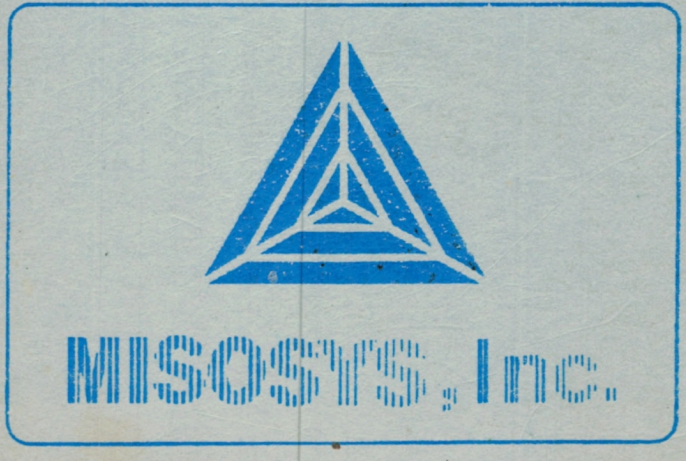
LDOS™ & LS-DOS™

Reference Manual

Cat. No. M-40-060



LDOS/LSDOS

Reference Manual

An 8-bit Operating System for

Z80 based Computers

MISOSYS, Inc.
P. O. Box 239
Sterling, VA 22170
703-450-4181

Introduction to DOS

These first few pages will introduce you to the basic principles of a disk operating system, and list some of the DOS commands and utilities you will be using in your day to day activities. You can use this as a convenient reference until you become familiar with operating your computer under the DOS system.

There may be a file named "README/TXT" on the DOS disk (Disk 2 for Model I). If so, this file will contain important information which may not appear in this printed documentation. You should read this file by issuing the command:

```
LIST README
```

Nomenclature used in this Reference Manual

Throughout this reference manual, illustrations of communications with the operating system are presented in various forms. These are:

EXAMPLE	font used to depict keyboard entries typed by you
Message	font used to indicate a message displayed by the DOS
`screen`	font used in presentation of display screens
[optional]	square brackets surround optional keyboard entries
on|off	a vertical bar is used to indicate either of two permissible entries, only one of which may be entered at a time.

Using this manual

The DOS User's manual is set up to be easily used. It is divided into several different sections. The first section is composed of general information about the DOS system. It contains the introduction to DOS, as well as descriptions of the commands and files available.

The second section contains reference material on all the DOS library and utility commands. These commands are the heart of the operating system, and provide the link between the user and the computer. They will be

listed in alphabetical order.

Another section contains device DRIVER and FILTER programs for some of the devices available under DOS. Programs include provisions for such features as keyboard type ahead and formatted line printer output.

The next section contains detailed operating instructions and information on the DOS Job Control Language (JCL); this is a batch processing facility. PAGE 279

Finally, the last section contains a GLOSSARY which describes significant terms used in this reference manual, and an extensive INDEX.

To locate the section of the manual you wish to access, refer to the *Table of Contents* or the *Index*. All commands or programs in each section will be in alphabetical order. Any time you encounter an unfamiliar word or definition, refer to either the *Glossary* or the information in the first section of this reference manual.

What is a Disk Operating System (DOS)?

Without a DOS, your computer is controlled by routines stored in ROM (read only memory). The ROM handles all I/O (input/output) to your system devices - the keyboard, cassette, video, printer, etc. It does this from within a BASIC language environment. A DOS takes over control of the I/O processing, and adds its own routines to handle the disk drives. Instead of starting out in BASIC, you will be in a new environment known as the "DOS Level". Now, any commands you give are interpreted by the DOS, rather than by the BASIC stored in the ROM. In some environments, such as DOS version 6, the ROM is disabled after booting and the configuration is all RAM.

What does this mean to you, the user? First of all, every DOS has a given set of commands, different than the ones available from the BASIC level. Commands may be entered when the "DOS Ready" prompt is on the screen. The commands are not programs that you would use to perform "application" type functions such as accounting or word processing. Rather, they are the means to load and invoke applications programs, and then maintain those programs and the data created by those programs. DOS commands also let you modify the way certain devices work, by providing things such as type ahead for the keyboard, a user-definable

blinking cursor, and an output format program for a line printer.

As you can see from the table of contents, there are many program files provided on your DOS disk. Before explaining their function, you should know how information, including the DOS programs, is stored on a disk.

Disk organization

Your disk drives store data in the form of magnetic pulses on the diskette media. In order for a disk operating system to access this data, the diskette obviously must be organized in some manner. The DOS FORMAT utility program will write all necessary information to a diskette to organize it into cylinders, tracks and sectors so it may be accessed in a structured manner. Only after a diskette is formatted may it be written to or read from by DOS.

The structure of a diskette surface may be thought of as a series of concentric circles, starting at the outer edge of the diskette media and moving in towards the center. These circles of information are referred to as tracks, starting with track number 0 at the outer edge. The number of tracks available on a diskette will depend on the type of disk drive you have purchased.

The term "sector" refers to a space on a track that can hold 256 characters of data. You will find the term "byte" is commonly used to refer to one character, and thus a sector is referred to as 256 bytes. The number of sectors available per track is dependent on the size (3.5", 5.25", 8", or hard) of the disk, and on the density specified when formatting the disk. With DOS, a double density floppy diskette will have approximately 80% more space available compared to a single density diskette with the same number of tracks. Four sectors, or 1024 bytes, will be referred to as "1K" (standing for about one thousand bytes or characters).

The minimum allocation of disk space made when storing a file on a diskette is called a *granule* or *gran*. The number of sectors that make up a gran will vary depending on the size and density of the diskette and the type of drive. This basic format structure is used by DOS for any type of drive it supports, 3.5", 5.25", 8", or hard disk. Diskettes formatted by other operating systems may not be directly readable by DOS.

To keep track of the various file names on the disk, and to record the tracks and sectors where a file is stored, DOS uses an area of the disk

called the *directory*. This directory is usually placed on the center cylinder of the disk during the formatting process.

FILES - How information is stored

The most important thing to remember about disk storage is the term *file*. Any information, be it a program or data, is stored as a file. This means that the information is written to the data areas of the disk, and the name of the file and the actual location of the data are stored in the directory. No matter what kind of file you are dealing with, the file name must use the following format:

FILENAME/EXT.PASSWORD:D

FILENAME - Up to eight alphabetic or numeric characters, the first of which must be alphabetic. All files must have a filename.

/EXT - An optional field called the file extension. If used, it can contain up to three alphabetic or numeric characters, the first of which must be alphabetic. An extension can be of great use to identify and deal with certain types of files, and it is strongly recommended that all files you create be given extensions.

.PASSWORD - This is an optional field that will assign a protection status to a file. If used, it can be up to eight alphabetic or numeric characters, the first of which must be alphabetic.

:D - This is an optional field called the drive specification. It is used to specify the particular drive number the file is on. It can be any number 0 through 7, depending on the number of drives in your system.

These four parts comprise the complete file specification, which will be referred to as a "*filespec*".

DOS organization and files

DOS is organized into different groups of files. The first group is the "SYSTEM" files, containing the necessary information to control the disk drives and your other devices. There are special system files known as the "LIBRARY". They contain the programs most commonly used to manipulate your files and devices. Another group is known as the "UTILITY" files. These provide added means of handling your program

and data files. Both the library commands and utility commands will be listed alphabetically in this manual for your convenience. The file group containing "DRIVER" and "FILTER" files gives added flexibility to your system devices. There is a *LANGUAGES* section containing "Job Control Language", which is a very powerful batch processing language. Usually provided on your DOS disk is BASIC, which is an extension of the BASIC language provided in the ROM [Model I/III] or complete disk BASIC [Model 4]. Documentation for BASIC is not provided in this reference manual; a separate reference manual for BASIC is available.

There are two types of disks that can be used with DOS. The first type is called a "SYSTEM DISK". This is a disk that contains the DOS system files.

A system as complex and flexible as DOS would occupy considerable memory space to be able to provide all of its features. DOS, however, makes extensive use of overlays in order to minimize the amount of memory reserved for system use. An overlay is a module that loads into memory, overlaying anything which was loaded there previously. In this manner, many functions can occupy the same area of memory, being loaded and used only when specifically needed. The compromise in using an overlay driven system is that while a user's application is in progress, certain disk file activities requested of the system may require the operating system to load different overlays to satisfy the request. This could cause the system to run slightly slower than a less sophisticated system which has more of its file access routines always resident in memory. The use of overlays also requires that a SYSTEM diskette be available in drive :0 - the system drive.

The second type of disk will be called a DATA DISK. This is a disk that has been formatted, but contains no DOS system files. This would be the type of disk you would normally use in a drive other than drive :0.

No matter which type of disk you are using, the formatting process will put two files on the disk; BOOT/SYS and DIR/SYS. These files contain information about the type of disk and the disk directory, and are normally invisible to the user. Under no circumstances should you ever copy these files from one disk to another, or attempt to delete them. Doing so can render the disks involved totally useless! DOS automatically will take care of updating any information in these two files.

DOS and Devices

Devices are generally thought of as a physical piece of your computer hardware; the video display, keyboard, printer, etc. The routines that control the I/O for these devices can be the ones provided in the ROM, if applicable, or can be ones provided by DOS. In either case, there is a small section of memory set aside as a control block for each device. With DOS, you will have a certain amount of "device independence". Device independence will allow you to treat each of your devices individually.

As with files, DOS uses a definite specification when accessing devices, called a "*device specification*", or "*devspec*". A devspec is very easy to understand. It consists of an asterisk followed by two alphabetic characters. For example, your keyboard devspec is ***KI** (Keyboard Input), the video is ***DO** (Display Output), and the printer is ***PR** (PRinter).

There are programs provided that will modify the I/O routines for certain devices. The DRIVER and FILTER section of this manual explains the functioning of these routines. You can read that section and determine if you need the extra features provided by those programs.

Using the DOS files

Now that you know what file groups are on an DOS disk, let's discuss some of the more important DOS commands and how to use them. The following descriptions will be general in nature, more to give you an idea of which commands and utilities do what than to explain them in detail. You should refer to the proper section of the manual for in-depth explanations.

Viewing filespecs and files

To see the files in a disk directory, you should use the **DIR** Library command. This will show you any DOS files on a disk, as well as any program or data files you have created. The **LIST** command will allow you to inspect the contents of any individual file, sending the display to either the video screen or the printer.

Moving files

Files may be moved individually from one disk to another with the **COPY** command. The **BACKUP** command lets you automatically move any or all

files from one disk to another. Of course, the disk to receive the files must have been previously formatted.

Removing unwanted files

Any file may be removed from a disk with the DOS version 5 **KILL** or DOS version 6 **REMOVE** command. Either will remove the information from the directory, and free up the data storage space previously assigned to that file.

Changing file names

The **RENAME** command will let you change the filename or extension of any file in the directory. The **ATTRIB** command will let you apply or change a file's password. Also, a file specification may be changed during the COPY process.

Viewing devices and disk drive parameters

The **DEVICE** command will let you see what devices you have active in your system. You will also see the information that DOS has stored in memory about the number of disk drives and the types of disks you are using. Certain disk drive information may be changed with the **SYSTEM** command.

Establishing or Removing devices

The **SET** and **ROUTE** commands will let you establish devices. The **LINK** command will also let you link multiple devices together. The **RESET** and **KILL** or **REMOVE** commands can be used to remove unwanted devices.

Making a BACKUP

Now that you have read the introduction, you should follow the next set of instructions. They will tell you how to make a "backup", which will be an exact duplicate of your master DOS disk.

- Your DOS master disk either is WRITE PROTECTED with a small adhesive tab, or does not contain a write-enable notch and is permanently write-protected. **Do not remove the write protect tab if it exists.**

- Power up your computer system and all peripheral hardware. Place the DOS Master diskette in drive :0 and press the RESET button to boot the DOS diskette into the system. The DOS logo will now appear on the screen. Enter in the correct date (mm/dd/yy) and time (hh:mm:ss), and the message DOS Ready will appear.

- The name of your Master diskette will be displayed in the center of the screen above the DOS logo. It will probably appear as something like "LDOS531" or "LSDOS631". Write this name down, as you will need it in the following procedure.

- Now you are ready to make several BACKUPS of your DOS Master diskette. Follow the step by step procedures listed below.

After a backup is complete, you may see the message "Cannot clear MOD flags - Source disk is write protected" on the screen. This is just an informative message, and is normal when there is a write protect tab on the source disk. An explanation of "Mod flags" can be found in the DIR command section.

> **CAUTION**
> The default drive step rate will be 6ms for the Model III and 4. If this is too fast for your disk drives, use the additional parameter **STEP=3** inside the parentheses in the following FORMAT commands.

For single drive owners:

- Type in the command: **FORMAT :0 (NAME,Q=N)**

  The screen will clear and the DOS disk FORMAT utility will be loaded. You will see the following prompt appear:

  Diskette name ?

- Answer the prompt with the Master Disk Name recorded earlier. You will then see the message:

  Load DESTINATION disk and hit <ENTER>

- At this point, insert a new, blank diskette in drive :0 and press <ENTER>. After the FORMAT is complete, this message will appear:

  Load SYSTEM disk and hit <ENTER>

- Put the DOS Master disk back in drive :0 and press **<ENTER>**. Now type in the command:

  BACKUP :0 :0

  The message Insert SOURCE disk <ENTER> will appear on the screen. Since your DOS disk is the SOURCE disk, simply press **<ENTER>**. The message Insert DESTINATION disk <ENTER> will now appear on the screen.

- Put the disk you have just formatted into drive :0 and press **<ENTER>**. You will be prompted several times to swap the SOURCE and DESTINATION disks until the BACKUP is completed. At that point, the message Insert SYSTEM disk <ENTER> will appear. Place the Master in drive :0 and press **<ENTER>**. The BACKUP is now complete.

For multiple drive owners:

- Place a new, blank diskette in drive :1 and type in the command:

  DISKCOPY :0 :1 **(for DOS version 6)**
  QFB :0 :1 **(for DOS version 5)**

  Follow the prompts displayed; DOS will now make a duplicate copy of itself on drive :1.

MODEL I - Backing up the DOSXTRA disk.

To make a copy of the DOSXTRA disk, you must again format a disk. Use the same format instructions as for the Master disk, except answer the "Diskette name ?" prompt with DOSXTRA. To make the backup on a single drive system, type in the command **BACKUP :0 :0.** You will now be prompted to swap the source disk (the DOSXTRA) and the destination

disk (the one just formatted) until the backup is completed.

On a multiple drive system, give the command **QFB :0 :1**, and you will be prompted to insert the source disk (the DOSXTRA) in drive :0. Do so, and with another blank disk in drive :1 press <**ENTER**>. The duplication will now begin. When prompted to insert a system disk, place the DOS system disk back in drive :0 and press <**ENTER**>.

After the Backups are completed:

After the initial backups of your DOS disk are completed, remove the DOS master diskette from drive :0 and put it in a safe place. Be sure to leave it in its original jacket to protect it from dust and other contamination.

Label the backup copies of the diskettes as original backups of the DOS master diskette. You should use these diskettes to make any other backups you require. Do not use the master diskettes except to create a backup as just described.

It is extremely important that you now completely read the next section of the DOS user's manual. This section contains an overall view of the operating system as well as explanations of certain terms and conventions.

Entering DOS commands

Looking through this manual, you should notice a very distinct structure regarding the command syntax of the DOS system.

Each command will begin with a very brief description of the function involved. Immediately following will be a "syntax block". This block will be laid out to show the command syntax, allowable parameters with permissible abbreviations in CAPS, if any. A typical block might show the following:

THE COMMAND any files or devices (parameters)

FILES/DEVICES Descriptions

Parameter descriptions

The first line(s) in the block will show the allowable command structure. In some cases, more than one command structure will be shown. Throughout this manual, several words may be used as prepositions separating commands and/or parameters. They are:

TO ON OVER USING

The use of these prepositions is **always** optional; the DOS command will function the same whether they are used or not. They are merely a convenience to allow the user to enter a command in more conversational syntax. If a preposition is not used, a single space must be used between words. **Note that you cannot name a file with a filename the same as one of these four prepositions.**

Throughout this manual, you will see references to "***filespec***" and "*devspec*". These are the abbreviations for "***file specification***" and "***device specification***". The ***Introduction to DOS*** described what filespecs and devspecs are. Due to the device independence of DOS, it is possible to interchange these two specifications in some commands. For example, you can copy your keyboard to your line printer, or to a disk file. You can even append information from a device onto the end of a disk file! Each command will give detailed instructions and examples of interchanging filespecs and devspecs, if applicable.

