the caller; however, in a multi-user system, the disk will only be physically dismounted after all users have dismounted it.

Entry Parameters:	[E] has disk number (A=0...)
Return Parameters:	[A] returns: 0-OK or, in multi-user systems, bit vector of users logged on the disk. (bit 0-user 1,bit 1-user 2...).

A. DDT-80 DEBUG SYSTEM

A.1. INTRODUCTION

This section describes the functions and operation of DDT-80 (Designer's Development Tool 80) which is provided with the M/OS-80 system. The DDT software provides a complete facility for interactively debugging relative and absolute Z80 programs. Standard commands allow displaying and modifying memory and CPU registers, setting breakpoints, and executing programs. Mnemonics are used to represent Z80 registers, thus simplifying the command language.

A.2. SOFTWARE CONFIGURATION

DDT-80 is a program that resides in EPROM (located from E000H to E7FFH). In addition to the EPROM, DDT uses 256 bytes of RAM which resides at locations FF00H - FFFFH.

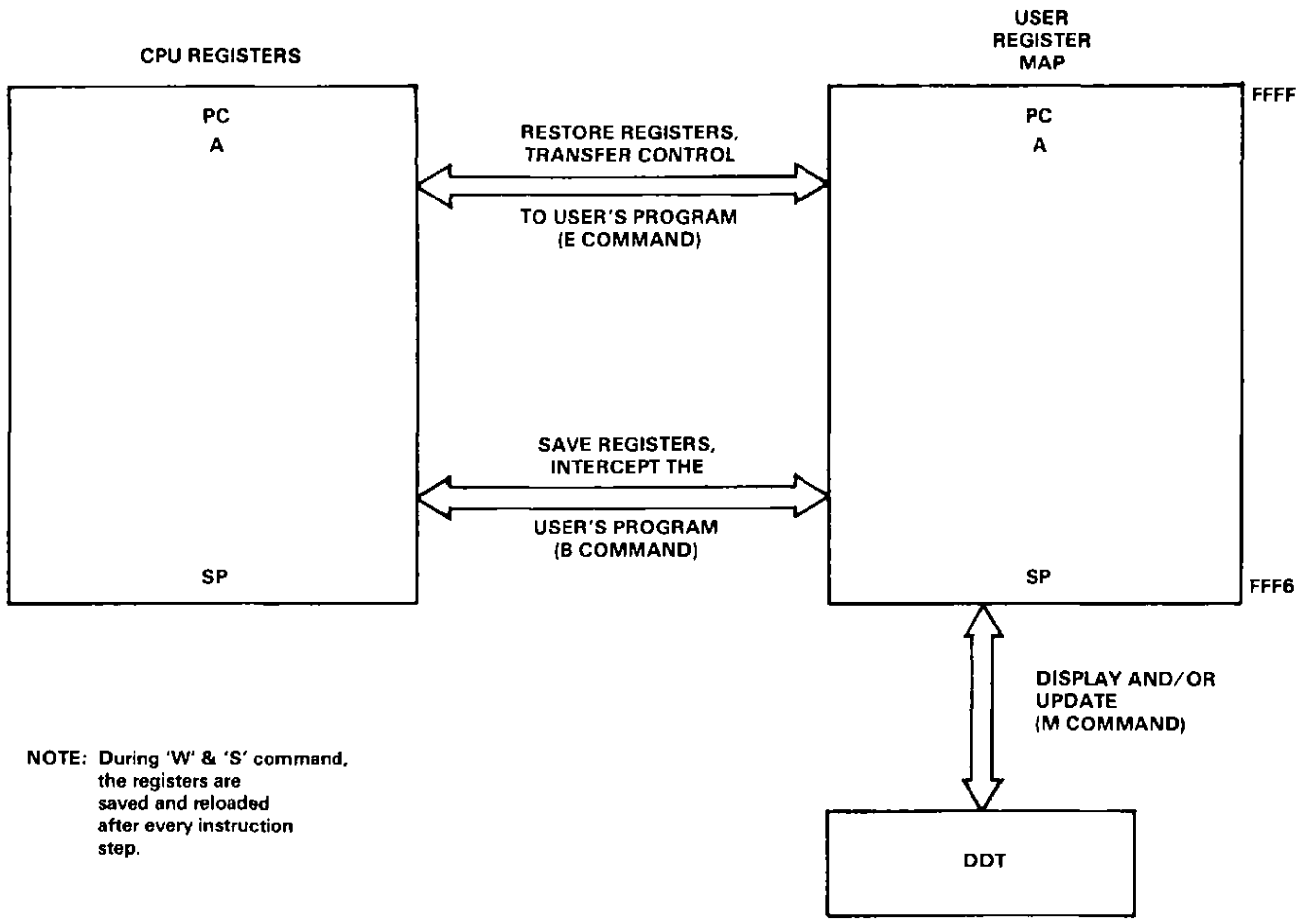
The scratchpad RAM is used by DDT for temporary storage and a push-down stack (for return address, etc.). This RAM also holds an image (or map) of all the user's internal CPU registers. Figure A-1 is a detailed memory map of the scratchpad RAM.

An important concept in DDT is preservation of the user's internal CPU registers. The state of the user program under CPU control is described by the contents of these registers and is kept in an image or map RAM area. This map is referred to as the User Register Map throughout this documentation. DDT installs or makes the CPU registers equal to the user register map when control is transferred from DDT to a user program (as in the E command discussed in paragraph A.13). DDT-80 saves the user register map when DDT is commanded to interrupt a user program (breakpoint command discussed in paragraph A.11). DDT allows modification of this register map with the display and/or update memory command (M command, discussed in paragraph A.17). The user register map resides in the scratchpad at locations FFE6H thru FFFFH as shown in Figure A-1 . Figure A-2 shows the data paths between the user register map and the CPU registers along with the modification path between DDT and the User Register Map.