Certain DOS commands allow the use of "*partspecs*" (partial file specifications) and "*not-partspecs*". A ***partspec*** is any or all parts of a filespec, generally excluding the password. For example, the full filespec for the DOS utility REPAIR is:

REPAIR/CMD.UTILITY:0

Some examples of partspecs would be:

REPAIR	**/CMD**	**REP/C**
REP:0	**R/C**	**REPAIR/CMD:0**

A *not-partspec* is simply a partspec preceded by a dash character, such as **-REPAIR, -/CMD,** etc. Also, a *not-partspec* would be used to exclude a certain file or group of files from a command, while a partspec is used to include a file or group of files. For example, using a partspec of **REP** would find a match with all of the following files:

REPAIR/CMD	**REPAIR/BAS**	**REPAIR/ASM**
REPEAT/BAS	**REPRESEN**	**REP102:3**

Since some of the DOS commands allow the use of partspecs, you can use the filename and extension fields to create files with common attributes, and then access them as a group. DOS also creates or uses default extensions during some operations. Other operations can then use these default extensions when searching for a file.

The parameters section of the syntax block will give a very short description of the allowable parameters for the command. This description will generally be very brief, as a complete explanation will be given in detail in the text of that section.

Please note that many command parameters may have a default value if they are not specified. For example, the DIR command used to view a disk's directory has a parameter called SORT. The default of this parameter is ON, so the directory display will automatically be in sorted alphabetic order.

A permissible abbreviation for each parameter is identified by the capital letter sub-string listed for the parameter; this may be a single letter. For instance, the *sOrt* parameter of the DIR command can be entered either as **sort** or the single letter **o**, as identified by its appearance in upper case. Note that **ON, YES,** and **Y** are completely interchangeable in most commands in the DOS system; **OFF, NO** and, **N** are similarly interchangeable in most commands.

Special command specifications

- *Drivespecs* must always be preceded by a colon, whether used as part of a filespec or as a stand alone parameter, except in the DIR

library command.

- The closing parenthesis may be omitted from any DOS command if it would have been the last character entered on the command line.

- It is totally acceptable to enter any filespec, command, or parameter in either upper or lower case.

- Numeric values for any parameter may be entered in either decimal or hexadecimal. Decimal numbers are entered in normal notation, such as **parm=32768**. Hexadecimal numbers are entered as **X'value'**, such as **X'F000'**, **X'0D'**, etc. Using **parm=X'8000'** would produce the same value as the previous example of **parm=32768**.

Immediate Execution Program

Using DOS version 6, you can create an Immediate Execution Program (IEP). Once you create an IEP, you can load and run it at the DOS Ready prompt simply by pressing <*><**ENTER**>

DOS stores an IEP in the SYS13/SYS file. Because DOS recognizes the program as a system file, DOS includes the file when creating backups and loads the program faster. To implement an IEP use the following syntax:

COPY filespec SYS13/SYS.SYSTEM6:drive (C=N) <ENTER>

The *filespec* can be any executable (/CMD) program file. The *drive* specifies the destination drive. The destination drive must contain an original SYS13/SYS file.

COPY TED/CMD:1 SYS13/SYS.SYSTEM6:0 (C=N) <ENTER>

DOS copies TED/CMD from drive :1 to SYS13/SYS in drive :0. At the DOS Ready prompt, when you press <*><**ENTER**,> DOS invokes the TED text editor.

If you type <*><**ENTER**> before you copy a file to the SYS13/SYS file, DOS displays the error message, No command <*> present, as SYS13.

Hardware related features

Your Disk Operating System (DOS) is a user-oriented, device independent system. DOS version 5 provides compatibility among the TRS-80 Models I and III, and the MAX-80, so your data files and most BASIC programs will be truly transportable. DOS version 6 is available for TRS-80 Models 4 and II/12/16/6000; most programs designed for the Model 4 may be usable on the Model II. Diskettes are compatible across Model I, II, III, and 4 computers, provided the floppy drive types are supported by hardware.

You will discover that this DOS reference manual will not answer questions about your computer's operation; consult your computer owner's manual for hardware reference material. This manual contains all reference material for the disk operating system; to get the greatest value out of your computer system, it will be necessary to read and study both your computer's hardware reference manual and this DOS reference manual.

This section will deal with generalized conventions that exist throughout the operating environment. It will also give an overall view of the total DOS system. Let's start by listing some of the hardware related features that you will find when using DOS.

- In DOS version 5, the **keyboard** will originally use the ROM driver. On the Model I, this will not provide key debounce, key repeat, type ahead, or any other advanced feature. The Model III has debounce, key repeat, and screen print built into the ROM driver; the supplied KI/DVR provides enhanced keyboard features, such as type-ahead.

- In DOS version 6, the **keyboard** driver is provided as part of the resident operating system.

DOS version 5 comes with a keyboard driver program called **KI/DVR**. The use of this driver is mandatory if functions such as key repeat, type ahead, screen print, printer spooler, KSM, MiniDOS, LCOMM, or the SVC table are to be used. It is strongly recommended that the KI/DVR program with the **TYPE** option be active in your runtime system. It requires very little memory space and will make using DOS even more pleasant. Use of the KI/DVR program will enable you to easily type in either upper or lower case. It also establishes the **<SHIFT><0>** key as a CAPS lock key.

If you are installing KI/DVR under DOS version 5 on a Model 4 computer, the <CAPS> key will be used for CAPS lock; the <CTRL> key will be used for CTRL, and the function keys will be active.

Once the KI/DVR program has been set, shifting between the CAPS lock mode and the normal upper/lower case mode can be accomplished by pressing the <SHIFT><0> (or CAPS) keys. In the normal upper/lower case mode, unshifted alphabetic keys are entered as lower case, and shifted keys as upper case, the same as on a standard typewriter. In the CAPS lock mode, any alphabetic character will be displayed as upper case, whether the <SHIFT> key is held down or not. On the Model III and under DOS version 6, you will initially find that all keyboard entries will be in lower case. The Model I will initialize in the CAPS lock mode. This may be changed on either machine, as described in the reference material for the SYSTEM command.

When using the KI/DVR - or resident DOS version 6 keyboard driver - program, the KSM and MiniDOS (where applicable) functions may also be used. Keys may be assigned special commands, functions, or characters with the KeyStroke Multiply (KSM) feature. These associated functions are then available when the <CLEAR> and desired unshifted key are pressed together. Due to this, it is necessary to press the <SHIFT><CLEAR> to clear the screen when the KI/DVR program is used; alternatively, you may use the CLS command. The DOS version 5 MiniDOS filter program will give you immediate access to certain DOS functions, such as a disk directory or amount of free space, a line printer top of form command, repeat the last DOS command, and a disk file delete command.

- The <SHIFT><BREAK> key combination will re-select a 5.25" disk drive that has "timed out" and hung up the system. This may happen if you attempt to access a drive with the drive door open, or if there was no diskette in the drive, etc. It should prevent having to reset the entire system. Ready the drive for access and then press the <SHIFT><BREAK> keys to complete the operation. Do not press the <SHIFT><BREAK> keys if the system is currently active!

- The disk drives in DOS can be 3.5", 5.25", 8", or hard disk. DOS will support a total of eight disk drives. The drives may be double/single sides and density, and up to 96 tracks on floppy disks. Hard disk support will be determined by the manufacturer

or distributor of the hardware. No more than four of any one drive type (3.5, 5.25, or 8 inch floppy, or hard drive) may be accessed. Of course, you must have the appropriate hardware and drivers for this. If you have purchased a supported hard disk system, driver programs and documentation for hard disk will be provided separately.

- The **video** display will allow display of upper and lower case characters, assuming your hardware is capable of lower case display. If you have a Model I with the switch type of modification, be sure to have the switch in the lower case position when booting. Keyboard entries will normally be displayed in all upper case unless the KI/DVR program has been set. If KI/DVR is used, keyboard entries will be displayed as determined by the mode (normal or caps lock) set with the <**SHIFT**><**0**> function.

- All **system hardware devices** are totally independent of the normal routing structure. Your system devices such as the video display and printer can be routed or linked almost any way you could desire - to each other, to a disk file, to another device, etc. You can even create your own logical devices!

- The **cassette** on the Model III can be used in either the 500 or 1500 baud mode. Use of the high speed (1500 baud) mode requires the use of the **HITAPE/CMD** program. Both the BASIC and CMDFILE utilities allow high speed tape operation.

- Once you have powered up your system, you can control the boot sequence to some extent. Note that if the <**BREAK**> key is held down during power up or reset, the computer will immediately enter ROM BASIC. Otherwise, you may be prompted to enter the date and/or time. After answering these prompts, there are several keys that will modify the remaining boot sequence if held down. They are presented in the reference material for the BOOT command. Once the system has booted and displayed the message "DOS Ready", it is ready to accept a command from the user.

DOS file descriptions

Throughout the manual, you will see references to "*filespec*" and "*devspec*". These are abbreviations for "***file specification***" and "***device specification***". Due to the device independence of DOS, it is possible to

interchange these two terms in most commands. For example, you can copy your keyboard to your line printer, or to a disk file. You can even append information from a device onto the end of a disk file! Each command will give detailed instructions and examples of the interchanging of filespecs and devspecs (if applicable).

Certain DOS commands allow the use of "*partspecs*" (partial filespecs). This will allow you to use the filename and extension fields to create groups of files with common filespecs, and then access these files as a group. DOS creates "**default**" extensions in the filespec during some operations. Other operations will use these default extensions when searching for a file. Following is a list of DOS default extensions along with suggestions for others that may help you "standardize" your file access.

ASM The extension used by most Editor/Assembler programs for source files.

BAS Used for BASIC programs. The BASIC provided with DOS version 5 uses this extension; the BASIC provided with DOS version 6 does not.

CIM DOS default for DUMP command. It stands for **Core IMage.**

CMD DOS default for LOAD and RUN commands, and PATCH and CMDFILE utilities. Used to indicate directly executable files.

COM Used by some systems to indicate COMpiled object code.

DAT Normal extension for data files.

DCT DOS default for the SYSTEM (DRIVER) command (Drive Code Table).

DVR DOS default for the SET command. Usually indicates a device "driver" program.

FIX DOS default for files to be used by the PATCH utility.

FLT DOS default for files used with the FILTER command.

JBL DOS default for Joblog files.

JCL DOS default for the DO command. It stands for Job Control Language.

KSM DOS default for KSM Utility. It stands for KeyStroke Multiply.

OVx BASIC extension for Overlay files (Overlay "x").

REL Used by some systems to indicate relocatable object code.

SCR DOS default for Scripsit text files.

SPL DOS default for the SPOOL command.

SYS DOS SYStem files only. Do not use for your own files!

TXT DOS default for the LIST and DUMP (with the ASCII parameter) command. This stands for TeXT file.

This next section will describe the various files found on your DOS master diskettes, and explain their functions. It will also describe how to construct a minimum system disk for running applications packages.

File group - system files (/SYS)

DOS's use of overlays requires that a SYSTEM diskette usually be available in drive :0 - the system drive. Since the diskette containing the operating system and its utilities leaves little space available to the user, it is useful to be able to remove certain parts of the system software not needed while a particular application is running. In fact, you will discover that your day-to-day operations will only need a minimal DOS configuration. The greater the number of system functions unnecessary for your application, the more space you can have available for a "working" system diskette. The following will describe the functions performed by each system overlay, identified in an DOS DIR command (using the SYS parameter) by the file extension, "/SYS". There are two system files that are put on the disk during formatting. They are "DIR/SYS" and "BOOT/SYS". These files are **never** to be copied from one disk to another! DOS automatically updates any information contained in these files.

SYS0/SYS

This is not an overlay. It contains the resident part of the operating system (SYSRES). Any disk used for booting the system **must** contain SYS0/SYS. It may be removed from disks not used for booting.

SYS1/SYS

This overlay contains the DOS command interpreter, the routines for processing the @FEXT system vector, the routines for processing the @FSPEC system vector, and the routines for processing the @PARAM system vector. This overlay must be available on all SYSTEM disks.

SYS2/SYS

This overlay is used for opening or initializing disk files and logical devices. It also contains routines for checking the availability of a disk pack (services the @CKDRV system vector), and routines for hashing file specifications and passwords. This overlay must also reside on all SYSTEM disks.

SYS3/SYS

This overlay contains all of the system routines needed to close files and

logical devices. It also contains the routines needed to service the @FNAME system vector. This overlay must not be eliminated.

SYS4/SYS

This system overlay contains the system error dictionary. It is needed to issue such messages as "File not found", "Directory read error", etc. If you decide to purge this overlay from your working SYSTEM diskette, all system errors will produce the error message, "SYS ERROR". It is recommended that you not eliminate this overlay, especially since it occupies only one granule of storage.

SYS5/SYS

This is the "ghost" debugger. It is needed if you have intentions of testing out machine language application software by using the DOS DEBUG command. If your operation will not require this debugging tool, you may purge this overlay.

SYS6/SYS

This overlay contains all of the algorithms and routines necessary to service the library commands identified as *"Library A"* by the LIB command. This represents the primary library functions. Very limited use could be made of DOS if this overlay is removed from your working SYSTEM disk.

SYS7/SYS

This overlay contains all of the algorithms and routines necessary to service the library commands identified as *"Library B"* by the LIB command. A great deal of use can be made of DOS even without this overlay. It performs specialized functions that may not be needed in the operation of specific applications. Use the PURGE command to eliminate this overlay if you decide it is not needed on a working SYSTEM diskette.

SYS8/SYS [DOS version 6]

This overlay contains all of the algorithms and routines necessary to service the library commands identified as *"Library C"* by the LIB command. A great deal of use can be made of DOS even without this overlay. It performs machine specific functions that may not be needed in

the operation of applications. Use the PURGE command to eliminate this overlay if you decide it is not needed on a working SYSTEM diskette.

SYS8/SYS [DOS version 5]

This overlay is needed to dynamically allocate file space used when writing files. It must be on your working SYSTEM diskettes.

SYS9/SYS

This overlay contains the routines necessary to service the extended debugging commands available after a DEBUG (EXT) is performed. This overlay may be purged if you will not need the extended debugging commands while running your application. In addition, if you purge SYS5/SYS, then keeping SYS9/SYS would serve no useful purpose.

SYS10/SYS

This system overlay contains the procedures necessary to service the request to delete a file. It should remain on your working SYSTEM diskettes.

SYS11/SYS

This overlay contains all of the procedures necessary to perform the Job Control Language execution phase. You may remove this overlay from your working disks if you do not intend to invoke any JCL functions. If SYS6/SYS has been purged (containing the DO command), keeping this overlay would serve no purpose.

SYS12/SYS

This overlay contains the routines to service the @DODIR and @RAMDIR directory vectors. These routines are used by the MiniDOS filter and may also be used by other applications programs that provide a directory display.

SYS13/SYS

This DOS version 6 overlay is provided as a base for copying an Immediate Execution Program (IEP).

File group - utilities (/CMD)

BACKUP	Used to duplicate data from one disk to another.
CMDFILE	A DOS version 5 disk/tape, tape/disk utility for machine language programs.
(L)COMM	A communications terminal program for use with the RS-232 hardware.
CONV	Used to move files from Model III TRSDOS to DOS.
DATECONV	A utility to update DOS disks earlier than x.3
DISKCOPY	A DOS version 6 command used to duplicate floppy diskettes.
DOS/HLP	A file of HELP information for DOS
FORMAT	Used to put track, sector, and directory information on a disk.
HELP	A utility used to access the ???/HLP files
HITAPE	Used for 1500 baud cassette operation on the Model III.
LOG	Used to log in a double sided diskette in drive 0. Also updates the drive code table information the same as the DEVICE library command.
PATCH	Used to make minor changes to existing disk files.
REPAIR	Used to correct certain information on non-DOS formatted diskettes.
SOLE	A Model I utility to install a DDEN boot driver under DOS version 5.

TED	A full screen ASCII text editor

File group - device drivers (/DVR)

COM	Used to access the RS-232 hardware, Model 4.
FDUBL	Used with double density modifications on the Model I under DOS version 5.
JL	The DOS JobLog driver program.
KI	The DOS version 5 Keyboard driver, providing Type Ahead, Screen Print, and special <CLEAR> key functions.
RS232L	Used to access the RS-232 hardware, Model I LX-80.
RS232M	Used to access the RS-232 hardware of the MAX-80.
RS232R	Used to access the RS-232 hardware, Model I.
RS232T	Used to access the RS-232 hardware, Model III.

File group - filter programs (/FLT)

FORMS	Allows the user to format printed output. [DOS version 6]
KSM	Establishes the KeyStroke Multiply feature, which allows assigning user determined phrases to alphabetic keys.
MINIDOS	Establishes certain immediate functions to shifted alphabetic keys. [Model I/III only]
PR	Allows the user to format printed output. [DOS version 5]

File group - BASIC and BASIC overlays

BASIC/CMD Disk Basic program.

BASIC/HLP A file of HELP information for DOS version 5 BASIC

BASIC/OV1 This file contains the library command overlay segment of the DOS version 6 BASIC interpreter. It contains the CMD"N" renumber feature overlay used with DOS version 5 BASIC.

BASIC/OV2 This file contains the routines for the DOS version 6 BASIC line copy, move, find, and search functions. For DOS version 5, it contains BASIC's cross reference CMD"X" feature.

BASIC/OV3 Error display and Sort routines for DOS version 5 BASIC.

BASIC/OV4 A DOS version 5 BASIC overlay to dump a list of active variables

Miscellaneous files

COPY23B/BAS Used to move files from Model I TRSDOS 2.3B or later.

FLOPPY/DCT Used by hard drive systems to set up the floppy disk drives. [DOS version 6]

MODx/DCT Used by hard drive systems to set up the floppy disk drives. [DOS version 5]

Creating a minimum configuration disk

All files except certain /SYS files may be removed from your run time drive :0 disk. Additionally, if the needed /SYS files are placed in memory with the SYSTEM (SYSRES) command, it will be possible to run with a minimum configuration diskette in drive :0 after booting the system.

For operation, SYSTEM files 1, 2, 3, 4, and 10 [and 8 for DOS version 5] should remain on drive :0 or be resident in memory. SYSTEM 2 [and 8 for DOS version 5 must be on the boot disk if a configuration file is to be loaded. SYSTEM 12 can be removed if the two "mini" directory routines are not needed. SYSTEM 11 must be present only if any JCL files will be used. Libraries (SYSTEM 6 and SYSTEM 7) [and SYSTEM 8 for DOS version 6] may be removed if no library commands will be used. SYSTEM 5 and SYSTEM 9 may be removed if the system debugger is not needed. SYSTEM 0 may be removed from any disk not used for booting. Note that SYSTEM 11 (the JCL processor) and SYSTEM 6 (containing the DO library command) require that both files be on the disk if the DO command is to be used - if you delete SYSTEM 6, you may as well delete SYSTEM 11.

When using DOS version 5 BASIC, the OV3 overlay must also be present. OV1 and OV2 may be removed if no renumbering or cross referencing will be done. When using DOS version 6 BASIC, you must retain OV1.

The presence of any utility, driver, or filter program is totally dependent upon the user's individual needs. Most of the DOS features can be saved in a configuration file with the SYSGEN command, so the driver and filter programs won't be needed on run time application disks.

DOS Passwords

The passwords for DOS files are as follows:

Group	/Ext	Password	
System files	/SYS	SYSTEM	[DOS version 5]
		SYSTEM6	[DOS version 6]
Filter files	/FLT	FILTER	
Driver files	/DVR	DRIVER	
	/DCT	DRIVER	
Utility files	/CMD	UTILITY	
BASIC		BASIC	
BASIC overlays	/OV$	BASIC	
CONFIG/SYS		CCC	

DOS system devices

The DOS operating system is a truly "Device Independent" system. Each device the system uses has its own "control area" of memory, called a DCB (Device Control Block). This is true for hardware devices as well as any "phantom" devices created by the user. Each device has its own driver routine, whether located in the ROM or in RAM.

An DOS device is used or created by specifying an asterisk followed by two alphabetic characters. Your original DOS master diskette is configured with six devices in the device control table. To view these devices along with the currently enabled disk drives, use the **DEVICE** command; DOS version 6 requires the "**B**" parameter. You will see the devices as listed:

***KI**	This is the Keyboard Input (your keyboard).
***DO**	This is the Display Output (your video screen).
***PR**	This is the PRinter output (your parallel printer).
***JL**	This is the Job Log (a time stamped log of commands).
***SI**	This is the Standard Input (presently unused by DOS).
***SO**	This is the Standard Output (presently unused by DOS).

Note that these are just the DOS system supplied devices; it is possible for you to create your own devices!

There is another DOS device that is referenced in this manual, even though not shown in a "*normal*" device table. This device is the Communications Line (***CL**); it is usually associated with your computer's serial port.

The DOS device independence makes it possible to route devices from one to another, or even to a disk file. It allows setting the device to a totally different driver routine. It makes possible single or multiple links of devices to other devices or to disk files. In other words, you may re-direct the I/O between the system, its devices, and the disk drives in almost any manner.

> **NOTE**
> Once a normal DOS device has been pointed away from its original driver routine, it may be returned to its normal power up state with the **RESET** command. A user created device may be either disabled or completely removed via the **RESET** and **[REMOVE | KILL]** commands.

Besides just sounding impressive, this device independence feature has many practical aspects. For example, the line printer is normally controlled by a very simple driver routine. However, the printer output may be filtered with the FORMS/FLT or PR/FLT program supplied with DOS.

This filter program allows you to set parameters such as the number of characters per line, the indent on line wrap around, the lines per page, the page length, etc. If you don't have a printer, simply use the **ROUTE** library command to route the printer to a disk file and all printing will be saved, enabling you to print it later, on a system with a printer. You could also route the printer to the display, and the characters will appear on the video instead of going to the printer.

Throughout this manual, you will see reference to the terms "*device*" (or "*devspec*") and "*logical device*". These terms represent the six system devices plus any devices the user has created. To create a device, please refer to the commands **LINK**, **ROUTE**, and **SET**. It is possible to use certain commands involving data I/O such as **APPEND**, **COPY**, and **[REMOVE | KILL]** with device specifications (devspecs) as well as file specifications (filespecs). It is not possible to imagine or describe all situations involving the possible uses and creation of devices. What this section will do is to explain the six DOS system devices. Any other device interaction or creation will be determined by the individual needs, sophistication, and imagination of the user.

*KI - The Keyboard

The ***KI** device is the keyboard. Using the KI/DVR program, or system resident DOS version 6 keyboard driver, will give you features such as type ahead and adjustable key repeat. It also establishes the **<CLEAR>** key as a special control key. This will be used by programs such as **KSM** and **MiniDOS** filters, and the **COMM** communications program. Certain other programs and commands, such as the **SPOOL** command, require that

KI/DVR be set for proper operation. These requirements will be specified where necessary. For DOS version 5, it is strongly recommended that the KI/DVR program be active during normal operation. It requires very little memory space, and enables many functions not available if KI/DVR is not used. The address of the *KI driver routine as shown with the **DEVICE** library command will be changed to a location in high memory if any of these functions are used, or if ***KI** is routed, set, or linked. The **DEVICE** command will also show the currently selected keyboard options.

You are advised not to route or link the ***KI** device unless you are extremely careful. You may inadvertently remove all input to the system or introduce totally unwanted characters. To send the ***KI** characters to a specific device or file, see the commands **APPEND** and **COPY**.

*DO - The Video Display

The ***DO** device is the video display. If your computer is equipped with a lower case modification, the power up video driver routine will include a lower case driver. Note that the ***DO** driver routine address will also change if *DO is involved in a route, set, or link. If you wish to send a screen display to a disk file, there are some simple ways to accomplish this. You can route the printer (*PR) to a disk file, and then link *DO to *PR. This will send all screen displays (including errors - backspace characters, etc.) to the printer, which is routed to a file so the output will really go to the disk. You could also enable the screen print function with the KI/DVR program and route the printer to a disk file. By pressing the **<SHIFT>**<⇓> (DOS version 5) or **<CTRL>** and the **<*>** keys (DOS version 6), the current screen display will be sent to the printer, and actually be written to a disk file. The ***DO** may also be linked to a disk file using a "*phantom*" user device as noted under the LINK command).

*PR - The Line Printer

The ***PR** device is the line printer. This device may be set to other drivers or routed to disk files very easily in the DOS system. A printer filter program is supplied on your DOS Master diskette, and is called either FORMS/FLT or PR/FLT. This program will allow you to set page size, line length, line indent on wrap around, and many other parameters. The operation of this driver routine is detailed in the *DRIVERS & FILTERS* section of the manual.

For serial printer owners, the supplied serial driver program (DOS version 5 - RS232/DVR; COM/DVR - DOS version 6) should enable you to interface with your printer. Use the SET command as follows:

SET *PR TO SERIALDRIVERNAME

Serial parameters may be altered with the SETCOM command; these should be determined by the internal settings of your particular printer. It should be noted that the BUSY or CTS line of your printer will usually have to be connected to the CTS line of your RS-232 cable to provide proper handshaking between the printer and the RS-232 hardware.

You may also use the **SPOOL** library command to create disk and/or memory buffers to store information being sent to ***PR** and spool it out as ***PR** becomes available (i.e. not in a busy state, such as printing a line).

If ***PR** is routed to a disk file or another device, it will not be necessary to have a line printer physically hooked to the system. All I/O to the printer will be sent to the appropriate device or file, and no lockup will occur. Using the PR/FLT program will also prevent the system from locking up if a print command is given with no line printer available.

*JL - The Job Log

The ***JL** device is the system's "Job Log". This unique feature will keep a log of all commands entered or received and most system error messages, along with the time they occurred. The time is determined by the setting of the real time clock, and may be set or changed with the **TIME** library command. To enable ***JL**, use the **SET** command to set ***JL** to its driver program called **JL/DVR** (see the section on *DRIVERS and FILTERS*). Every command or request processed through the DOS command line interpreter will then be logged in the ***JL** file, along with the time of the request. You may also enter comments or data directly into the Job Log by using a DOS comment line (any line beginning with a period), or by opening a Job Log file from BASIC and writing to it. The Job Log may be turned off by using the **RESET** library command to reset ***JL**, which will also close the associated disk file.

*SI - The Standard Input
*SO - The Standard Output

The *SI and *SO devices are system generated devices provided by DOS, although they are not presently used by the system. Both devices will initially be shown pointed (NIL). These devices are available to the user.

*CL - The Comm Line

This device has been designated *CL strictly for the purpose of standardizing examples throughout this manual, although any other available devspec could have been used (such as *DK, *CJ, *RS, *TM, etc).

The *CL device stands for the Communication Line. It is not an actual physical piece of hardware, but an area of memory used to talk to the RS-232 hardware. This device will allow characters to be sent and received using the RS-232 interface. The *CL will not be shown in the device table unless *CL is set to a serial driver program, but is always available to you. To enable *CL, simply use the SET command to set it to an appropriate driver program. There is a SERIAL/DVR program supplied on your DOS disk for this purpose. Please note that the (L)COMM/CMD communications program examples also use *CL as its RS-232 link (see the (L)COMM utility program).

Whenever I/O is needed via the RS-232 interface, the *CL will provide it. The RS-232 driver program allows *CL to interface between the DOS system and external devices such as a serial line printer, an acoustic coupler (commonly called a modem), a hard wired telephone data set, a paper tape reader, etc. Please refer to the *DEVICE DRIVER* section for a complete description of the allowable configurations of the RS-232 hardware.

DOS even provides a method to put your computer into a "*host*" mode for access by a remote terminal. To do this, set *CL to the serial driver program. It may be necessary to use SETCOM to specify the RS-232 word parameter as "WORD=8" and the parity parameter as "PARITY=OFF", depending on the terminal you are using. Then, issue the following commands:

```
LINK *KI *CL
LINK *DO *CL
```

This will logically connect the display and keyboard to the serial interface, and allow inputs to be taken from and output to be sent to a remote station via the serial hardware.

DOS disk drive access

Your DOS master diskette comes configured to access 3.5" or 5.25" floppy disk drives. The initial default drive configuration comes on the master diskette, and will remain consistent on any backup copy made from the master. To view your initial configuration, type in the command **DEVICE** at the "DOS Ready" prompt. Be sure no configuration file has been loaded. Do not have diskettes other than your master DOS diskette in any drive at this time, or you will not get a true picture of the default drive configuration.

DOS reserves a certain area of memory called the Drive Code Table (referred to as the DCT) to store information about the disk drive configurations. This drive code table is used by DOS any time access is attempted to a disk drive. The exact details of the drive code table will not normally concern the average user.

DOS provides features such as type ahead, the spooler, and (L)COMM that use the hardware interrupt clock. Because a disk rotational speed of 300 RPM may be evenly divisible by the interrupt clock rate, from time to time it may appear that the disk drives "go to sleep" for 10 to 30 seconds. To avoid this, we recommend that the drives be set to run at 302-303 RPM. This will assure optimum disk operation without degrading the overall reliability of the system. Alternatively, you may use the **SYSTEM (SMOOTH)** command to alter the DOS disk driver in those cases where it is not practical to adjust the floppy disk drive.

As stated above, issuing a **DEVICE** library command will show all currently enabled disk drives. Two of the display areas are very important to understand - the "logical" and the "physical" drive numbers. Refer to the following information:

- **LOGICAL DRIVES** - any number between 0 and 7

  Any time the DOS manual refers to a "drivespec", it will be referring to the logical drive number. The logical drive number will be shown in examples preceded by a colon. ":0" means drive zero, the first drive - drive ":1" means drive 1, the second drive.

- **PHYSICAL DRIVE** - shown as 1, 2, 4, or 8

  The physical drive number refers to a drive's position on the drive cable. For floppy drives, the numbers 1, 2, 4, and 8 will be shown by the **DEVICE** command, corresponding to the first, second, third, and fourth positions on a cable. When using special hardware such as a hard disk, an appropriate disk driver program will be supplied so you can set the logical and physical drive locations to match your actual needs. Since there are only four physical locations available, no more than four of any single drive type may be used.

The **DEVICE** command display also shows other drive information - the number of cylinders, the density, number of sides, step rate, and delay (5" floppies only). This information will be divided into two groups for ease of explanation. Group one contains information about the diskette in the drive, namely the cylinders, density, and side information. Group two contains information about the disk drive itself; the step rate and delay.

Diskette parameter information

All three of the diskette parameters shown in the device display (cylinder, density, and sides) are dependent on the diskette that was in the drive when it was last accessed. These parameters are established when the disk was initially created with the **FORMAT** utility. You will notice that the device display will show information for an enabled disk drive even if there is no diskette in the drive. This information will reflect the diskette that was last accessed in that drive.

DENSITY will be shown as SDEN (single density) or DDEN (double density). If you have a Model 4 or Model III, or have an appropriate double density board and are using the FDUBL driver for the Model I, you will see the prompt "Single or Double density ?" when formatting a diskette. The den-

sity of the diskette will be determined by how this question is answered.

CYLINDER and **SIDES** are interrelated terms. Most disk operating systems use the term "*tracks*" when referring to a diskette. A track is limited to one side of a diskette. The term "*cylinder*" refers to a track number on all surfaces of a disk. On a single sided diskette, a track is the same as a cylinder. On a 2-sided diskette, there are two tracks with the same number for each cylinder, one on the front and one on the back. A multi-platter hard drive may have as many as eight tracks per cylinder when using DOS. Again, the number of sides and cylinders are established by the hardware capabilities and by answering questions when formatting the diskette.

Drive parameter information

Two of the drive parameters shown in the device display may be adjusted with the **SYSTEM** library command. They are the drive "step rate" and the access "delay" for 3.5" and 5.25" floppy drives.

STEP RATE refers to the speed of the disk head movement when moving from one cylinder to another. To assure compatibility with all drive types, the DOS version 5 master disk comes with the step rate set to the slowest rate; the DOS version 6 master disk is set for the fastest step rate. **Most Model III users will be able to utilize the fastest step rate and should do so!** The step rates for all drives can be seen with the **DEVICE** Library command. If desired, they may be changed with the **SYSTEM (DRIVE=,STEP=)** command. The manufacturer of your disk drives should provide the maximum step rate the drive can handle. If you specify too fast a step rate, you will not be able to access the disk. You will also be asked to set a "bootstrap step rate" when formatting a diskette. This is the step rate that will be used for booting if the disk will be used as a system disk in drive :0. Again, too fast a step rate will keep the system from booting, so be sure to check out the fastest rate your drives can handle.

The bootstrap step rate will have no effect on any drive except drive :0 - you must use the **SYSTEM** command as previously explained to adjust your other drive step rates.

DELAY refers to the amount of time between the drive motor startup and the first attempted access. It is valid for 3.5" and 5.25" floppy drives only, as most 8" floppies and hard drive motors are always running. Some delay is necessary to allow a 3.5" or 5.25" drive motor to come up to its normal speed. The default will be 1 second for the Model I, and 0.5 seconds for

the Models III and 4; these values may be changed with the **SYSTEM (DRIVE=,DELAY=)** command. The drive manufacturer should provide information on the minimum delay time you can use.