Fig. A-1: User Register Map Diagram

MEMORY LOCATION	USER REGISTER	
FFFF	PC PROGRAM	MSB
FFFE	COUNTER	LSB
FFFD	A	
FFFC	F	
FFFB	I	
FFFA	IF	
FFF9	B	
FFF8	C	
FFF7	D	
FFF6	E	
FFF5	H	
FFF4	L	
FFF3	A'	
FFF2	F'	
FFF1	B'	
FFF0	C'	
FFEF	D'	
FFEE	E'	
FFED	H'	
FFEC	L'	
FFEB	IX	MSB
FFEA		LSB
FFE9	IY	MSB
FFE8		LSB
FFE7	SP STACK	MSB
FFE6	POINTER	LSB

Fig. A-2: Data Paths - User Registers and DDT


CPU REGISTERS
PC
A
SP
USER REGISTER MAP
FFFF
PC
A
SP
FFF6
RESTORE REGISTERS, TRANSFER CONTROL
TO USER'S PROGRAM (E COMMAND)
SAVE REGISTERS, INTERCEPT THE
USER'S PROGRAM (B COMMAND)
DISPLAY AND/OR UPDATE (M COMMAND)
DDT

NOTE: During 'W' & 'S' command, the registers are saved and reloaded after every instruction step.

A.3. COMMAND SUMMARY

To invoke DDT in systems without the DDT firmware (Phantom PROM or hard disk systems, type the following command while in the M/OS-80 command mode:

```
A.DDT(CR)
```

To invoke DDT in system using the DDT firmware, type the following command while in the M/OS-80 command mode:

```
A.EXEC E11D(CR)
```

A.3.1. Console Interaction

.	Prompt character.
(CR)	Terminate a command.
. or cntl-U	Abort.

A.3.2. Commands

B	aaaa	Insert a breakpoint in user's program.
C	aaaa,bbbb,cccc	Copy memory aaaa thru bbbb to cccc and above.
E	aaaa	Execute user program.
F	aaaa,bbbb,cc	Fill memory aaaa thru bbbb with data byte cc.
H	...	Perform hexadecimal arithmetic.
L	aaaa,bbbb,cccc	Locate all occurrences of data cccc in memory aaaa thru bbbb.
M	aaaa, bbbb	Display, update, or tabulate memory or registers.
O	aaaa	Set offset constant for relocatable programs.
P	aa	Display and update port.
Q		Quit - return to Monitor.
R	a,b	Display user registers.
V	aaaa,bbbb,cccc	Verify that two blocks of memory are identical.
W	aaaa,bb	Single step starting at address aaaa for bb steps.

NOTE: Commas may be replaced with blanks. All entries upper-case ASCII.

Fig. A-3: DDT Commands

A.4. NOTATION CONVENTIONS

Hexadecimal numbers are denoted by the number followed by a subscript H. E.g., 0AF3H. If the first digit of the hexadecimal number is a letter (A-F), the hexadecimal number must be preceded by a zero (0). In a command sequence, user input is underlined. (CR) means carriage return. Bracketed items [] in a command line are optional. Items in a command line which must be entered exactly as they appear are shown as upper case. Items in a command line which are variables are shown as lower case.

A.5. PREPARATION

Create, assemble, and link your program as would be normally done.

You should now be ready to debug a binary file which has your Z80 program on it. To debug the program, use the LOAD program to get the program loaded into RAM.

```
        A.LOAD file (CR)
```

Where file is the name of the binary .COM file created by the LINK process or INTEL Hex file created by older assemblers or the LINK process.

Then execute DDT:

```
        A.DDT (CR)

        .
```

The dot (.) indicates that DDT is ready to accept commands.

A.6. DESCRIPTION OF DDT COMMANDS

A.6.1. Command Format

DDT recognizes commands which consist of three parts:

1. A single-letter command.
2. An operand or operands separated by commas or blanks.
3. A terminator to either abort the command or cause it to be executed.

```
            Example
            .M 100, 102 (CR)

  (Parts:)  1. 2.       3.
```

In the command mode DDT prompts on the user console with a dot (.). The user may enter any single-letter command. A space is then printed on the console by DDT. The user may then enter any required operands and a terminator. Operands are separated from each other by a space or a comma. The terminator may be a carriage return, dot (.) or control-U. Carriage return causes execution of the command. A dot or control-U aborts the command, and the user is prompted again.

A.6.2. Operands

Operands are separated from each other by a space or comma. An operand may take any one of the following forms:

Hexadecimal number. Leading zeros need not be entered. The last four digits are used for the value entered for address values. The last two digits are used for data values.

ASCII literal value. Any characters preceded by the letter "L" are converted to their ASCII equivalent value. E.G., LA(=41H), LAB(=4142H).

Relative Address. A hexadecimal number preceded by the character "R" causes the offset specified by the O command to be added to the number. A relative address is identified by an apostrophe next to it. E.g., (assuming offset = 100H) R0(=100H), R4FF(=FFH).

The offset and relative address functions are useful when debugging modules of a program which have been relocated by the Linker.

Program Counter. The character "$" is used to represent the current address. It is used with the M command to calculate relative branch displacements.

Added or subtracted numbers. Hexadecimal numbers may be added to or subtracted from each other to represent an operand. E.g., A = A (=14H), 5A = A - 10 (=54H).

Equal Sign. An equal sign (=) may be entered at any time to display the current value of an operand as 4 hexadecimal digits. E.g., 5A = A - 10 = 0054, LAC = 4143.

Mnemonic. A mnemonic consists of one or two characters following a colon (:). Mnemonics are used to represent Z80 CPU registers. Figure A-4 lists all the allowed mnemonics in DDT and their meanings.

MNEMONIC	ADDRESS REPRESENTED BY THE MNEMONIC	DATA SAVED AT THAT ADDRESS
:PC*	FFFE	User's PC Register
:A	FFFD	User's A Register
:F	FFFC	User's F Register
:I	FFFB	User's I Register
:IF*	FFFA	User's IFF Register
:B	FFF9	User's B Register
:C	FFF8	User's C Register
:D	FFF7	User's D Register
:E	FFF6	User's E Register
:H	FFF5	User's H Register
:L	FFF4	User's L Register
:A'*	FFF3	User's A' Register
:F'*	FFF2	User's F' Register
:B'*	FFF1	User's B' Register
:C'*	FFF0	User's C' Register
:D'*	FFEF	User's D' Register
:E'*	FFEE	User's E' Register
:H'*	FFED	User's H' Register
:L'*	FFEC	User's L' Register
:IX*	FFEA	User's IX Register
:IY*	FFE8	User's IY Register
:SP*	FFE6	User's SP Register

* = two-byte mnemonics

Fig. A-4: Mnemonics Recognized By DDT-80

Unrecognized mnemonics are resolved with a value of zero.