Special Floppy disk drive types

The "standard" DOS floppy disk types are:

- 40 track, double density, single side - Models III and 4
- 35 track, single density, single side - Model I

DOS will also support any track count up to 96, double density, and double sided drives **provided you have the correct hardware!** For the Model III or 4, all that is needed to add double sided support is a double headed drive **and the proper drive cable.** No special driver programs are required.

For the Model I, double density requires a double density controller board and use of the FDUBL driver program. The resident driver provides support for double headed single density drives. Also, any time double headed drives are used, an appropriate drive cable is needed. The **standard drive cable will not work.** Note that you may be able to utilize a Model I drive cable if you invert the orientation of all connectors on their respective edge-card positions.

Standard disk formats

This table reflects the format used on standard DOS data diskettes.

Size	Density	Tracks	Sides	Sectors/ Granule	Grans/ Cylinder	Directory Sectors	Total Files	User Files	Free Space
3.5"	Double	80	2	6	6	32	256	240	710K
5"	Single	35	1	5	2	8	64	48	84K
5"	Single	35	2	5	4	18	144	128	169K
5"	Double	35	1	6	3	16	128	112	152K
5"	Double	35	2	6	6	32	256	240	305K
5"	Single	40	1	5	2	8	64	48	96K
5"	Single	40	2	5	4	18	144	128	194K
5"	Double	40	1	6	3	16	128	112	174K
5"	Double	40	2	6	6	32	256	240	350K
5"	Single	80	1	5	2	8	64	48	196K
5"	Single	80	2	5	4	18	144	128	394K
5"	Double	80	1	6	3	16	128	112	354K
5"	Double	80	2	6	6	32	256	240	710K
8"	Single	77	1	8	2	14	112	96	302K
8"	Single	77	2	8	4	30	240	224	606K
8"	Double	77	1	10	3	28	224	208	568K
8"	Double	77	2	10	6	32	256	240	1138K

Memory usage and configuration

Certain features of DOS are user selectable (i.e. the keyboard driver KI/DVR, the printer filter PR/FLT, Model I double density drivers, etc). To implement these features, DOS will load the necessary program into high memory; DOS version 6 allows some use of low-memory for drivers and filters. There is one term that is very important in the DOS operating system - HIGH$.

This term is pronounced "*High dollar*", and refers to a location that holds the address of the highest unused memory location. If DOS is using no high memory, then HIGH$ will contain X'7FFF', X'BFFF', or X'FFFF' for 32K, 48K, and 64K machines, respectively. To see the current HIGH$ value, use the **MEMORY** command. When DOS needs to use high memory, it does so in the following manner:

- Find the highest unused memory address by looking at the value

stored in the HIGH$ location.

- Install the necessary code in memory below the current HIGH$ value.

- Lower the HIGH$ value to protect the added program code.

Any code that DOS stores in high memory is written to be relocatable. This means that it can load anywhere in memory, and is not restricted to a specific area. Since DOS always respects the HIGH$ value, it will never attempt to overlay any programs loaded and protected by using the HIGH$ value in this manner.

Unfortunately, some applications programs do not always respect the HIGH$ value. As a result, programs or BASIC USR routines that load in high memory are not always written in a relocatable manner. They have a fixed load address, and **must** be loaded there to execute properly. If DOS has previously put program code in that memory area, it will be overwritten. This results in what is called a "memory conflict" - two pieces of program code that want to use the same memory area at the same time. When the DOS code is something like the KI/DVR program, this usually results in an immediate system crash.

Fortunately, it is possible to get around this problem by using the **MEMORY** command. To resolve a memory conflict, you need only to know the load address and length of the unrelocatable code. We will consider two cases - when the code loads at the very top of memory, and when it loads at some other point.

When the conflicting code loads at the very top of memory, it is very easy to resolve the problem. Since you know the load address of the code, use the **MEMORY** Library command to change the HIGH$ value to one byte below that address. For example, if a module of code loads from address X'F900' and goes to the top of memory, you would issue a **MEMORY (HIGH=X'F8FF')** command. DOS will now put any of its own high memory code below X'F900', protecting the module that will load there.

When the conflicting code does not load at the top of memory, you can use the same method just described to protect it. However, this will waste any memory between the end of the program and the top of memory. Let's consider the case where a module loads at X'F200' and extends to X'F3FF'.

There is 3K of space between the end of the module and the top of memory. To avoid wasting this space, use the following procedure.

- Load a DOS module into high memory (i.e., SET KI/DVR, install a filter, etc).
- Type in the command **MEMORY** with no parameters to see the current HIGH$ value.
- If the HIGH$ value is above X'F3FF', repeat the two prior steps. If the value has gone below X'F3FF', you will need to start over, stopping before you load the module that caused the HIGH$ value to go below X'F3FF'.
- Now, issue a **MEMORY (HIGH=X'F1FF')** command. This will protect the block of memory that will be needed by the unrelocatable module.
- Continue to load any other DOS modules as desired.

DOS has a command that will let you save your current memory configuration to a disk file, and have it load automatically every time you power up or reset the computer. This will let you store the memory allocation you have created, so you can "permanently" resolve any memory conflicts. Refer to the next section, *System Configuration.*

System configuration

Certain DOS features can be configured by the user and stored in a disk file. They will automatically be loaded each time the system is powered up or rebooted. The **SYSTEM** command gives a description of the configuration procedure in its (**SYSGEN**) parameter section [or **SYSGEN** command for DOS version 6]. Using the **DEVICE** command will show the current system configuration, including active disk drives and drive parameters, device I/O paths, and some user selected options currently active. Any high memory driver or filter programs will be saved in the configuration file. Be aware that all memory from the physical top to the current value stored in the HIGH$ pointer will be written to the disk. Be sure there is enough room on the disk to store the configuration file, or a "Disk Space Full" error may occur. Once saved on disk, any configuration may easily be changed by setting the desired parameters and doing another **SYSTEM (SYSGEN)** command. A configuration file may be deleted with

the **SYSTEM (SYSGEN=OFF)** command, or may be bypassed by holding down the **<CLEAR>** key when resetting or powering up the system.

Compatibility with other operating systems

To use files created on other operating systems, it will be necessary to move them onto diskettes that have been formatted by DOS. The DOS utilities **BACKUP, REPAIR,** and **CONV,** along with the **COPY,** and **COPY (X)** commands and the **COPY23B/BAS** program will usually provide the means to transfer your program and data files onto DOS formatted diskettes.

Under no circumstances should you use files on other than DOS formatted diskettes in your actual day to day operation. DOS provides either direct access or utility programs to read data from Model I TRSDOS 2.1, 2.2, 2.3, and 2.3B, single or double density DOSPLUS, Model III TRSDOS 1.2 and 1.3, single density DOUBLEDOS, NEWDOS and NEWDOS80.

In case of difficulty

Your DOS operating system was designed and tested to provide you with trouble free operation. If you do experience problems, there is a good chance that something other than the DOS is at fault. This section will discuss some of the most common user problems, and suggest general cures for these problems.

- The system seems to access the wrong disk drives, or cannot read the diskettes.

There are two main causes of this problem. If you have special hardware, it must be configured properly with the **SYSTEM (DRIVE=,DRIVER)** command. Check the drive table display with the **DEVICE** command and make sure that it shows the correct drive configuration.

If you have trouble reading diskettes created on other operating systems, refer to the REPAIR and CONV Utilities. Those sections will explain what is needed to make these types of disks readable.

- RS-232 communications do not work, or function incorrectly.

If you experience RS-232 problems, the first thing you should do is to make sure both "ends" are operating with the same RS-232 parameters (baud rate, word length, stop bits, and parity). If these parameters are not the same at each end, the data sent and received will appear scrambled.

Some hardware, such as serial printers, require handshaking when running above a certain baud rate. It may be necessary to hook the hardware's handshake line (such as the BUSY line) to an appropriate RS-232 lead, such as CTS.

- Random system crashes, re-occurring disk I/O errors, system lock up, and other random glitches keep happening.

If you encounter these types of problems, the first thing to check is the cable connections between the TRS-80 and the peripherals. The contacts can oxidize, and this can cause many different random problems. Clean the edge card connectors on the CPU unit and the peripherals, and be sure all other cable connections are secure.

If you experience constant difficulty in disk read/write operations, chances are that the disk drive heads need cleaning. There are kits available to clean disk heads, or you may wish to have the disk drive serviced at a repair facility. If you need to frequently clean the disk heads, you might be using some defective disk media. Check the diskettes for any obvious signs of flaking or excess wear, and dispose of any that appear even marginal. Tobacco smoke and other airborne contaminants can build up on disk heads, and can cause read/write problems. Disk drives in "dirty" locations may need to have their heads cleaned as often as once a week.

One common and often overlooked cause of random type problems is **STATIC ELECTRICITY.** In areas of low humidity, static electricity is present, even if actual static discharges are not felt by the computer operator. Be aware that static discharges can cause system glitches, as well as physically damage computer hardware and disk media.

If the system boots, but things continually go wrong from then on, hold down the **<CLEAR>** key during the boot and then re-configure the system.

DOS Commands

APPEND

This command lets you append (add) one file onto the end of another. Its primary use is with data files or ASCII-type text files. Files that are in load module format such as "CMD" or "CIM" type files, cannot be appended properly using the APPEND command; to append these types of files refer to the CMDFILE utility. BASIC programs cannot be appended unless saved in the ASCII mode. The syntax is:

APPEND filespec1 [TO] filespec2 (STRIP)
APPEND devspec [TO] filespec (ECHO,STRIP)

filespec1 **filespec2**	are valid DOS file specifications, including drivespecs.
devspec	is any valid, active device capable of generating characters.
Echo	is an optional parameter that will echo the characters to the screen when appending a device to a file.
Strip	is an optional parameter that will backspace the destination file one byte before the append begins.

APPEND copies the contents of file1 onto the end of file2. File1 is unaffected, while file2 is extended to include the contents of file1. The files must each have the same LRL (Logical Record Length) or the append will be aborted with the error message "Files have different LRLs" and neither file will be touched.

For example, suppose you have two customer lists stored in data files WESTCST/DAT and EASTCST/DAT. You can add the WESTCST/DAT file onto the end of EASTCST/DAT file with the command:

APPEND WESTCST/DAT:1 TO EASTCST/DAT:0

EASTCST/DAT will now be extended to include WESTCST/DAT, while WESTCST/DAT will remain unchanged.

You can also append a device (capable of sending characters) to a file. For example:

APPEND *KI TO WESTCST/DAT:2

This command will cause characters that are input on the keyboard to be appended to the file WESTCST/DAT on drive :2. Depressing the <BREAK> key at any time will terminate this type of append. Note that the keystrokes will not be shown on the display during this append, as the ECHO parameter was not specified.

APPEND *KI TO WESTCST/DAT:2 (ECHO)

This example will perform identically to the last, except that any key typed will also be echoed to *DO (the video screen).

APPEND PAGE2/SCR:0 TO PAGE1/SCR:0 (STRIP)

This example would append PAGE2/SCR to the end of PAGE1/SCR in the following manner. PAGE1 would be backspaced one byte, in effect allowing the first byte of PAGE2 to overwrite the last byte of PAGE1. This would be necessary when appending files such as SCRIPSIT files that have an internal end of file marker in the file. If the STRIP parameter was not used, SCRIPSIT would load the appended file only up to the first end of file marker, and ignore the appended PAGE2 section of the file.

ATTRIB - files

This command allows you to alter or remove the protection status of a file by changing passwords or the degree of access granted by a password. ATTRIB also allows the defining of whether a filename will be visible or invisible when a normal directory of the disk is displayed. ATTRIB will also allow you to alter the diskette name, master password, and lock or unlock all visible, non-SYStem files. The syntax is:

ATTRIB filespec.password:d (Owner="s",Prot="s",Vis|Inv)

password	password, used only if a password already exists.
Inv	makes file "invisible" to a directory command
Owner=	the new owner password
Prot=	the new protection level: [NOne, EXec, REad, WRit, UPdate, NAme, KIll, FUll]
Vis	makes file "visible" to a directory command

When you create a file, the password you specify becomes the owner password. If you don't specify a password, a string of eight blanks is assigned as a default password, in effect creating no password. If a protection level has not been assigned to a file, full access will automatically be granted regardless of any owner password. The OWNER password will still be required for full access on password protected files that have a protection level other than FULL. To have a file that allows no access whatsoever without the use of the owner password, change the protection level to NO (or NONE).

Without entry of a password, access up to and including the level of protection that is specified will be granted. The password that follows the "Owner=" is the owner password and always allows complete access to a file.

If the VIS or INV parameters are not specified in an ATTRIB command, they will remain unchanged. If the file is currently visible, it will remain so, and vice versa.

The levels of protection associated with the passwords are as follows:

LEVEL	PRIVILEGE
NOne	No access without password entry
EXec	execute only
REad	read, execute
UPdate	read, write (without extend), execute
WRite	write, read, execute
reNAme	rename, write, read, execute
KIll	All access except re-attrib; [V5]
ReMove	All access except re-attrib; [V6]
ALl	Allows total access
FUll	Same as ALL

The protection levels form a hierarchy, with the highest protection level (NONE) allowing the least amount of access.

ATTRIB sets or changes the protection of a file which already exists on a disk. There are several ways to use this feature. Here are some examples of the ATTRIB command:

```
ATTRIB CUSTFILE/DAT:0 (OWN=BOSSMAN,PROT=READ,VIS)
ATTRIB CUSTFILE/DAT:0 (O=BOSSMAN,P=RE,V)
```

This will protect the file CUSTFILE/DAT on drive :0 so that it can only be read by a file read routine. No password will be required to open and read the file because the protection level has been set to "READ". It can't be changed or written to in any way unless the owner password (BOSSMAN) is used when specifying the file, in which case full access would be given. Notice that the file will be visible in the directory.

```
ATTRIB ISAM/BAS:0 (OWN=SECRET,PROT=EXEC,INV)
ATTRIB ISAM/BAS:0 (O=SECRET,P=EX,I)
```

After invoking this command no one will be able to list this program when it is brought into BASIC, because the protection level has been set to

EXECute only. The only way this file can be read into the computer is with the RUN command in BASIC (RUN "ISAM/BAS"). This file cannot be loaded, listed or printed without using the owner password SECRET. Full access will be granted if the file is specified as ISAM/BAS.SECRET because the owner password has been given. Remember that the owner password will allow complete access regardless of the protection level that has been set. Notice that this file will be invisible in the directory because the INV parameter has been specified.

EXAMPLE: RUN"ISAM/BAS"

Any attempt to look at this program after it is running will cause the program to be deleted from memory. Listing the program in BASIC cannot be done unless the program is loaded using the password SECRET. Any attempt to interrupt the execution of the program will cause the program to be erased from memory.

ATTRIB ISAM/BAS.SECRET:0 (OWN=,VIS)
ATTRIB ISAM/BAS.SECRET:0 (O=,V)

This command will get rid of all passwords and make the file ISAM/BAS visible in the directory. Notice that the owner password of SECRET was required to re-attrib the file.

ATTRIB HOST/CMD:0 (INV)
ATTRIB HOST/CMD:0 (I)

This command will make the file invisible to the normal directory command DIR, without assigning any passwords to the file. To see invisible files, use the (I) parameter of the DIR command.

ATTRIB - disks

ATTRIB :d (Lock | Unlock,Mpw="s",Name="s",PW="s")

:d	is an optional drivespec, defaults to :0.
Lock	Locks all visible non-system files not currently protected by changing their password to the master password of the disk.
Mpw=	Specifies the disk's current master password.
Name=	Allows changing the disk name.
PW=	Allows changing the disk master password.
Unlock	This removes the password from all visible, non-system files, as long as their password matches the master password of the disk.

The ATTRIB command will allow you to change the disk name, the disk master password, and the password protection of all visible and non-system files. Any time the ATTRIB command is used, you will be prompted for the disk's master password if it is other than PASSWORD and not specified with the MPW= parameter.

ATTRIB :0 (UNLOCK, NAME="MYDISK")

This command will remove all passwords from the user's visible non-system files on drive :0, as long as the files' current password matches the master password of the disk. It will also change the disk's name to MYDISK. Since the current master password was not specified with the MPW= parameter, you will be prompted for it before this command is actually invoked, if it is other than PASSWORD.

ATTRIB :1 (NAME="DATA",PW="SECRET",MPW="BOSSMAN")

This command will change the name of the disk in drive :1 to DATA. It will also change the master password to SECRET. Note that the current master password was specified as BOSSMAN with the MPW= parameter.

ATTRIB (LOCK)

This command will first prompt you for the disk's master password, if other than PASSWORD. It will then change the passwords of all the user's visible, non-system, non-password protected files to the disk's current master password. This command will be carried out on drive :0, as no drivespec was used.