A.7. OPERAND EXAMPLES

4F7F	The operand value is equal to 4F7FH.
:PC	The mnemonic PC is equivalent to the save location of the user's program counter.
5038-5000	The operand value is 38H,
5038-5000=0038	The same as above except "=" was entered to display the operand value.
5038-$	If current address = 5000H, then $=5002H and the operand value equals 36H for relative instructions.
5038-$=0036	The same as above except the equal sign was entered.
305038	More than 4 digits entered, therefore only the last 4 have meaning. Operand value = 5038H.
305038=5038	The same as above except the equal sign was entered.
LAB=4142	Operand is equal to the ASCII value of "AB".
LA=2041	Operand is equal (LSB) to ASCII value of 'A'.
R100=1100	Assumes offset = 1000.

A.8. COMMAND TERMINATORS

The command terminator immediately follows the operand(s) and signals DDT that the command has been entered. Depending on the terminator, DDT will do one of the following:

Terminator	Action
(CR)	Carriage return. DDT executes the entered command.
. or CNTL-U	Period or CNTL-U. DDT aborts the command. The user is prompted for another.
^	Carat or up arrow. This terminator is valid only for the M and P commands. To examine the previous memory location or port relative to the location just examined.
/	Slash. This terminator is valid only for the M command. This causes the data entered to replace the old data and then return to the command mode. If no data was entered, it is treated as a period.

A.9. SPECIAL KEYS

Several keys have special meaning in DDT:

period (.)	Memory printouts on the console (L, M, or V commands) may be aborted by entering a period. Single stepping (W command) may also be aborted this way. DDT then enters the command mode.
space bar	The space bar may be used to start and stop single stepping (W command).

A.10. ERRORS

When erroneous inputs are detected, a question mark (?) is printed and DDT returns to the command mode.

A.11. B (BREAKPOINT) COMMAND

FORMAT

.B aaaa (CR)	Set breakpoint at memory address aaaa.
.B (CR)	Clear previous breakpoint.

OVERVIEW.

When the breakpoint command is used, a "trap", which consists of three bytes, is placed into the user's program. The original program bytes are automatically saved. The user then uses the E (execute) command to start execution of the program. When the trap is encountered, DDT is signalled and execution is stopped. The CPU registers are then transferred to DDT and printed out on the user console. To resume execution of the program, the user must use the E (execute) command again or the W (single step) command.

DESCRIPTION.

The user types the command identifier B followed by the address where it is desired to place a breakpoint "trap". DDT proceeds to remove any pre-existing breakpoint, extracts and saves 3 bytes of the user's program at the breakpoint address, and places a 3-byte trap into the address. DDT then returns to the command mode. The user may start program execution via the E (execute) command. When the breakpoint trap is encountered, execution is stopped and control is transferred back to DDT. DDT then restores the three bytes of user code at the breakpoint address, reads all the target CPU registers and prints them out (see R-register command).

DDT then waits for the user to enter one of the following characters:

1. Period (.) returns DDT to the command mode.

2. Carriage return causes one program instruction to be stepped. After the instruction is executed, the target registers will be printed again and DDT will wait for user input.

3. Line feed has the same effect as carriage return, but a heading to identify the registers will be printed out.

4. Space bar starts automatically single stepping. Single stepping will continue for 256 steps or until the space bar is pressed again. The user can thus start and stop single stepping of his target program. (See W-Step command).

<u>NOTE:</u> The contents of the registers reflect the effect of the last instruction before the breakpoint was encountered.

One breakpoint can be set at a time before execution is begun. a breakpoint can be reset by entering the B command with no operands. A breakpoint at a specific address can be cleared by executing that address.

There are certain characteristics of the DDT breakpoint facility which the user should be aware of during debugging.

1. The trap sequence used by DDT-80 is as follows:
   ```
   JP DDT      Jump to DDT Breakpoint Processor
   ```

2. Since DDT replaces three bytes of the user program, a breakpoint should be set such that when the user program is executed, control can only be transferred to the first byte of the trap sequence of an instruction. For example, in the following sequence:

   ```
   L1 JR NZ, L3-$

   L2 LD A,0

   L3 LD B,0F
   ```

   A breakpoint should not be set at L2 because when the branch condition at L1 is met, control would be transferred to the third byte of the trap sequence.

3. No error indication is given if one attempts to set a breakpoint in ROM.

4. After a breakpoint has been set, it can be changed simply by entering a new breakpoint. The act of entering a new breakpoint automatically clears the previous breakpoint.

5. When a breakpoint is encountered in a user program, DDT-80 saves the state of interrupts (through IFF) in the :IF register. The state of interrupts is restored or set according to the contents of :IF when control is transferred to the user program.

6. Breakpoint will not work in areas where executable code is modified by the program.

EXAMPLE

```
.B 24E (CR)
      -Set a breakpoint at location 24EH.

.O 100 (CR)
      -Set offset.

.B R4F3 (CR)
     -Set breakpoint at relative address 4F3H (=5F3H absolute).
```

A.12. C-COPY MEMORY BLOCKS COMMAND

FORMAT.

.C aaaa,bbbb,cccc (CR)	Copy locations aaaa through bbbb inclusive to the memory block starting at address cccc.

DESCRIPTION.

The user enters the command identifier C followed by the starting address aaaa and ending address bbbb of the block to be moved, followed by the starting address cccc of the block receiving the data. The operands may be absolute or relative and are separated by commas or blanks. Upon terminating with a carriage return, DDT performs the requested copy operation, and returns to the command mode. The copy command permits any block of memory data to be moved to any area of memory. The move may be forward or backward and the new block may or may not overlap with the original memory block. Entire programs or subroutines may be moved around in this way. Care should be taken to copy complete instructions on both ends of the block when copying programs, and any relative jump instructions contained within a block to be moved should not jump outside the block. If the second operand entered (bbbb) is smaller than the first (aaaa), a question mark (?) is printed and control returns to the command mode.

EXAMPLE.

.C 100,200,1200(CR)	Copy memory locations 100H through 200H inclusive to locations 1200H through 1300H.
.C 100,200,150(CR)	Copy locations, 100H through 200H inclusive to locations 150H through 250H. (overlapping copy)
.O 100(CR)	Set relative offset to 100H.
.C R0,R100,R50 (CR)	This would be the same as the previous example.

A.13. E-EXECUTE COMMAND

FORMAT.

.E aaaa (CR)	Transfer control to the program starting at address aaaa.
.E (CR)	Transfer control to the address specified by register:PC.

DESCRIPTION.

To cause execution of a program, the user types the identifier E followed by the desired entry address of his program. Upon typing carriage return, DDT loads the Z80 CPU registers and then transfers control to the program entry point. The contents of the register map reflect the effect of the last instruction before the breakpoint was encountered. If no entry address is specified after the E command, DDT will transfer control to the address specified by the :PC register (program counter).

EXAMPLE.

.E 1200(CR)	Execute the program starting at location 1200H.

To return control to DDT, the user's program must encounter a breakpoint (see B-Breakpoint Command).

```
.M :PC (CR)          Examine user's program counter
                     (PC).
.PC 62FF 1220(CR)    Set user's PC to 1220H
.E (CR)             1220H
```

The execute command may be used together with the breakpoint command to execute portions of programs while debugging.

A.14. F-FILL MEMORY COMMAND

FORMAT:

```
        .F aaaa,bbbb,cc (CR)  Fill memory locations aaaa  through
                               bbbb inclusive with cc.
```

DESCRIPTION.

The user enters the command identifier F followed by the starting address aaaa and ending address bbbb, followed by the data cc. The operands are separated by commas or blanks. Upon terminating with a carriage return, DDT performs the requested fill operation and then prints a "." to indicate that DDT is ready to accept another command.

Example

```
        .F 100,1FF,5A (CR)     Insert  a  5A  in  every  memory
                               location from 100H 1FFH.
        .O 100(CR)             Set offset to 100H.
        .F R0,RFF,5A(CR)     Fill   same  addresses   as   first
                               example.
        .                      DDT waiting for next command.
```

A.15. H-HEXADECIMAL ARITHMETIC

FORMAT.

```
.H +aaaa-bbbb+...+yyyy=zzzz(CR)  Perform hexadecimal
                                 arithmetic.
```

DESCRIPTION.

The user enters the command identifier and then enters the arithmetic expression. Only + and - are legal operations. If the sign of the first operand is omitted, it is assumed +. The equal sign causes the 4 digit (least significant 4 digits) result to be displayed. When the terminator is entered, DDT returns to accept another command.

EXAMPLES.

```
.H 5000-4FFF=0001(CR)       Subtract 4FFFH from 5000H.
.H 5000+4FFF=9FFF (CR)      Add 4FFFH to 5000H.

.                           The Equal sign caused the 4-
                            digit result to be printed. DDT
                            waiting for next command.
```

A.16. L-LOCATE DATA PATTERN COMMAND

FORMAT.

.L aaaa,bbbb,cccc(CR)	Locate and print the address of every occurrence of cccc from aaaa to and including bbbb.

DESCRIPTION.

The user enters the command identifier L followed by the starting address aaaa and ending address bbbb followed by the data cccc to be located. Upon terminating with a carriage return, DDT prints every address between aaaa and bbbb which contains cccc. If cccc is less than 100H, then a one-byte comparison is made. If cccc is greater than or equal to 100H, then a two-byte comparison is made. The data to be located should be entered with the most significant two digits of data followed by the least significant two digits of data (if location 1000H contained 13 and location 1001H contained 92, the user would enter 9213 as the data to locate).

EXAMPLE:

.L, 0,750,35(CR)	Locate every occurrence of 35H between address 0 and 750H.
0052 35	Every location between 0 and 750H containing 35H is printed.
00F3 35	
0542 35	
0750 35	
.L 0,750,35FF (CR)	Locate every occurrence of the 2-byte value FF35H between 0 and 750H. Note that the value is low-high.
00F3 35	Every address where FF35 is found is printed out.
0145 35	The location previous to the location printed out contains the least significant two digits.

A.17. M-DISPLAY AND UPDATE MEMORY OR REGISTER COMMAND

FORMAT:

.M aaaa(CR)

DESCRIPTION.

The user enters the command identifier M and the operand aaaa followed by a carriage return. DDT prints the memory address or mnemonic on the next line, followed by the contents of that particular address in hexadecimal. If the contents are to be changed, the new value is entered. Any number of digits may be entered, but only the least significant two (or four) digits are accepted.

Terminators. When the user is examining and/or modifying a register or memory location, the accompanying terminator signals the action DDT is to take. The possible operand (new value entered) and terminator combinations are:

Terminator	Meanings
(CR)	No operand entered, display next address or register.
^	No operand entered, display previous address or register.
/	No operand entered, exit to command mode.
aa.	Operand aa entered but "." aborts command with no change to value at address.
aa^	Operand aa entered, change value at address to aa and display same address with the new value aa displayed.
aa/	Operand entered, change value at address to aa then exit to command mode.

Memory display. Memory locations are accessed as follows:

M 16A(CR)	Examine memory location 016AH.
016A 3F(CR)	It contains 3FH. Do not change, step to next location.
016A 3F 34FF^	Change contents of 016A to FFH and display same location. Note that only the last 2 digits typed are stored in 016A (the entry 34 was in error).
016A FF (CR)	New contents displayed, step to next.
016B 92 .	
.	DDT waiting for next command.

When accessing relative memory locations, the user sets the offset with the "O" command and uses the "R" prefix with the memory address. Assuming the offset was set to 1000:

.M RO (CR)	
'0000 1000 xx.	The relative address, absolute address and data re-printed out.
.	DDT waiting for next command.

Register display. The user may examine and change his CPU registers. They may either be initialized prior to program execution or after a breakpoint has been encountered in the program to be debugged. The contents of the user's registers may be accessed through the use of the mnemonics discussed previously.

.M :A(CR)	Examine user's accumulator.
:A 18 25(CR)	Change register A to 25H, examine next location.
:PC 0010 .	User's PC Register, return to command mode.
.M :PC(CR)	Examine user's PC (program counter) register.
:PC 0010 .	Return to command mode.
.	DDT waiting for next command.

When resuming execution of the user's program, these new values will be inserted into the user's Z80 CPU registers.

Relative branches. A special feature of DDT allows the user to conveniently compute relative addresses used in relative branch instructions. The value of the symbol "$" is defined as the value of the current location and only has meaning during display and update commands.