ATTRIB :1 (NAME)

This command will first prompt you for the drive :1 disk's master password, unless it is PASSWORD. It will then prompt you for the new name to be given to the disk.

AUTO

The AUTO command lets you modify the power up BOOT sequence, by specifying a command to be invoked immediately after power-up, reset or re-boot. The syntax is:

AUTO [= | ?][:d][*][dos-command]

=:d invokes the AUTO command on :d.

? displays the AUTO command currently on :d.

:d the designated drive otherwise drive :0 is used.

***** is optional and if used will disable the ability of <**ENTER**> to suspend the execution of the AUTO dos-command and also disable the <**BREAK**> key.

dos-command can be any executable DOS command with or without parameters; the maximum length is 31 [V5] 74 [V6] characters in length.

If the AUTO *dos-command has disabled the <**BREAK**> key, it is possible to re-enable the <**BREAK**> after the AUTO command has finished execution. See the *SYSTEM (BREAK=ON)* command for complete instructions.

Here are some examples of the use of the AUTO command.

AUTO BASIC

Will write the command BASIC as an "automatic key-in" on the drive :0 diskette, replacing any previous auto command. From that point on, every time you power up or press the reset button with that diskette in drive :0, BASIC will automatically be loaded into memory and invoked. An AUTO command takes the place of a keyboard input, just as though the command had been typed in and <**ENTER**> had been pressed.

AUTO *DO INIT/JCL:0

After this has been written to the drive :0 disk, power-up or pressing the reset button will cause the DO file INIT/JCL:0 to be invoked, which will allow several commands to be invoked automatically (see DO command and JCL). Note the asterisk immediately preceding the command. This is optional; when used it will disable the ability of the **<ENTER>** key to halt the auto command. The **<BREAK>** key will also be disabled from this point.

To restore the power up sequence to the normal DOS Ready, type:

AUTO

This will eliminate any stored automatic key-in by removing it from its storage place on the disk.

If a disk has an active auto command, it will be carried over to the destination disk when doing a mirror image backup.

You can override any breakable AUTO command during power up or reset by holding down **<ENTER>** during initialization. This may be your only way of regaining control of the system, if the dos-command is not a working program. If the AUTO command disables the **<BREAK>** key and the program is non-functional, it may seem impossible to regain control of that disk. Should this occur, simply boot another (non-AUTOed) disk in drive :0. When the DOS Ready appears, place the non-functional disk in drive :0, type **AUTO**, and press **<ENTER>**. The runaway AUTO command will then be removed from that disk.

BACKUP Utility

The BACKUP utility is provided to duplicate data from a source disk to a destination disk. The syntax is:

BACKUP :s [TO] :d (parm,parm)
BACKUP [-]partspec w/wcc:s TO :d (parm,parm)

:s, :d	the SOURCE and DESTINATION drivespecs.
Date=	Used to specify files within a range of dates. Selection format is: "M1/D1/Y1-M2/D2/Y2"; "M1/D1/Y1"; "-M1/D1/Y1"; and "M1/D1/Y1-".
Inv	Designate that files invisible to the directory are to be included as well as visible files.
Mod	indicates files Modified since last backup.
MPW="s"	passes the source disk's Master Password.
New	will backup only those files not already on the destination disk.
Old	will backup only those files already existing on the destination disk.
Query	parameter indicating Query each file before moving. The switch ON or OFF may be specified
Sys	Designate that system files are to be included in addition to visible files.
X	allows backups with no system disk in drive 0.

BACKUP will move all or part of the data from a specified source disk to a specified destination disk. The parameters of the BACKUP command may be used to determine which data will be moved. All of the parameters are optional, with only the source and destination drivespecs being

prompted for if not entered. If the source disk contains a password other than "PASSWORD", it will be prompted for if not passed with the **MPW=** parameter.

The BACKUP utility requires that destination disks must be formatted before the backup begins. Having the destination disk formatted will allow the BACKUP utility to determine if a Mirror Image (exact cylinder for cylinder copy) backup is possible, or if it will be necessary for the backup utility to do a file by file duplication.

When doing a backup by class from a disk of DOS version prior to x.3 to an x.3 or later disk, the access|user password (if any) will be removed and the x.3 style directory date and time information will be established. It will not be permissible to backup SYSTEM files from an earlier disk to an x.3 disk during a backup by class.

BACKUP will construct a SYSTEM disk from a formatted DATA disk during a backup by class if the SOURCE is a system disk. The **SYS** parameter also reconfigures a DATA disk into a SYSTEM disk by allocating directory entry codes for /SYS files in the Hash Index Table.

Both Model I and Model III DOS 5 versions of BACKUP can perform a mirror image duplication of a Model I dual-density system diskette provided the destination diskette has been previously formatted with a dual-density configuration.

There are three types of backups available with DOS. They are MIRROR IMAGE, BACKUP BY CLASS, and BACKUP RECONSTRUCT. Certain rules determine which type of backup will be done.

A mirror image backup will be attempted if the size (3.5" or 5.25", or 8" floppy), the density, and the number of sides are identical on the source and destination disks. The number of cylinders need not be identical as long as the destination disk has a cylinder count greater than or equal to the source disk.

A backup by class will be done if the user specifies a partspec or any parameter except "X" or "MPW" in the command line.

A backup reconstruct will be done if the size (3.5" or 5.25", 8", or hard), the density, or the number of sides differs between the source and destination disks.

A backup by class and a backup reconstruct function identically, doing a file by file copy. The only difference is that a backup by class is initiated by the user and a backup reconstruct is initiated by the system.

It may be necessary for backup to turn off the system real time clock during certain operations. For this reason, the message:

Note: Real time clock no longer accurate

may appear after the completion of the backup. This is merely an informative message reminding you the clock is no longer accurate.

If the backup is being done from a JCL file, the following rules will apply:

- If the backup is mirror image, the Pack IDs (disk name and master password) must be the same or the backup will abort.
- Backup with the **(X)** parameter, single drive backups, and backups with the **(Q)** parameter cannot be done from a JCL file.

As a final note, it is not allowable to specify passwords in any BACKUP command line. The BACKUP utility will ignore any password protection on a file, whether doing a backup by class or a mirror image backup.

Mirror image backups

A mirror image backup is basically a cylinder for cylinder copy from the source to the destination disk, with only those cylinders that actually contain data being moved. The date on the destination disk will be changed to the current system date. However, the boot sector containing the bootstrap step rate will remain untouched on the destination disk.

A mirror image backup will always compare the disk Pack ID's (disk name and master password) to make sure they are identical. If they are not, you will see the following message:

Different pack ID's! Abort backup?

Answer this question <Y> to abort the backup or <N> to continue the backup. If you use informational disk names when formatting diskettes, this checking of Pack ID's should help prevent you from backing up the wrong disks.

If the source and destination disks have different cylinder counts, the following message will appear:

```
Cylinder counts differ - Attempt mirror image backup?
```

Answer this prompt <Y> to attempt a mirror image backup or <N> to force a backup reconstruct. The destination disk will have its directory on the same track as the source disk, even though this may not have been the case before the backup began. The information on the destination disk will be updated to reflect the true cylinder count and available free space.

You may also see the following message appear at times:

```
Backup aborted! Destination not mirror-image.
```

This will occur if the destination disk is missing a cylinder that contains information on the source disk. This may be the case if the destination disk was formatted with fewer cylinders than the source disk, or if cylinders were locked out on the destination disk during formatting. You can use the FREE command to check the destination disk for locked out cylinders.

After all cylinders are moved to the destination disk, the backup utility will attempt to remove the mod flags from the source disk. If the disk is write protected, you will see the message:

```
Can't remove MOD flags - Source disk is write protected
```

This does not indicate an error. The normal use of the modification flag is to indicate that a file has been updated without a backup. Thus, one function of the BACKUP utility is to reset a file's *mod flag* after it completes the backup; this cannot be performed if the source disk is write protected.

After the backup has completed, the destination disk will have the same Pack ID as the source disk. The destination disk date shown with the DIR or FREE command will be changed to the current system date.

Backup by class and Backup reconstruct

These two backup types function identically, doing a file for file copy from the source to the destination disk. Unlike a mirror image backup, files that exist on the destination disk but are not on the source disk will remain untouched by the backup. When the backup is complete, the destination disk will contain all files moved from the source disk plus any other files that existed on the destination disk before the backup began. The destination disk Pack ID and date will not be changed by the backup. These types of backups may **not** be done on a single drive.

There are some things done when the file SYS0/SYS is included in this type of backup that are not readily apparent. Certain information about the default drive types and the state of the SYSTEM (SYSGEN) configuration parameter are moved from the source to the destination disk. The destination will have the following set equal to the source disk, regardless of how the destination disk was previously configured.

- The state of the SYSGEN (on or off) of the destination disk will be changed to match that of the source disk.
- The initial date and time prompts (on or off) on power up will be set to match those of the source disk.
- The default drive configurations will match those of the source disk.

It is possible to backup from a disk with a capacity greater than that of the destination disk, such as from a hard drive to a 5.25" floppy. To do this, format as many destination disks as will be needed to hold all of the information to be moved. As the backup progresses and the first destination disk is filled, you will be prompted with the flashing message:

Disk is full. Enter new formatted destination disk <ENTER>

At this point, remove the full destination disk and insert a new formatted disk in the drive. Pressing **<ENTER>** will cause the backup to continue. You may perform this disk swap as many times as necessary to complete the backup.

Backup will not allow a single file to be split across destination disks. If you have a file that is larger than the capacity of the destination disk, you will not be able to copy it with the BACKUP command.

Both backup by class and backup reconstruct will attempt to remove the mod flags from the source disk. If the source disk is write protected, you will see the following message appear after the first file has been copied:

Can't remove MOD flags - Source disk is write protected

To provide a more readable display, this message will not be displayed after every file, although the mod flags will not be removed from any source files.

Backups with the (X) parameter

The **X** parameter will allow you to perform backups without the need for the system files to be on the drive :0 disk. This will allow backing up data disks of different sizes or capacities on a two-drive system. Single drive owners will be limited to mirror image type backups.

If the backup will be by class or a reconstruct, two drives must be used. Also, the system modules 2, 3, 8 [V5], and 10 must be resident in memory, (see the **SYSTEM (SYSRES=)** command).

When doing a backup with the **X** parameter, you will be prompted to insert the proper disk in drive :0. You may be prompted to switch drive :0 diskettes, depending the type of backup you are doing and the system modules you have resident.

Using the backup parameters

Many of the backup parameters are identical to those in the DIR and PURGE commands. These parameters will allow you to choose the groups of files you wish to backup to your destination disk. All parameters may be used singly or in combination with any other parameters.

If no parameters are specified and a backup reconstruct is initiated by the computer, all visible files will be moved from the source to the destination. You may include invisible or system files with the **INV**, or **SYS** parameters.

The **MOD** and **DATE** parameters will allow you to choose only those files that have been modified since their last backup, or fall within a specified range of dates. This will be very useful on drives with large capacities, as it will not be necessary to backup the whole disk to obtain new copies of files that have changed. The **DATE** parameter accepts four formats to provide for selecting specific ranges of dates; note that the parameters are character strings and must be enclosed in quotes. These formats are:

- **DATE="M1/D1/Y1-M2/D2/Y2"** copies only those files whose mod dates fall between the two dates specified, inclusive.
- **DATE="M1/D1/Y1"** copies all files with mod dates equal to the specified date.
- **DATE="-M1/D1/Y1"** copies all files with mod dates less than or equal to the specified date.
- **DATE="M1/D1/Y1-"** copies all files with mod dates greater or equal to the specified date.

The **OLD** and **NEW** parameters provide an easy method to update disks without placing unwanted information on the destination disk. For example, using the **OLD** parameter will allow you to update your working disks if changes are made to the system, copying over only those files which are already on your working disks.

The **QUERY** parameter will show you each file before it is backed up, including the file's date and mod flag status. You may tell backup to copy the file by pressing the <Y> key. Pressing <N> or <ENTER> will bypass the file and show you the next. Pressing the <C> key will copy the current file, and shut off the query function. All files from that point on will automatically be copied.

The use of *partspecs*, *-partspecs* (not partspecs), and the *wcc* (wildcard character) will let you choose files based on their filename and extension. You may use these in combination with any of the previously mentioned parameters.

Examples of backup commands

Following are some examples and descriptions of the backup command. Please note that in all examples, the source disk's master password will be asked for if it is other than PASSWORD and is not specified with the MPW parameter. If the Q parameter is specified, the file's mod date and mod flag will be shown along with the filespec.

BACKUP :0 :1

This command will attempt a mirror image backup, using drive :0 as the source drive and drive :1 as the destination drive. If the drives are differently configured, a backup reconstruct will be invoked. All files will be moved from drive :0 to drive :1, with the exception of DIR/SYS and BOOT/SYS if a reconstruct is invoked.

BACKUP $:0 :1

The wcc ($) in this command will cause a backup by class. All files will be examined, and all files (except BOOT/SYS and DIR/SYS) will be copied because they will "match" the single wcc. This command is the way to force a backup by class in situations where a mirror image would normally have been done. This might be to remove unwanted "extents" from files on the source disk by copying them onto a cleanly formatted destination drive.

BACKUP :0 :1 (Q)

This command will function identically to the previous example, except that you will be asked before each file is moved. You will also see the mod date and mod flag for each file.

BACKUP :2 :1 (INV)

This command will copy visible files as well as all files that are invisible in drive :2's directory to drive :1, invoking a backup by class. In other words, this command will copy all files except the system files.

BACKUP :0 :1 (SYS)

This command will backup all visible and system files from drive :0 to drive :1, invoking a backup by class.

BACKUP :0 :1 (MOD,Q,MPW="SECRET")

This command will copy all visible files that have been modified (written to) from drive :0 to drive :1. It will query each file before it is copied, also showing the file's mod date and flag. The master password was passed with the (MPW=) parameter and will not be asked for.

BACKUP /CMD:0 :1
BACKUP $/CMD:0 :1

This command will force a backup by class, with the file class specified as /CMD. All visible files with the extension /CMD will be copied from drive :0 to drive :1. Note that the wcc ($) has no actual effect on the backup. Specifying the /CMD will look at all /CMD files, just as the $/CMD will. If the file exists on drive :1 it will be overwritten, otherwise it will be created at this time. No files on drive :1 will be touched except for the /CMD files copied from drive :0.

BACKUP $$$$$$AT:2 :3 (MOD)

This command will backup all visible files whose filename is eight characters long and contains "AT" as the last two letters. Only those files that meet this criteria and have been modified will be copied. A backup by class will be invoked.

BACKUP /$$S:1 :2

This command will backup all visible files whose extensions are three characters long, ending with the letter "S". The wcc ($) masks the first two characters of the extension, so the extensions /BAS, /TSS, /SYS, etc. would all match. A backup by class will be invoked.

BACKUP -/CMD:0 :1

This command will backup all visible files from drive :0 to drive :1, except those that have the extension /CMD.

BACKUP :1 :1

This command will backup between two disks in drive :1. You will be prompted to switch between the source disk and destination disk at the appropriate times. The disks involved in this type of backup must allow a mirror image backup, or the backup will abort. This command could be used to backup a data disk.

BACKUP :0 :1 (X)

This command will backup a disk in drive :0 to a disk in drive :1. Its primary use is to backup non-system disks, such as data disks, in a two drive system. When using this backup parameter, you will be prompted to insert the proper disk in drive :0. You may be prompted to re-insert a system disk into drive :0 during certain backups.

When the backup is complete, you will be prompted to insert a system disk back in drive :0. If the backup will be by class or a reconstruct, SYS overlays 2, 3, 8 [V5], and 10 must be resident in memory (see the SYSTEM (SYSRES) command).

BACKUP :1 :2 (OLD)

This command will backup visible files from drive :1 to drive :2, only if they already existed on drive :2.

BACKUP :1 :2 (NEW,Q)

This command will backup visible files from drive :1 to drive :2, only if they do not already exist on drive :2. You will be prompted before each file is moved, as the Q parameter was specified.

BACKUP /ASM:3 :2 (D="05/06/91-05/10/91")

This command will backup all visible files with the extension "/ASM", as long as their mod dates fell between the two dates specified, inclusive.

Many more examples of the power of BACKUP could be given, but the best method for the user to understand the scope of BACKUP is through its use. Experiment until you are comfortable with this utility.You can see exactly what files will be moved by a particular BACKUP command by doing a "DIR" command of the source disk using the same partspec and/or parameters you intend to use with the BACKUP.

BOOT

This command causes the disk in drive :0 to be booted into the system. It has the same effect as pushing the reset button or a power up condition. The syntax is:

BOOT <CLEAR> <⇒> <ENTER> <D>

Holding down the indicated key during the BOOT will result in the following actions:

<CLEAR>	No sysgened configuration will take place.
<ENTER>	No breakable AUTO commands will be done.
<D>	The system debugger will automatically be entered. Note that no sysgened configuration will be loaded.
<⇒>	For the Model III only, the video driver will be the ROM driver, not the normal DOS driver.

By typing in the BOOT command, the DOS system disk in drive :0 is booted back into the system. All devices will be returned to their normal power up configuration as if the system had been turned off and then turned on again. Any required filtering, linking, routing, or setting of the SYSTEM command parameters must be done again at this point, unless a SYSTEM config file has been generated on drive :0 by the use of "SYSTEM (SYSGEN)". If the system has been sysgened, the user configuration will be loaded and invoked at this time, and any AUTO command will be done.

If you wish to depress a BOOT over-ride key, you must do so immediately after the **<ENTER>** following the time entry. If the time prompt has been disabled, then depress the over-ride immediately after the **<ENTER>** following the date entry. And if the date prompt has been disabled, depress the over-ride key immediately after the **<ENTER>** following the BOOT command entry.

Holding down the <CLEAR> key will prevent any configuration file stored on the disk from being loaded. The configuration would have been created and stored with the SYSTEM (SYSGEN) or SYSGEN commands.

If you hold down the <ENTER> key, you will prevent the execution of any breakable AUTO commands from taking place. Refer to the AUTO command for additional details.

The <D> key will cause the debugger (non-extended) to be loaded and invoked. No configuration file will be loaded, and all memory above X'5200' for DOS version 5 or X'2400' for DOS version 6 (with the exception of X'4300'-X'43FF') will be untouched. Use of this debug function is explained under the DEBUG command.

On the Model III, holding down the <⇒> key during booting will prevent the DOS front end to the video driver from being loaded. The system will use the ROM video driver instead which may be necessary for certain programs. Caution: Using the ROM video driver will cause problems with keyboard type ahead, LCOMM, the SPOOLer, and any other DOS function that uses interrupt processing, and should not normally be done!

On the Model 4P, certain keys are used by the BOOT process. Depressing <F1> or <1> instructs the hardware to load DOS from a supported hard disk; <F2> or <2> from a floppy disk; <F3> or <3> specifies a Model III emulation mode which forces a load of a special MODELx/III ROM image file. <P> prompts for another disk which contains the image file; <L> forces a new load of the image file; and <N> mandates no load of an image file. Finally, <ENTER> is used to over-ride 4P hard/floppy automatic boot search procedures to force floppy booting; thus, if you use <ENTER>, you will need to release the key then re-depress <ENTER> to inhibit the automatic breakable AUTO command invocation.

BUILD

This command allows the user to build a file of desired character strings and save this file under any valid filespec. BUILD is in the system mainly to build ASCII files for use with the DO, KSM and PATCH features of DOS, although you may build files containing any characters X'00 to X'FF' with the HEX parameter. The syntax is:

BUILD filespec[/JCL] (Hex,Append)

filespec	is any valid DOS filespec .
Append	optional parameter that allows appending the BUILD data to the end of existing files.
Hex	optional parameter allowing a "packed" hexadecimal format only.

The **BUILD** command is used to create a file (or append to an existing file), a series of commands, comments, or character strings entered from the keyboard. If the filespec does not contain an extension, DOS will automatically assign a default extension of /JCL, for Job Control Language (see DO and JCL). If a file with the identical name exists, the BUILD command will abort with the error message "File already exists", unless the APPEND parameter has been specified.

The APPEND parameter will allow you to add to the end of an existing file. Be aware that some programs like the SCRIPSIT word processor place their own end of text marker at the end of a file. To properly extend this type of file:

- Use the BUILD command to create a new file consisting of the information you wish to append to the existing file.
- Then use the APPEND library command with the STRIP parameter to properly append the new information to the file.

Should you wish to create a KSM type file (see KSM filter), the file extension should be /KSM. This will prompt you with each key identifier

as you enter what you wish that key to represent. This type of build is detailed in the section on the KSM utility.

After the file has been opened, all characters that are typed will be placed in the file just as they appear on the video. Lines are limited to a length of 255 characters. Each line that is entered should be terminated by pressing the <ENTER> key. The file will be written to the disk when the <BREAK> key [for DOS version 5] or the <CTRL><SHIFT><@> key combination [for DOS version 6] is pressed as the first character of any new line.

Note: If you are building a /JCL file, lines will be limited to the maximum allowable characters permitted for a JCL command line.

The *HEX* parameter will allow you to enter characters other than those directly available from the keyboard. Any one byte character value may be entered in the HEX format "nn". The line length during a hex build will be 254 characters, allowing 127 hexadecimal characters to be entered. The HEX parameter uses a "packed" format, with no spaces or delimiters between bytes.

For example, you could create a character string containing graphics characters in the following manner:

818A90A10D

This line contains the hexadecimal bytes 81, 8A, 90, and A1. Note that the byte values are entered "packed" together, with no spaces or other delimiters between them. One of the possible uses for this format may be to build graphics strings to be used with the KSM function. If a file is to be used with the KSM function, do not embed the bytes 0D or 3B in the string unless you actually intend for these characters to be present, as these represent the Carriage Return and Semi-colon characters. They will be acted upon by the KSM file as end of line (0D) and embedded <ENTER> character (3B). Note that each logical line must be terminated with a 0D. Therefore several "logical lines" may appear on each "physical line". Each logical line is terminated with a 0D in the entered string, and each physical line terminated by pressing <ENTER>. The <ENTER> does not terminate the logical line.

EXAMPLE: F50DF10DFA0D<ENTER>

This would represent three logical lines in a KSM type file. Notice the three 0D's in the string.

Important: The HEX parameter will not cause the file to be stored in load module format; it will remain a normal ASCII image type file, even though some of the characters may be well out of the pure ASCII range.

When building files other than KSM or HEX, the line input length should be limited to 63/79 characters (for clarity). The build will be terminated when the <BREAK> key is entered as the first character in a line.

Following are some examples of the BUILD command.

BUILD THISFILE:2

This will check drive 2 for a file named THISFILE/JCL. If it exists, a "File already exists" error will occur. Otherwise, the file will be opened on drive :2. Note that the default extension /JCL was used, as no extension was specified in the command line. A /JCL file will not allow more than 63/79 characters per line to be entered.

BUILD MYKEYS/KSM

This command will search all available drives for a file named MYKEYS/KSM. If the file exists, a "File already exists" error will occur. Otherwise, this file will be created on the first available drive. Since the extension was specified as KSM, the prompts A>, B>, C>, D>, etc. will appear one at a time so each of the alphabetic characters may be assigned the character string(s) they are to represent (see the KSM filter). This build will terminate after the letter Z, or when a <BREAK> is used as the first character of a line.

BUILD SPECIAL/:0

This will build a file using the name SPECIAL with no extension. Using the "/" with no following characters is the only way to build a file without an extension (overriding the default /JCL extension). Note that the file SPECIAL cannot already exist, or an error will be generated.

BUILD MYJOBS/JCL (APPEND)

This command would search all available drives for a file named MYJOBS/JCL. If not found, it would be created on the first available drive. If the file already existed, any input from the build would be appended onto the end of the file. This is the way, for example, to extend an incomplete JCL file.

```
BUILD DISPLAY/BLD (HEX)
```

This command would build a file allowing the use of the "packed" HEX format. The file must not already exist, or an error will be generated. Information may be entered into this file as hexadecimal bytes, and will be stored as a normal ASCII format file. This format will allow 127 hex byte representations per physical line. Logical lines may continue on more than one physical line as long as a 0D does not appear, which would terminate the logical line. The **<ENTER>** is used to terminate a physical line.

If a non-hex digit is entered, the error message "Bad hex digit encountered" will be displayed, and the build will abort.

```
BUILD MYPROGA/FIX:0
```

This would build a file with the desired extension of /FIX for use with the PATCH program.

CAT

This DOS version 6 library command generates an abbreviated disk directory. It's syntax is:

CAT [-][partspec | filespec][:d1][-:d2] (parm,parm,...)

-	Exclude files which match.
partspec	Partial filespec with possible wild card characters.
:d1	Optional drive specification.
-:d2	Optional Model 4 drive designation to request a directory display of drives d1-d2 inclusive.
parm	Paramaters identical to that shown for the DIR command.

It's operation is identical to the **DIR** command with the parameter entry, **A=N.**

CLS

This library command clears the video screen. It's syntax is:

CLS

It's operation is identical to the simultaneous depression of the **<SHIFT><CLEAR>** keys. The action also restores the video to wide column mode and restores normal video if reverse video was active. The cursor is repositioned to row 0 column 0.

CMDFILE Utility

This is a Model I/III general purpose disk-to-disk, tape-to-disk and disk-to-tape, utility that provides the capability of appending two or more command (CMD, CIM, OBJ) files (load modules) or SYSTEM tape files that can be loaded with the BASIC "SYSTEM" command. Inherent in its capability of performing I/O to disk or tape are the following functions:

- Append two or more "COMMAND" disk files or "SYSTEM" cassette tape files into one file. This is useful to concatenate two or more separately assembled OBJECT code files, concatenate two or more non-contiguous blocks of code, or also couple two or more programs together so they load together.

- Offset a tape or disk file so that it loads into a region other than originally programmed. A driver routine is optionally appended that moves it back to its original load region. User options provide for disabling the clock interrupts and keyboard debounce routines in the event that the SYSTEM program would have overlayed the debounce routine of DOS.

- SYSTEM cassette tape files can be created from non-contiguous blocks of memory.

- For the disk user, during input of "COMMAND" files, the load address range of each block of code is displayed to the CRT and optionally to a line printer. The file's transfer address or entry point is also displayed.

Invoking CMDFILE

At DOS Ready simply type:

```
CMDFILE
```

The program will load and invoke.

Command Structure

All functions and procedures are specified by responding to a series of queries. Some queries request yes/no responses (abbreviated Y/N), some request disk/tape responses (abbreviated D/T), while others request specific information (i.e. file names, new addresses, etc.). Most yes/no and disk/tape responses can also be answered with a "C" to cancel the request and return to the main prompt as noted above. If you want to return to DOS, responding with "E" for EXIT will return you to the respective system. Each query displays the valid responses acceptable to it.

Address load log to printer (Y,N,E)? >

The address load log will be displayed only for files read in from disk. The timing on tape reads is too critical to perform the extra processing necessary to detect the load limits and display them during a tape read. If you are a disk user, have a line printer, and want this log displayed on your printer, respond with a "Y", otherwise respond with an "N". If you want to exit CMDFILE, enter "E". Whenever tis query is displayed, the memory buffer, used to store input files, will be reset to its beginning position to initialize for a series of input requests. This effectively "clears" the input buffer.

Input from disk or tape (D,T,E,C,Q) or <ENTER> to end reads? >

This displays when CMDFILE is ready to read in another file. Any file read in will be appended to any file previously input since the main query prompt. If you want to read in a disk file, respond with a "D". If the file is to be input from tape, respond with a "T". You may quit and return to DOS by entering an "E". A response of "C" will cancel the input and return you to the main query, thus reinitializing the memory buffer. If you have read in file(s) and want to begin a writing operation, respond with <ENTER>. The <Q> response permits display of a diskette directory. Its syntax is:

Qd

Where **d** is the drive spec. If you omit the drivespec, the drive :0 will be assumed. If you enter an erroneous drive spec, your entry will be ignored. If you enter a drive which is not in your system, the command will time out after about 10 seconds and you will receive another prompt.

If you responded with "D" in order to read in a disk file, you will be prompted for the filespec via the query:

```
Enter input file filespec >
```

Enter the filespec to begin the read operation. If you leave the file extension blank, a default extension of /CMD will be assumed. If any disk I/O error results, or any disk problem that results in the file not being read to completion, no fragment of the file will be added to the memory buffer. A disk file that is re-read will properly append any file previously read.

In order to read in a cassette file, you will be prompted to ready the tape with:

```
Ready cassette and depress <ENTER> [Model I]
Ready cassette and depress <H,L> [Model III]
```

Model III users should depress the <H> (1500 baud) or <L> (500 baud) key after preparing the tape for input. If the 1500 baud rate is used, the **HITAPE** utility must have been previously invoked. There is no need to enter a file name. The next program found on the tape will be read. The upper right screen will show flashing asterisks during the load. The letters **C**, **D**, or **BK** may appear if an error is detected. A checksum error during the load will display the message:

```
Tape checksum error detected - reread tape!
```

Any previously read in file will not be destroyed. The partial tape load will be ignored and subsequent reads will properly append any previously read in file.

If the file being loaded uses up your machine's memory, the message "Out of memory" will appear. Again, no file or files previously read into the memory buffer will be disturbed. You can proceed to save the buffer contents prior to the file that exhausted your machine's memory.

If you attempt to read in a file that is not a "COMMAND" or "SYSTEM" file, the read operation will cease and you will receive the message:

```
Requested file is not a COMMAND or SYSTEM file!
```

As a disk file is read, each block detected will produce the message:

```
Block loads from XXXX to XXXX
```

At the conclusion of the file read, whether from disk or tape, you will prompted for another tape or file after the transfer address (the program start address) is displayed as:

```
Transfer address (entry point) is XXXX
```

Once you have ended input and one or more files were input from disk or tape, the following prompt will be issued:

```
Program loads from base address XXXX to XXXX
Enter new base address or <ENTER> >
```

If you do not need to offset the output file, just depress the <**ENTER**> key. In general, if you are transferring a SYSTEM tape file to disk, and the tape file would ordinarily overlay the operating system's resident program (X'4200'-X'51FF'), you cannot load the disk file into memory from disk unless it is offset from the resident system. Once in memory, a block move routine can restore it to its original load point.

CMDFILE will not offset any part of a load module that loads below 4200H. This is to permit programs that purposely affect system variables or display messages to the memory mapped video (3C00H-3FFFH) to load properly. If you have input a program file that loads below 4200H and you are requesting to OFFSET the program, the following message will be displayed:

```
Program loads below 4200H
Enter Address to restrict offset or <ENTER> >
```

This gives you the option of restricting the offsetting operation below a specified address. For instance, if the program loaded a message directly to the screen, it would have a load block within the range 3C00H-3FFFH. You can maintain that load block in its original location (to the screen) by entering the lowest address above the screen area as identified in the ADDRESS LOAD LOG in response to the above query. This would provide the offset to any portion of the program originally loading at an

address greater than the screen end (3FFFH) and maintain the original load addresses for any block loading into an area below the address entered.

For example, the ADDRESS LOAD LOG begins with:

```
Block loads from 3C00 to 3C7F
Block loads from 5200 to 5282
Block loads from 5283 to 5304
...
```

The entire program module can be offset starting at 5200 by entering "**5200**" in response to the "Enter Address to restrict offset or <ENTER> >" query. In this manner, the load address of the load block addressed to the screen memory will be retained as 3C00 to 3C7F.

CMDFILE can read the library modules of DOS. If CMDFILE interprets the module being loaded as one conforming to the load format of DOS's ISAM files, then the query:

```
File has ISAM overlays - enter # >
```

will be displayed. If you enter the 2-character overlay number, CMDFILE will read only the desired overlay into its memory buffer. If you respond with "FF", then the entire module will be loaded. The DOS ISAM file structure is not described in this documentation.

If you want to change the load addresses of the output file (offset it), enter the new base load address. For example, if the existing load is from 4300H to 5000H and you want it to load starting at 5300H, enter the base address 5300H. After entering the new base address, you will receive the query:

```
Do you want to add the offset driver routine (Y,N,E,C)? >
```

A response of "**E**" will exit the program, while "**C**" will cancel the request and return you to the main query. If you do not want the restoring driver routine appended, respond with "**N**", otherwise respond "**Y**". It may not be immediately obvious why you would want to offset a file but not add the appendage. One use would be to change the base address of a relocatable driver routine. Another would be to change the load address of "Tiny PASCAL" files.

Do you want to disable the interrupts (Y,N,E,C)? >

A response of "E" will exit the program, while "C" will cancel the request and return you to the main query. If you want to disable the interrupts (which should be done if the program does any tape operation or will overlay the disk operating system's interrupt processing routine), then respond "Y", else "N".

Do you want to disable the keyboard debounce (Y,N,E,C)? >

A response of "E" will exit the program, while "C" will cancel the request and return you to the main query. If you want to disable the keyboard debounce routine (which should be done if the output file will overlay the disk system's debounce routine between approximately 4300H and 4400H), respond with a "Y", else respond with "N". The next query will be bypassed, as the driver routine appendage dictates the transfer address.