This example shows the entering of a jump relative instruction at location 0H to branch to location 38H.

```
.M 0(CR)              Examine location 0H.

.0000 20 18(CR)       Insert first byte of jump (JR 38H-$)

.0001 F8  38-$=0036^  Compute and display relative dis-
                      placement for branch from 0H to 38H.

.0001 36 .            Branch displacement of 36 shown.

.                     DDT waiting for next command.
```

It should be noted that the maximum allowed displacement value for forward branches is 7FH and for backward is 80H. It is simple to determine if the relative branch is within its range by examining the most significant two digits of the computed displacement. For forward branches, the most significant two digits should be 00H and for backward branches, the most significant two digits should be FFH.

A.18. M-TABULATE MEMORY COMMAND

FORMAT

```
    .M aaaa,bbbb(CR)   Display  memory location  aaaa  through
                        bbbb.
```

DESCRIPTION.

The user enters the command identifier M followed by the starting (aaaa) and ending (bbbb) address of the memory block. Upon terminating with a carriage return, DDT prints a line feed, and then prints the contents of aaaaH to bbbbH inclusive, with up to 16 values per line. DDT then returns to the command mode. The tabulation may be stopped at any time by entering "." on the console. When the 'R' prefix is used, the relative address is printed before absolute.

EXAMPLE:

```
 .M 4100,4127(CR)         Display    memory    locations    4100H
                          through 4127H inclusive.


4100 2B 90 12 20   00 B7 A5 21   10 94 04 20   CA B7 44 18
4110 81 11 34 21   07 94 17 45   12 55 A5 18   21 80 C5 55
4120 90 0C A5 81   09 21 40 22

 :O 4100(CR)        set offset to 4100
 .M R0,R27(CR)

'0000 4100  2B 90 12 20  00 B7 A5 21  10 94 04 20  CA B7 44 18
'0010 4110  81 11 34 21  07 94 17 45  12 55 A5 18  21 80 C5 55
'0020 4120  90 0C A5 81  09 21 40 22
```

A.19. O-SET OFFSET CONSTANT COMMAND

FORMAT:

```
        .O aaaa(CR)          Set offset constant to aaaa.
```

DESCRIPTION.

The user enters the command identifier O followed by the offset aaaa. Upon terminating with a carriage return, DDT saves the 16-bit offset. After the offset has been set, both relative and absolute addresses are printed anytime addresses are displayed and until the offset is cleared. The offset can be cleared by entering the O command with no operands.

EXAMPLE:

```
      .O 200(CR)        Set offset
      .H RO=0200(CR)    Display value of offset
      .                 DDT waiting for next command.
```

A.20. P-DISPLAY AND UPDATE PORTS COMMAND

FORMAT.

```
.P aa (CR)
```

DESCRIPTION.

The user enters the command identifier P followed by the port address aa and a carriage return. DDT responds by printing the port address and the value at that port. If the value at that port is to be changed, the user enters the new value. The new value entered is a two-digit hexadecimal operand. When the user is examining and/or modifying a port, the terminator signals the action DDT is to take. The possible operand (new value entered) and terminator combinations are:

Terminator.	Meaning
(CR)	No operand entered, display next port.
^	No operand entered, display previous port.
.	No operand entered, return to command mode.
aa.	Operand aa entered, but "." aborts command with no change to the port.
aa(CR)	Operand aa entered, change the port value to aa and step to display the value at the next port.

EXAMPLE:

.P E2(CR)	User displays port E2H.
E2 00 12(CR)	User changes value to 12H.
E3 15 .	Return to command mode.
.	DDT waiting for next command.

A.21. Q-QUIT COMMAND

FORMAT

```
        .Q (CR)
```

DESCRIPTION:

This command does not function properly in the M/OS-80 environment. It should not be used. It is possible, however, to jump back into M/OS-80 by entering the following command if the RAM image of the operating system has not been contaminated.

```
        .E 0 (CR)
```

A.22. R-DISPLAY CPU REGISTERS COMMAND

FORMAT

.R (CR)	Print the contents of the CPU registers.
.R 1(CR)	Print a heading to label the CPU registers on one line. On the next line, print the contents of the CPU registers.
.R 1,a(CR)	Print a heading to label the CPU registers and set the long/short flag. LONG causes all registers to be printed after breakpoint and single step. SHORT causes only PC and AF to be printed. The LONG/SHORT FLAG remains set until changed by the 'R' command.

DESCRIPTION.

The user enters the command identifier R. If the user wants a heading to be printed that labels the register contents, an operand of 1 is entered. If no heading is desired, then no operand is entered. If the '0' command has been used to set an offset, the relative PC is also printed (PC'). The second operand is optional and has the following meaning:

a=0 - Short form. Only the Z80 program counter and AF register will be displayed.

a=1 - Long form. All CPU registers will be displayed.

Note that "a" remains set to the value entered during all following commands until it is reset.

Example:

```
     .R(CR)
A000 0100 0104 CFB3 C094 FFEE EDF6 9C3E C3DC FE9B D6EB F1BE FFB4


     .R 1(CR)
PC   AF   IIF  BC   DE   HL   A'F' B'C  D'E  H'L'  IX   IY   SP
A000 0181 0104 CFB3 0010 C09A FFEE EDF6 C3DC FE9B D6EC F1BE FFB4
```

```
    PC contains A000H
     A contains 01H
     F contains 81H


               Bit
    7  6  5  4  3  2  1  0
F = 1  0  0  0  0  0  0  1
    S  Z  X  H  X P/V N  C

             S = sign flag

             Z = zero flag

             X = indeterminate flag

             H = half carry (for BCD operations)

           P/V = parity or overflow flag

           N   = BCD add//subtract flag

           C   = carry flag

 I contains 01H
IF contains 04 (Bit 3 = 1 implies IFF = 1)
IY contains F1BEH
SP contains FFB4H
```

A.23. V-VERIFY MEMORY COMMAND

FORMAT

.V aaaa,bbbb,cccc(CR)	Compare memory location aaaa to bbbb with the memory starting at cccc.

DESCRIPTION.

The user enters command identifier V followed by the starting address aaaa and ending address bbbb, followed by the starting address cccc of the second memory block. The operands are separated by commas or blanks. Upon terminating with a carriage return, every address from aaaa to bbbb is compared with the corresponding address starting at cccc. Any discrepancies are printed on the console ("address data address data"). When the comparison is complete, DDT is ready to accept another command. Printing of addresses may be aborted by entering a period (.) from the user console at any time.