Enter new transfer address or <ENTER> to use XXXX >

If you want to change the transfer address (entry point), you can enter the new address. This is useful when appending two or more files since the transfer address used would default to the transfer address of the last file read in unless otherwise specified.

Output to disk or tape (D,T,E,C) or <ENTER> to restart? >

Again, a response of "E" will exit the program, while "C" will cancel the request and return you to the main query. Just depressing <ENTER> will also return you to the main query. Cancellation is available if you do not want to create an output file but rather just want to determine disk files' load addresses.

If you want to create an output disk file, respond with a "D". You will be prompted for the filespec with:

Enter filespec to write output >

After entering the filespec (remember /CMD will be used as a default extension), the output file will be written to disk (using VERIFY).

If you want to create an output tape file, respond with a "**T**". You will be prompted to enter the filename with:

```
Enter tape file name >
```

After entering the filename (up to six characters), you will be prompted to ready the cassette. The tape will then be written.

At the conclusion of the disk or tape writing operation, you will receive the query:

```
Module write is complete - Write another (Y,N,E,C)? >
```

The "**E**" and "**C**" responses are as before. A response of "**N**" will also return you to the scond query. If you had selected **TAPE** output, you would be prompted to ready the cassette and another copy would be written using the same file name as was previously entered.

COMM & LCOMM Utilities

The COMM utility [LCOMM for Model I/III] is a terminal program that provides communications capabilities between two computer systems, between your computer and a Bulletin Board System such as Forum-80, PSBBS, or other similar systems, or between your computer and a large public system such as CompuServe™, or other main-frames.The syntax of the COMM command is:

COMM *devspec [(parm,parm,...)]

devspec	is a valid DOS active device, usually *CL, SET to a serial driver.
NULL=	ON or OFF, the default is ON. If OFF is specified, it will prevent any nulls (00's) from being received.
XlateS=X'ffff'	Translates character ff to tt before sending.
XlateR=X'ffff'	Translates received character ff to tt.
XOFF=d	Sets the XOFF character to **d** [CTRL-S].
XON=d	Sets the XON character to **d** [CTRL-Q].

DOS Version 5 notes: You must have established the DOS keyboard driver (KI/DVR) via the command:

SET *KI to KI/DVR (parms...)

before attempting to enter LCOMM as LCOMM makes extensive use of control and function keys only available with the KI/DVR program. Also, it is imperative that the SPOOLer not be in use while using LCOMM since the SPOOLer shares the same task slot as LCOMM (LCOMM has its own printer spool buffer).

COMM provides the capabilities of keyboard send/receive, automatic spooling to a printer through a dynamic memory buffer, and the transfer of

files from system to system, without the need for handshaking when operating at 300 baud. For those users interfacing to systems supporting XON/XOFF protocol, COMM provides for optional XON/XOFF support using Device Control 1 (DC1) and Device Control 3 (DC3) ASCII controls.

COMM does not "talk" directly to the RS-232 hardware, but rather to a device that has been previously coupled to the RS-232 hardware through an appropriate software driver. This is accomplished with the SET library command and one of the supplied serial driver programs. The device COMM will interface with is passed as a device specification in the command line. The device name normally utilized for this purpose is "*CL", an acronym for "Communication Line", although any other device name could be used. However, throughout this section, the *CL device will be used for reference purposes.

It is also useful when receiving large files to pre-allocate the file space with the CREATE library command. This reduces the system overhead while running COMM.

COMM provides many user options. It interfaces with the user by utilizing the top row of keys as "Programmed Function" (PF) keys used in concert with the <CLEAR> key - just as the KeyStroke Multiplier (KSM) Filter uses the <CLEAR> key to provide special functions with the A-Z keys. Since most of the top row of keys are used in both their shifted and unshifted form, a brief user menu is provided to aid in jogging your mind until you become familiar with the programmed functions. This menu can be displayed by simultaneously depressing the <CLEAR><8> keys.

In describing the functions of COMM, the following conventions will be used:

<CLEAR> will represent the <CLEAR> key.

<CTRL> will represent the Model 4 <CTRL> key, or the Model I/III <LEFT SHIFT><⇩> keys.

<SH> will represent the <SHIFT> key.

<X> will represent the actual key that is to be used such as: <!>, <#>, <%>, <6>, <9> etc.

Some of the PF keys are used to select logical devices so as to be able to turn them on or off - indicating whether the device should be acceptable for I/O. Other PF keys control the selection of parameters associated with filespecs. Still others control additional functions which aid in the interface between two communicating users.

You may find the need for characters not normally visible on your keyboard. DOS provides all ASCII characters in the range 00-127. Accessing these characters is described completely in the KI/DVR section.

COMM will generate a modem break (extended null) if you press the **<BREAK>** key. To produce a normal TRS-80 "break" code for DOS version 5 (X'01'), press **<CTRL><A>**; for DOS version 6 (X'80'), press **<CTRL><C>**. A local *clear screen* function is also available, and can be requested by pressing the **<CLEAR><SH><(>** keys.

COMM uses all of available memory below (HIGH$) for dynamic buffering of certain device I/O. The amount of buffer space devoted to each device dynamically expands and shrinks according to how fast characters are sent to the device and how fast the device can accept them. Each buffer is essentially a variable length First-In-First-Out (FIFO) storage compartment. The amount of free space available for the buffers is noted in the bottom line of the menu display. When this free space shrinks to less than 2K (<2048 characters), a warning message is displayed and an X-OFF is automatically sent to the Communications Line. This is quite useful when receiving a file from a system that supports handshaking. A more detailed discussion on the use of handshaking will be presented in the "*Communicating With Other Computers*" section.

Using the PF keys

The first six function keys reference different devices in your system. The following table summarizes these references:

Function Key	Referenced Device	Devicespec
<CLEAR><1>	Keyboard device	*KI
<CLEAR><2>	Display	*DO
<CLEAR><3>	Printer Device	*PR
<CLEAR><4>	Communications Line device	*CL
<CLEAR><5>	File Send device	*FS
<CLEAR><6>	File Receive device	*FR

When COMM is first entered, *KI is in an "on" state. If you desire to turn it off to avoid accidentally brushing the keyboard while you are transmitting a file, you can turn off the keyboard by <CLEAR><1> followed by a <CLEAR><->, which indicates the "off" function. While the *KI is off, all PF keys are still active.

When COMM is first entered, the *DO is in an "on" state. If you desire to turn it off when, for instance, the printer has been activated, a simple <CLEAR><2> followed by a <CLEAR><-> will perform the requested function. The video display will be re-activated by a <CLEAR><2> followed by a <CLEAR><:>.

When COMM first starts, the printer device is off. If you want to direct the communications transactions to your printer, do a <CLEAR><3> followed by a <CLEAR><:>. Output being routed to the printer is fully buffered through dynamic memory buffers. Therefore, it is not necessary for your printer to be capable of operating at the communications line transmission rate. Even after transactions cease, you may discover the printer still typing away.

COMM starts with *CL in an ON state. You may wish to temporarily block output from being sent to the *CL so as to be able to review a file prior to transmission. Depending on your PF switch setup, if you go to a half-duplex mode (DUPLEX-ON) after turning off the *CL, you could perform a File Send (FS) which would display the file to your screen without actually sending it to the communications line. Of course, if the

distant end attempted to send to you while you had the *CL off, you would not receive their transmission.

With the "File Send" device, you can cause a file to automatically be transmitted to the distant end. This PF key works in concert with a number of other keys. Other PF keys exist to open a designated file, rewind a designated file, position to the end of a designated file, and close a designated file. As with the other devices discussed, the functions available to this File Send device are activated by the two-step process of first depressing **<CLEAR><5>** followed by some other PF key appropriate to the intended function. Specific details will be presented as the other PF keys are discussed.

The file receive devivce is used for either receiving a file being transferred to you, or for making a file copy of the communications line dialogue. This device will also be used to download from a bulletin board system. All of the PF keys available to the FS device are also available to the FR device. These will be discussed later. This device may be turned on or off by control characters received from the communications line if the HANDSHAKE switch is on. The characters received will be put in a memory buffer, and may be written to disk with the DTD function.

<CLEAR><7> - DTD

Dump To Disk (DTD) is used to write the memory buffer used with FR to the disk. DTD may be turned on before or after a file has been received. If turned on before, the file will be written to disk as it is being received. This will be necessary if the file will exceed the size of the FR memory buffer. When COMM first initializes, DTD is set to ON. When an FR RESET is performed, DTD is set to its OFF mode. Model I users may want to set DTD to OFF until an entire receive file (FR mode) has been received to guard against occasional dropping of a character during disk I/O. On the Model III, it will be necessary to wait until the entire file is received before turning on DTD if you are using floppy disks and any baud rate above 300.

<CLEAR><8> - Menu

This PF command will display a menu to the screen. It looks like this:

```
  *     *                                    *          * 00
DUPLX ECHO  ECOLF ACCLF REMND PEOF  DCC CLS 8-B  CMD  HNDSH EXIT
--1-- --2-- --3-- --4-- --5-- --6-- -7- -8- -9- --0-- --:-- ----
 *KI   *DO   *PR   *CL   *FS   *FR  DTD ??? ID   RES   ON    OFF
  *     *           * *          *

FR File: DOWNLOAD/TXT:3  MEMORY: 36K
```

Notice that the display will be left to right and in the relative positions of the keys which are used for the functions. This is not intended to be a complete menu, it is like having a built in "quick reference card". The <CLEAR><8> may be invoked at anytime. The screen display will be altered to display the menu (scrolled five lines), but no data will be lost as COMM will still be buffering the incoming characters. The buffered characters will be displayed after the menu is placed on the screen.

The menu display will show which devices and functions are active, as well the amount of available memory. The asterisks above and below the PF labels will indicate whether the function is active or not. Those above the labels are for the shifted PF functions; those below for the unshifted functions. *CL is shown with two asterisks, denoting that it is capable of both input and output. A single asterisk under a device indicates single direction I/O. If handshake is active, the auto X-OFF character selected will be shown in hex. Also, any FS or FR file that has been ID'd will have its filespec displayed.

<CLEAR><9> - ID

The ID function is used with either FS or FR to designate and open the desired file. If you are going to receive a file, you will perform an FR-ID by depressing <CLEAR><6> followed by a <CLEAR><9>. You will receive the prompt:

File name:

You should enter the file specification of your choice. The system will then open the file for receiving and await your next command. At this point the file is open and ready but is not turned ON (see ON, <CLEAR><:>).

If a receive file is already open, the system will ignore your ID request and issue the warning message:

File already open

This is to guard against inadvertantly issuing another FR-ID before you have closed the last file received. The same protection is available to FS. Only one FS file can be open at a time. You should close your send file after transmitting it.

<CLEAR><0> - RESET

The RESET PF key performs the function of closing either a receive file or a send file. A receive file must be closed so its directory is updated. Don't forget to turn "off" a receive file before closing it. You also must close a receive file to be able to receive a subsequent file. If a device is reset, its buffer is cleared.

<CLEAR><:> - ON

This PF key is used with one of the six previously mentioned device PF keys to turn "on" the device. For instance, once you have defined a receive file to the system with the FR-ID functions, you must do a FR-ON before any data will be written to it. Before a send file will start transmitting after the FS-ID, the FS-ON must be done.

<CLEAR><-> - OFF

This PF key performs the opposite function of the **ON** key. It is used in conjuction with any of the device keys to turn off the keyboard, video screen, printer, communications line, file send, or file receive. Just remember that like the ON function, the OFF function is performed in two steps. If you want to stop the receive file from further receiving, you FR-OFF by **<CLEAR><6>** followed by a **<CLEAR><->**.

The program function keys also have second functions programmed for them. These additional functions are accessible by depressing the **<CLEAR><SH>** keys along with the specified PF key. The following explains the capabilities of these second functions.

<CLEAR><SH><!> - DUPLEX

This PF key is the full-duplex/half-duplex switch. In the COMM ON/OFF arrangement, DUPLEX-ON designates half-duplex operation. In this mode, your video display screen will display your key entries or file

transmission as it is taking place. The DUPLEX-OFF mode is a full-duplex operation. Your video display will display what you send only if the distant end echoes back to you what it receives from you. Although it may be convenient to operate half-duplex (DUPLEX-ON) when communicating with another computer, it may be more useful for one of the computers to play HOST and operate in half-duplex with echo to the distant end while the distant end is full-duplex (DUPLEX-OFF). This will become more evident under the discussion of file transmission.

COMM initializes in the full-duplex or DUPLEX-OFF state. The PF key is also one that operates in concert with the ON and OFF keys. If you want to go to half-duplex after COMM initializes, you must depress the <CLEAR><SH><!> keys followed by the <CLEAR><:> keys.

<CLEAR><SH><"> - ECHO

This will provide the function normally undertaken by a host system. If ECHO-ON is specified, everything received from the communications line will be re-transmitted. This mode is desirable if the distant end must operate full-duplex and has no "local" copy. A caution to be observed is that if both ends are set for ECHO-ON, then the first character sent will be echoed back and forth indefinitely - at least until one end turns ECHO-OFF.

<CLEAR><SH><#> - ECHO-LINEFEED

The echoing of a line feed is the desired mode if the distant end is a dumb hard copy terminal that has its own local copy but expects the line feed to be sent by the host computer. With ECHO-LF-ON, anytime a carriage return is received from the communications line, a line feed character will be sent back, and a line feed will be added to any carriage return transmitted.

<CLEAR><SH><$> - ACCEPT-LINEFEED

COMM normally ignores the first line feed received after a carriage return. If this function is turned on, all line feeds will be accepted. This may be desirable if the distant end is another TRS-80.

<CLEAR><SH><%> - REWIND

The REWIND function works only with the *FR and *FS devices (FILES). It is used to rewind either file back to its beginning. For instance, say during the transmission of a file, the transmission is aborted prior to its completion. In order to resend it, it should be rewound to its beginning so the NRN pointer is pointing to the first record. REWIND performs that function.

<CLEAR><SH><&> - PEOF

This function is used to position a file to its end. A common use would be to append to an existing receive file. If you open a file for receiving by means of the FR-ID and then do a FR-PEOF, the existing receive file would **not** be overwritten with the new data, but rather the new data received will be concatenated to the old data.

<CLEAR><SH><'> - DCC

The DCC (Display Control Characters) function will force a display of any character received that has a value less than an X'20' as a two-digit hexadecimal number surrounded by braces. This will be useful if you suspect that unwanted control characters are being received.

<CLEAR><SH><(> - CLS

The CLS (Clear Local Screen) function will erase the contents of the screen without transmitting any character to the communications line. The cursor will be positioned at the upper left hand corner of the screen.

<CLEAR><SH><)> - 8-BIT

The 8-BIT switch is used to indicate that all 8 bits of each character received from the communications line are valid. If it is not turned on, bit 7 is stripped from each character received. Do not specify this switch unless the serial driver word length was set to 8.

<CLEAR><SHIFT><0> - DOS Library command

COMM makes use of the new @CMNDR command-and-return vector of DOS to provide you the ability to access DOS library commands while

running COMM. After the keystroke sequence, **<CLEAR><SHIFT><0>**, COMM prompts for the command with:

Enter command

When the command completes, COMM will regain control. This function may be used, for instance, to access SETCOM to change serial parameters.

<CLEAR><SH><*> - Handshake

If the handshake switch is turned on, COMM will respond to the following four control codes received from the communications line:

- X'11' DC1 - Resume transmission (standard X-ON character)
- X'12' DC2 - *FR ON
- X'13' DC3 - Pause transmission (standard X-OFF character)
- X'14' DC4 - *FR OFF

The DC2 and DC4 characters function identically to the *FR ON and *FR OFF programmed function keys. DC3 causes transmission through the *CL device to be halted until a DC1 is received. Reception is not affected. You can override a DC3 with the *CL ON keyboard command. Also, the actual character codes used for XON and XOFF may be altered when COMM is invoked.

HANDSHAKE may also be turned on with the sequence **<CLEAR><SH><*>**, followed by any non-PF key (rather than the ON key). If this is the case, any time COMM sends the specified character it will pause transmission until a DC1 is received or a ***CL ON** is issued directly from the keyboard. Typically, the **<ENTER>** key would be specified so that line-at-a-time transmission could occur with automatic stopping at the end of each line.

<CLEAR><SH><=> - EXIT

This PF key is used to return to the DOS command level. It does not require any ON or OFF. It is a stand-alone key sequence. Prior to exiting COMM, the *FR device is checked to see if an open file exists. In the

event that one does, it will be closed before the exit to DOS is made. This little feature will protect against your inadvertant exit without explicitly saving an open receive file.