Example:

.V 0,FF,1000(CR)	Compare every location from 0 to FFH inclusive with locations beginning at 1000H.
.O 100(CR)	Set offset.
.V R0,RFF,R1000(CR)	Compare relative address.
'0000 0100 BC '1000 1100 CC	Relative and absolute address is printed for non-matched locations.

A.24. W WALK THROUGH A PROGRAM COMMAND

The walk command, also known as software single-step, allows stepping through a program which is contained in RAM. The user's registers are saved and displayed after each step.

FORMAT.

.W aaaa,nn,xxx(CR)	Begin software single-step at address aaaa, for nnH steps, xxx = HD requests register heading, xxx = DIS requests disassembly (AIM-Z80B required for DIS).
.W Raaaa,nn,xxx(CR)	Relative address.

DESCRIPTION.

The user enters the command identifier W followed by the starting address aaaa, the number of steps to take nn, and the options operand xxx. The operands are separated by commas or spaces. Upon terminating with a carriage return, DDT begins "walking" through the user's program (RAM-resident). After each step, the user's register are displayed (See 'R' command). When nn steps have been taken, DDT waits for the user to enter a carriage return, line feed, space, or ".". A carriage return causes the next instruction to be executed and wait again for input. A line feed causes the register heading to be printed before executing the next instruction. A space causes single stepping to continue for 256 instructions or until another space is entered to stop stepping. If nn is omitted, the default is 1. If aaaa is omitted, the last value of the user's program counter (:PC) is used to begin "walking". The stepping may always be stopped by entering any of the characters described above. When the address entered is relative, the 'PC is also printed (relative PC).

Restrictions to W command.

1. Only operates with programs in RAM.
2. Cannot CALL OR RESTART to an address one or two locations before the CALL or RESTART.
3. Walking through self-modifying code is not allowed.

B. SYSTEM SETUP

The following sections describe how to make use of your M/OS-80 system diskette. The currently-distributed version of M/OS-80 is provided on a single diskette which will boot up only on those MOSTEK computers which have the correct PROMs installed. The part number for the current version is shown below.

MK77984-4 - (One Diskette, 4 EPROMS)

Note: EPROMS provided are designed for MD systems only.

B.1. SYSTEM REQUIREMENTS

In order to use the M/OS-80 operating system for your particular hardware system, you may have to make some changes to your system. M/OS-80 is designed to run on MOSTEK's MATRIX Series systems (SDB-80) and on systems built with MD series boards using the MDX-CPU1 and MDX-CPU2. All of these systems use the standard Designer's Development Tool (DDT) and Disk Controller Firmware (DCF) EPROMs.

B.1.1. Minimum Hardware Required

Processor:	MDX-CPU1 or MDX-CPU2 or SDB-80
Console Interface:	MDX-EPROM/UART or MDX-SIO(A)
Printer Interface:	MDX-PIO or MDX-SIO(B)
Floppy Interface:	MDX-FLP1 or MDX-FLP2 (Single density only)
Memory:	64K MDX-DRAM or RAM80B or equivalent

Note that all M/OS-80 configurations now require 64K of system RAM. To generate M/OS-80 systems using less then 64K RAM or to use other than the standard peripheral drivers, users must buy the new MOSGEN system generation package which is sold separately.

MK77984-4 is configured to support **single**-density floppy disk operations only. If the MDX-FLP2 is used, and other then dual-sided/single-density operation is required, then one of the two alternate configurations must be activated. See: Alternate Configuration Section.

B.2. EPROMS PROVIDED

M/OS-80 (MK77984-4) is provided with four EPROMs which contain the appropriate firmware to run the various acceptable configurations of M/OS-80. In systems currently running FLP-80DOS or an older version of M/OS-80, these EPROMS are not needed but should be retained to provide upgrading to MDX-FLP2 if desired at a later time. EPROMs shipped with the MDX-FLP boards cannot be used in the M/OS-80 configuration. Shown below are the part numbers for the suppied firmware.

MK6293 - Designer's Development Tool (DDT) for UART console systems.

MK6235 - Designer's Development Tool (DDT) for SIO console systems.

The DDT firmware must be located at E000 Hex in the system address space.

MK6286 - Disk Controller Firmware (DCF) for MDX-FLP1.

MK6340 - Disk Controller Firmware (DCF) for MDX-FLP2.

The DCF firmware must be located at E800 Hex in the system address space.

Choose the appropriate EPROM part number based on the type of Console and MDX-FLP card that is to be used.

As a general rule, pre-configured systems sold by MOSTEK use UART Consoles and PIO List devices. These systems include the AID-80F (discontinued), MATRIX-80/SDT, and MATRIX-80/SDS. All of these systems are sold with or cna be expanded to use 64K of system RAM as required by M/OS-80. In these systems the EPROMs provided with the system will provide all needed firmware for proper operation of M/OS-80 provided that the MDX-FLP2 board is not used. To use the MDX-FLP2, the existing MK6296 DCF EPROM must be replaced with MK6340 as described above. This change will permit single-sided/single-density through dual-sided/single-density operation with the standard M/OS-80. Dual density operation with the MDX-FLP2 requires use of an alternate configuration. See Alternate Configuration Section.

The function of the supplied firmware is to provide the user with a comprehensive diagnostic/debugging tool and to boot M/OS-80. All EPROMs provided (including the MDX-FLP2 EPROMs) expect that the system diskette to be formatted as **single density.** It is **not** possible to boot MK77984-4 from a dual-density diskette. It **is** possible to use dual-density diskettes in systems using the MDX-FLP2 provided that the system is not required to boot from the dual density diskette.

B.3. MEMORY CONFIGURATION

In systems which are not currently running at the 64K RAM level, the following paragraphs outline the need and considerations that need to be made in the configuration of the system memory map. For users that already have running 64K systems, these paragraphs may be skipped.

B.3.1. Scratch-Pad Conflicts

When configuring the RAM memory map for M/OS-80, it is vital that no conflict exist between the CPU-resident scratch-pad RAM (FF00-FFFF Hex) and the other memory cards. In most cases the bipolar PROM on the DRAM-32 card is used to map out the proper area which leaves room for the firmware at E000-EFFF Hex and for the DRAM-resident scratch-pad at F000-FFFF Hex. Improper strapping of the DRAM or failing to de-activate the RAM on the CPU card will result in erratic operation of M/OS-80.

B.3.2. Further RAM Mapping Information

In case of difficulty, consult the appropriate MDX-RAM and MDX-CPU operations manuals.

B.4. CONSOLE AND PRINTER CONFIGURATION

The newly-created version of M/OS-80 (MK77984-4) will automatically recognize either the standardly-addressed UART or SIO serial port attached to the system console device. Once the proper EPROMs are installed, the system will do the rest. Note that this automatic feature will only work if the SIO or UART port is correctly configured.

No provision is made for automatic List device recognition so an alternate configuration has been provided to permit use of an SIO-driven instead of the standard PIO-driven List device. For further details of alternate printer selection see the Alternate Configuration section.

Console and Printer Configurations:

```
Address/Control port strapping:
-----------------------------------
MDX-PIO  D0-D3 Hex  (J4 Pin 5-6, 9-10)

MDX-UART DC and DD Hex (J4 Pin 5-6)

MDX-SIO  (J9 Pin 5-6)
          Channel A:(Console) DC and DD Hex (CON:)
          Channel B:(Printer) DE and DF Hex (LST:)

MDX-SIO2 (J9 Pin 5-6)
          Channel A:(Console) DC and DD Hex (CON:)
          Channel B:(Printer) DE and DF Hex (LST:)

Functional Strapping:
-----------------------------------
MDX-PIO  Direction: J3 Pin 1-2, 3-4, 5-6, 19-20, 21-22
     Centronics Printer requires: U2 74LS242

MDX-EPROM/UART :Set baud rate with switch U8
     Typical settings - Position:1 2 3 4    X=Closed, -=Open
     --------------------------|-------|------------------
                  19200 baud    - - - -
                   9600 baud    X - - -
                   1200 baud    - - - X
                    300 baud    - X - X
```

```
MDX-SIO1:
     RS232 (DCE):
          Channel A:J3 Pin 1-2, 5-6, 9-10, 13-14, 17-18, 21-22
          Channel B:J4 Pin 1-2, 5-6, 9-10, 13-14, 17-18, 21-22

     Baud Rate :
          Channel A: J8 Pin 15-16 (9600 Baud) (no Jumpers = 19200)
          Channel B: J8 Pin 7-8   (9600 baud)

     XMIT and REC frequency sources:
          Channel A: J7 Pin 1-2, 5-6
          Channel B: J7 Pin 11-12


MDX-SIO2:
     RS232 (DCE):
          Channel A:J3 Pin 1-2,11-18,12-17,13-20,14-21,15-22,16-19
          Channel B:J4 Pin 1-2,11-18,12-21,13-20,15-22,16-19
```

NOTE:Strapping shown for channel B is for the DIABLO 630. The strapping for other printers may be different. The DIABLO was also configured with block "A60" jumpers 1-2, 5-6 which enables the RTS/CTS protocol.

```
     Baud Rate :
          Channel A: J8 Pin 7-8 (9600 Baud) (no Jumpers = 19200)
          Channel B: J8 Pin 15-16 (9600 Baud)
                     J8 Pin 9-10 (1200 Baud)

     XMIT and REC frequency sources:
          Channel A: J7 Pin 1-2, 3-4
          Channel B: J7 Pin 5-6, 7-8
```

Various other strapping options can be determined by consulting the appropriate documentation manuals. The options specified above, however, are known to function with M/OS-80 with the appropriate (SIO or UART) EPROMs.

Be sure to include active operation of the DTR and RTS modem control lines for all SIO Console and Printer installations.

B.5. ALTERNATE CONFIGURATIONS

In order to permit a wider range of hardware combinations, M/OS-80 (MK77984-4) is supplied with four alternate configurations. These are supplied for those users who require dual-density operation with the MDX-FLP2 disk controller and/or MDX-SIO List operation. To select the correct combination of hardware and features, consult the table shown below.

MK77984-4
ALTERNATE CONFIGUATION

Disk Interfaces Density/Sides	Printer Interfaces MDX-PIO	MDX-SIO
MDX-FLP1 Single/Single	Standard	MOS-AS64.COM
MDX-FLP2 Single/Dual	Standard	MOS-AS64.COM
MDX-FLP2 Dual/Dual	MOS-DAP6.COM	MOS-DAS6.COM

Choose the appropriate alternate system name from the table. The method to active that file as the current M/OS-80 system is described below in the system setup procedure. If the word "standard" is shown as the desired combination of features, then the standardly-configured M/OS-80 will suit your needs.

In the following steps you will backup the standard systems disk and, if necessary will replace (by renaming) the current SYSTEM.COM file (which is the stardard M/OS-80 system brought in by the boot operation) with the alternate system of your choice.

B.6. ALTERNATE SYSTEM CONFIGURATION

If you have a functioning 64K MATRIX-80/SDT, 64K MATRIX-80/SDS (SDE-based) or a 64K AID-80F, an alternate configuration is not needed.

```
|          "EPROMs installed, System Functional"---->
|
↓
System Identification - Begin:
|
↓
MDX-FLP1 Systems: Use MK6286 as the system DCF EPROM.
MDX-FLP2 Systems: Use MK6340 as the system DCF EPROM.
|
↓


↓
MDX-UART Console Systems: Go to: UART Systems Begin ------>
|
↓
MDX-SIO Systems: (Begin)
|
↓
```

Find the EPROMs shipped with your M/OS-80 package numbered MK6325 and the system DCF EPROM as chosen above.

|
↓

Mount these two EPROMs in appropriate sockets addressed as described below. The SIO DDT EPROM MK6325 is to be addressed at E000 hex and the DCF EPROM is to be addressed at E800 hex. These EPROMS can either be mounted on one of the CPU boards, the UMC card, the EPROM/UART card, or some other appropriate ROM/EPROM card. See hardware strapping notes for details of typical configurations.

|
↓

Perform any other system preparations as necessary to get system into a functional state. See hardware strapping notes for additional details. Also consult various operations manuals as required. Ensure that the console baud rate matches the baud rate set on the SIO card.

|
↓

Power up the system. Place the MK77984-4 system diskette selected above into drive "A" which is the drive strapped as device 0. To ensure documentation compatibility, ensure that this is the right-most drive in a two-drive system (when viewed from the front).

|
↓

Type (cr) on the console connected to SIO port A.

|
↓

Continue with:SIO Configuration Setup (next page)

```
|  SIO Configuration Setup  (cont.)
↓
M/OS-80 Should now begin to boot up. This process takes several
seconds (more than 10).
|
↓
The signon message should appear on the screen. If this message
is garbled, check baud rate, parity settings, and bit-counts.
|
↓-> Problems?  ---------> Go to: In Case of Difficulty Section.
|
↓
You have now booted the 64K, SIO Console/PIO Printer version  of
M/OS-80 Version 3.0.
|
\
 \
  -------------> Go to: Custom System Selection Section
```

MDX-UART Systems Begin:

|
↓

Find the MK6293 EPROM shipped with your M/OS-80 package and the system DCF EPROM selected above (based on the MDX-FLP card in use). Mount these two ROMs in appropriate sockets in place of the DCF ROM presently on the system. These replacement ROMS can be moun- ted either on one of the CPU boards, the UMC card, the EPROM/UART card, or some other appropriate ROM/EPROM card. Make sure to strap these EPROMs/ROMs at location E000 hex. See hard- ware strapping notes for details of typical configurations.

|
↓

Perform any other system preparations as necessary to get system into a functional state. See hardware strapping notes for additional details. Also consult various operations manuals as required. Ensure that the console baud rate matches the baud rate set on the EPROM/UART card. This step should not be necessary for a already-functioning system.

|
↓

Power up the system. Place the MK77984-4 M/OS-80 diskette into drive "A" which is the drive strapped as device 0. To ensure documentation compatibility, ensure that this is the right-most drive in a two-drive system.

|
↓

Type (cr) on the console connected to the MDX-UART.

|
↓

M/OS-80 Should now begin to boot up. This process takes several seconds (more than 10).

|

The signon message should appear on the screen. If this message is garbled, check baud rate, parity settings, and bit-counts.

|

You have now booted the 64K, UART Console/PIO Printer version of M/OS-80.

\
 --------> Go to: Custom System Selection Section. (next page)

<u>Custom System Selection Section:</u>

|
↓

You should have already booted a 64K/PIO Printer version of M/OS-80 for your particular console. If you have not, go back to the beginning. The prompt on the console should be:

|
↓

```
Mostek M/OS-80 Version 02.3n-nn.nn
Configuration number nnnnnn
Serial Number nnnnnn
A.
```

|
↓

The cursor should be just after the period. The signon message should be just above the "A." as shown.

|
↓

In the next steps you will backup your MK77984-4 diskette. If necessary, you will then select an alternate configuration to better match your particular hardware needs.

|
↓

Backup System Diskette
Perform the following step first to prevent loss of data. This is somewhat tedious in a single drive system, but must be done to ensure no loss of data.

```
|
v
|\
|  Dual  Drive  Systems:
|        v
v        v
|        Type: COPY /V B:=A: (cr)
|        |
|        v
|        When asked to mount Source Disk on drive A, type:(cr).
|              The source disk in this case is the already moun-
|              ted M/OS-80 system disk.
|        When  asked to mount Destination Disk, mount a blank,
|              formatted diskette in Drive B. This is to become
|              the backup copy of M/OS-80.
|        |
|        v
|        System should now copy then entire diskette on drive A
|        (right drive) to the diskette on drive B.
|        |
|        v
|        Proceed to RENAME step.------------------------->
|
 \
   Single Drive Systems:
         |
         v
         Type: COPY /V A:=A: (cr)
         |
         v
         When asked to mount Source Disk on drive A, type (cr).
               This is the MK77984-4 M/OS-80 system disk.

         When  asked to mount Destination Disk mount a blank disk
               as discussed above, type (cr). You will have to
               remove the system disk mounted in drive A to do
               this.
         |
         v
         System should now copy the entire diskette on drive A
         (right drive) to the blank diskette alternately mounted
         on drive A. You will be asked to remove and replace both
         disks several times.
         |
         v
         Note: Be very careful when changing disks.
         |
         v
         Proceed to RENAME step.
```

RENAMEStep:
This step is only necessary if you need to select one of the alternate configurations of M/OS-80 provided with MK77984-4. These alternate configurations, as discussed previously, will permit the user to make use of an SIO printer or dual-density floppy disk operation or both. If your system uses a PIO printer and the MDX-FLP1 or if you are satisfied with single-density/dual-sided operation of the MDX-FLP2 then you are finished.

|
↓

Type: A.ATRIB A:SYSTEM.COM 0(cr)

|
|

The replacement name used in the next step is the name chosen from the alternate system selection table shown above.

|
↓

Type: A.XFER/V A:SYSTEM.COM=A:<new system name>(cr)

|
↓
|
↓

At this point, you should now have a diskette which will boot up with the desired Console, Printer, and size.

|
↓

To boot this system, mount this diskette in drive A (it should already be there), hit RESET on the front panel, and type (cr).

|
↓

Backup this disk using COPY /V as discussed previously. See M/OS-80 Operations Manual for details of this backup operation.

|
↓

End.

In Case of Difficulty

Consult the M/OS-80 manual for additional details of various system-oriented problems that may be encountered. The following suggestions may help you decide what areas are causing you difficulty.

If the system does not boot at all:

> Check system functionality: disks, terminals, boards etc. Be sure of the strapping, memory configurations, board and EPROM addressing.
>
> Will the system enter DDT monitor? If it will, boot problems must involve the disk controller, disk drives, cabling, strapping etc. If it will not, suspect a more CPU-oriented problem in the jump-on-reset strapping, EPROMs in use and their addressing, or the RAM area strapping and configuration.

If the system tries to boot but cannot.

> Consult M/OS-80 manual for suggestions.

NOTE: M/OS-80 uses the modem control lines for both the console and SIO print devices. Be sure that DTR and RTS are exercised by the device to ensure proper operation. If the device does not support this protocol you must ensure that these lines are held high (active) so that M/OS-80 will recognize them as ready.

SUBJECT INDEX

SUBJECT INDEX

UNITED TECHNOLOGIES MOSTEK

1215 W. Crosby Rd. • Carrollton, Texas 75006 • 214/323-6000
In Europe, Contact: MOSTEK Brussels
270-272 Avenue de Tervuren (BTE21), B-1150 Brussels, Belgium;
Telephone: 762.18.80

PRINTED IN USA February 1982
Publication No. 4420064