Usage Tips

Some TRS-80 bulletin board systems permit the reception of graphics characters. In order to be able to accept these graphics, the RS-232 driver will have had to be initialized at 8-Bit word length and the 8-Bit mode in COMM will have to be used (<CLEAR><SH><)> followed by <CLEAR><:>).

The beginning COMM user may find it useful to make up a tape containing each key's function and place the tape directly above the keys. Avery or Pres-A-Ply self-adhesive removable labels may be used for this purpose. Any other pressure sensitive label will suffice. The labels can even be typed to provide a neater appearance. Your keys should be labeled as follows:

KEY	UNSHIFTED	SHIFTED
1	*KI	DUPLEX
2	*DO	ECHO
3	*PR	ECHO-LF
4	*CL	ACCEPT-LF
5	*FS	REWIND
6	*FR	PEOF
7	DTD	DCC
8	???	CLS
9	ID	8-BIT
0	RESET	CMD
:	ON	HANDSHAKE
-	OFF	EXIT

Communicating With Other Computers

Two examples will be given to show how COMM can be used to communicate with other computers. The first example will describe operations when communicating with a main frame. The second example will describe how COMM can be used to communicate between two TRS-80's.

When communicating with a main frame computer, it will not generally be necessary to change the default device or function settings when entering COMM. Most main frames operate in the host mode, and will provide echo functions for you. You must be sure, however, to have

specified the RS-232 parameters to match those expected by the main frame. To download a file, use the following procedure:

- Type in the command to cause the main frame to list the file, but do not press the <ENTER>.
- ID your receive file by pressing <CLEAR><6> followed by <CLEAR><9>. Type in the filename in response to the prompt.
- Type in <CLEAR><6> followed by <CLEAR><:> to open the receive buffer. If the file you wish to receive will be larger than your available memory buffer, you should press <CLEAR><7> followed by <CLEAR><:> at this time. This will cause the file to be written to the disk as it is being received.
- Press <ENTER> to start the file listing.
- When the listing is complete, type in <CLEAR><6> followed by <CLEAR><-> and if you have not already done so, <CLEAR><7> followed by <CLEAR><:> to write the file to disk. When the write is complete, type <CLEAR><6> followed by <CLEAR><0> to turn off FR and DTD and close the receive file.

Most main frame computers and some bulletin board systems support XON/XOFF handshaking. This mode is used for transmitting files to the host as a series of single lines. Each line is transmitted to the host while your machine pauses until the host acknowledges receipt by transmitting an XON to you thus resuming your transmission with the next line. To accomplish this, your file must have each line terminated with some line terminating character (usually an ENTER). As hosts generally have a maximum line length that they accept, you should be sure that your file does not have any lines exceeding the host's limit. The upload can follow this series of steps:

- Designate the file that you want to send by entering <CLEAR><5> followed by <CLEAR><9> and entering its filespec in response to the ID query.
- Turn on the handshake mode by entering <CLEAR><SH><*> followed by <ENTER> (assuming that the line terminating character in your file is <ENTER>). Open the file at the host end

and ready it for receiving by whatever command process your host requires. Then turn on your file send by <CLEAR><5> followed by <CLEAR><:>. You will note that one line of your file will be transmitted and then your machine will pause. Once the host sends you the XON, the next line of the file will be automatically transmitted. If you are operating in half-duplex, you may see the entire file displayed without any pauses as the file is read from your disk and is buffered awaiting transmission.

- When the transmission is complete, turn off the handshake mode by <CLEAR><SH><"> followed by <CLEAR><->. Then close up the file at the host end by whatever command process the host accepts. You may then close your file send by entering <CLEAR><5> followed by <CLEAR><0> which will turn off the FS and close the file.

If at any time you want to force the transmission to resume after a line is ended, you may turn the *CL back on by entering <CLEAR><4> followed by <CLEAR><:>. This is also explained under handshake.

When using COMM to communicate between two TRS-80's, it will be necessary for one end to turn on half duplex <CLEAR><SH><!> followed by <CLEAR><SH><:> and echo <CLEAR><SH><"> followed by <CLEAR><:>. If files are to be sent and received, this should be done at the receiving end. To transfer files, use one of the following two methods. If the receiving end's system is subject to character loss during disk I/O (some TRS-80 Model I machines) or you are operating above 300 baud on a Model III, then Method A should be used. If you are operating with a hard drive, your system can usually dynamically write to disk during transmission; thus, Method B should be chosen.

Method A - Intermediate Memory Buffering

- The sending end should do a <CLEAR><5> followed by a <CLEAR><9> and enter in the filespec to be sent.

- The receiving end should do a <CLEAR><6> followed by a <CLEAR><9> and enter in the filespec to be received. The dump-to-disk (DTD) must be turned off by entering a <CLEAR><7> followed by a <CLEAR><->. This will buffer the file in memory as it is being received. If the sending end supports XON/XOFF

handshaking (is it another COMM user?), then you should engage handshake by entering <CLEAR><SH><*> followed by <CLEAR><:>

- When both ends are ready, the receiving end should do a <CLEAR><6> followed by <CLEAR><:>, and the sending end should then do a <CLEAR><5> followed by <CLEAR><:>.

- If your free buffer space decreases to less than 2K during receipt of the file, a warning message will be issued and an X-OFF will automatically be sent to the sending end. Transmission from the sender should cease. Once it does, dump the receive buffer to disk by turning on DTD with <CLEAR><7> followed by <CLEAR><:>. You can observe the increase in available free buffer space by displaying a menu as the buffer is written to disk. Once ample free space is available, turn off the DTD with <CLEAR><7> followed by <CLEAR><->. You then can manually restart the sender's file by transmitting an X-ON from your keyboard with <CONTROL><Q>.

- After the file is totally received, the receiving end should do a <CLEAR><6> followed by a <CLEAR><->. The last receive buffer should be dumped to disk by turning on DTD with <CLEAR><7> followed by <CLEAR><:>. The sending end should do a <CLEAR><5> followed by a <CLEAR><-> then a <CLEAR><5> followed by a <CLEAR><0>.

- When the receiving end file is finished writing to the disk, close the file by resetting the FR device with a <CLEAR><6> followed by <CLEAR><0>. This will do a FR-OFF, a DTD-OFF followed by a close of the file just received.

Method B - Dynamic Dump to Disk

- The sending end should do a <CLEAR><5> followed by a <CLEAR><9> and enter in the filespec to be sent.

- The receiving end should do a <CLEAR><6> followed by a <CLEAR><9> and enter in the filespec to be received. The dump-to-disk (DTD) must be turned on by entering a <CLEAR><7>

followed by a <CLEAR><:>. It may already be ON. Its state can be ascertained by obtaining a menu and noting if an asterisk is positioned beneath its key in the menu display.

- When both ends are ready, the receiving end should do a <CLEAR><6> followed by <CLEAR><:>, and the sending end should then do a <CLEAR><5> followed by <CLEAR><:>. This will turn on the receive and send files.

- After the file is totally received, and the file is finished writing to the disk, close the file by resetting the FR with a <CLEAR><6> followed by <CLEAR><0>. This will do an FR-OFF, a DTD-OFF followed by a close of the file just received.

CONV

The CONV utility will allow you to move files from a Model III TRSDOS diskette onto an DOS formatted diskette. Two drives are required. Model I owners must have double density hardware to use this utility. The syntax is:

```
CONV :s (Dir)
CONV [partspec w/wcc]:s :d (parm,parm,...)
```

:s	is the source drive. It cannot be drive :0.
:d	is the destination drive.
Dir	Requests a directory of files on the source disk.
Inv	Convert invisible files.
New	Convert files only if they do not exist on the destination disk.
Old	Convert files only if they already exist on the destination disk.
Query	Query each file before it is converted.
Sys	Convert system files.
Vis	Convert visible files.

Parameters - (VIS,INV,SYS)

If none of these parameters are specified, all file groups will be considered. Specifying only one parameter will automatically exclude the other two. Thus to move all files except the system files, you would specify both VIS and INV.

Parameters - (NEW,OLD,QUERY)

The **NEW** parameter is used to move files onto the destination disk only if they do not already exist. The **OLD** parameter will move only those files that already exist on the destination disk.

Unless you specify **QUERY=NO**, CONV will ask you before each file is moved onto the destination disk. You should answer the prompt <Y> to move the file, or press <N> or <ENTER> to bypass it.

You may specify a *filespec* or *partspec* to be used to determine which files to move. Wildcard characters are also acceptable. Refer to the following examples.

CONV :2 :1

This example will allow you to move all files from drive :2 onto drive :1. You will be asked before each file is moved. If the file already exists on drive 1, you will be asked again before it is copied over.

CONV :1 :0 (VIS,Q=N)

This example will move all visible files from drive :1 onto drive :0. You will not be asked before each file is moved.

CONV /BAS:2 :0 (NEW)

This example will move only those files with the extension /BAS from drive :2 to drive :0. Because the **NEW** parameter was specified, only those files that do not already exist on drive :0 will be moved.

CONV $$$DATA:1 :2 (OLD)

This example will move those files that have the characters "DATA" as the fourth through seventh letters in their file name. You will be asked before each file is moved, and only those files that already exist on drive :2 will be considered.

COPY

Copies data from one file or device to another file or device. The syntax is:

```
COPY filespec1 [TO] filespec2 | partspec | devspec
        (Lrl=nnn,Clone=sw)
COPY filespec1 [TO] :d (Lrl=nnn,Clone=sw)
COPY filespec:d (X)
COPY devspec [TO] filespec | devspec  (Lrl=nnn,Echo)
```

Clone	indicates the desire for an exact duplicate of the directory entry of filespec1. All ATTRIButes will be copied with the file. CLONE defaults to ON.
Echo	will cause any characters copied from a devspec to be echoed to the screen.
Lrl	specifies the logical record length at which filespec2 is to be set; nnn=(1 to 256).
X	is an optional parameter that allows a single drive copy from a non-system diskette.

Important: COPY should not be used to move System (/SYS) files from one disk to another. The BACKUP utility must be used for this purpose.

When copying from an x.2 or earlier version disk to an x.3 disk, the old user (access) password, if any, will be removed and the x.3 style date and time information field will be established, the time being set to 00:00:00 (12:00am).

You should become familiar with this important command as it is used in the DOS system, so the full power of this feature can be utilized. Special attention should be given to the ECHO, LRL and CLONE parameters.

LRL is a parameter that allows the establishment of a new logical record length for a file during the copy process. If not specified LRL will default to the LRL of the file being copied. This can be very useful when

restructuring data files and for changing ASCII type files to be compatible from one application to another. It may also be needed when converting a source file from one language to another to allow the file to be read by another application language.

With *CLONE*, the copy will not only duplicate the contents of the file but will also duplicate the directory entry. The owner password will be copied as well as the assigned protection level, the visibility, the create flag, and the modified status of the file. Files that are copied with the CLONE parameter will not have their date altered. The same last-written-to date that appeared in the original will be moved to the CLONE-copied file. If the CLONE parameter is turned off, the date that was set as the system date will be written to the directory as the last-written-to date for the copied file.

If *CLONE* is not used, and an existing destination file was being copied over, the attributes of the destination file (except for the date) will be unchanged. If the COPY command creates a new file, any password included will become the owner password of the destination file, and the file will be visible, even if the file it was copied from was invisible. See the ATTRIB library command for more on file attributes.

ECHO can only be specified when the source for the copy is a device. If specified, all characters will be echoed to the screen as they are sent to the destination file or device.

The *X* parameter provides a means of copying between two non-system diskettes. During this copy, the user will be prompted to switch disks as necessary. See the example of an X parameter copy at the end of this section.

In the following examples, the use of the word "TO" between filespecs or devspecs is optional. *Spec1* and *spec2* need only be separated by a space.

Examples of COPYing: filespec1 TO filespec2

Note that when copying files, if filespec2 already exists on the destination drive, it will be overwritten by the copy.

When copying files, the filename, extension, and password of filespec2 will automatically default to those of filespec1 if they are not specified. See the following examples.

```
COPY TEST/DAT:0 TO TEST/DAT:1
COPY TEST/DAT:0 TO /DAT:1
COPY TEST/DAT:0 TO TEST:1
COPY TEST/DAT:0 :1
```

These four commands will invoke identical copies. All parts of filespec2 will default to those of filespec1 if not specified. The use of the word TO is optional in any copy command.

```
COPY TEST/DAT TO :1
```

This command will search the disk drives until it finds a file named TEST/DAT and then copy it onto drive :1, using the filespec TEST/DAT.

```
COPY TEST/DAT.PASSWORD:0 TO :1
```

This command would copy the password protected file TEST/DAT.PASSWORD from drive :0 to drive :1. The file on drive :1 will be named TEST/DAT.PASSWORD. Remember that all parts of filespec2 including the password will default to those of filespec1 if they are not specified.

```
COPY TEST/DAT:0 TO TEST/DAT.CLOSED:1 (C=OFF)
COPY TEST/DAT:0 TO .CLOSED:1 (C=OFF)
```

These commands will copy the file TEST/DAT from drive :0 to drive :1. The file on drive :1 will be named TEST/DAT, and have the owner password set to CLOSED. Notice that the second command dynamically assigned the name and extension of filespec1 to filespec2 and then added the password CLOSED. If a password exists on the file being copied it cannot be changed during a copy. Also, it will be necessary to turn off the CLONE parameter when assigning a password with the copy command. To change a password see the ATTRIB command.

```
COPY TEST/DAT:0 TO MYFILE:1
```

This command would copy the file TEST/DAT from drive :0 to drive :1, with the file on drive :1 named MYFILE/DAT. Notice that the extension of filespec2 was not specified and defaulted to /DAT.

COPY TEST/DAT:0 TO MYFILE/:1

This command will copy the file TEST/DAT from drive :0 to drive :1, with the file on drive :1 named MYFILE. There will be no extension on MYFILE because the "/" with no other characters was specified in filespec2.

COPY DATA/NEW:0 TO /OLD:0

This command will copy the file DATA/NEW from drive :0 to a file named DATA/OLD on drive :0. The filename was not specified for filespec2 and defaulted to that of filespec1 (DATA).

COPY DATA/V56:0 TO DATA/V28:1 (LRL=128)

This command will copy the file DATA/V56 on drive :0 to a file called DATA/V28 on drive :1. These two files will contain the same data but the logical record lengths will not be the same. We will assume that the original file had a record length of 256. This would be a normal "random" type data file. The file DATA/V28 will be created by the copy and will have a record length of 128 bytes. This ability to reset the LRL of a file is very useful when converting data to be used by a BASIC that can deal with blocked files (record lengths less than 256), such as the BASIC you run with DOS. This function is also necessary when you wish to append two files but cannot because they have different logical record lengths. By copying one of these files and setting the LRL parameter to the desired length, the record length can be adjusted and the APPEND library command will then function. Also see RESET to change a file's LRL.

COPY MANUAL/TXT.JWY:0 TO :1 (L=128)

This command will copy the file MANUAL/TXT with the password JWY from drive :0 to drive :1. In the process of doing the copy the LRL will be changed from whatever it was to 128. Note that the LRL parameter was abbreviated to "L" in the above example.

COPY CONTROL:0 /ASC:1 (LRL=1)

This will copy the file CONTROL to CONTROL/ASC on drive :1. The LRL of the file will be changed from whatever it was in CONTROL to one byte in length. This is an excellent way to convert a data file to a file that could be handled by a word processor (providing the data file was ASCII to start with).

Examples of COPYing: devspec to devspec

When copying from devspec to devspec, it is very important that all devices specified be assigned and active in the system. Any routing or setting that has been done to the devices may affect the copy. Some caution is necessary when copying between devices, as non-ending loops can be generated and lock up the system. In other words, **please** do not involve devspecs in your copies unless you thoroughly understand the procedures and constraints involved. Destruction of files and/or locking up the system could easily result from lack of user understanding when using this complex structure of the copy command. Following are a few examples of possible devspec to devspec copies. The results produced by these copies may possibly be duplicated through the use of other DOS commands (such as ROUTE and/or LINK).

COPY *KI TO *PR

This command will copy the keyboard to the printer. As keys are pressed, they will be sent to the line printer. Depending on the printer, the characters may be printed immediately or may require that a linefeed/carriage return be sent before printing. The keystrokes will not be visible on the video because the ECHO parameter was not specified. Hitting <**BREAK**> will terminate the copy.

COPY *CL TO *PR (E)

This command will COPY *CL (the RS-232 Comm Line) to *PR (the line printer). Each character that is received by the RS-232 will be processed by the RS-232 driver and then presented to the line printer. Since the ECHO parameter (E) was specified, each character will also be echoed to the screen. Prior to executing this command, the *CL device must have been set to an appropriate serial driver.

COPY *KI TO KEYIN/NOW:0

This would allow the sending of all keyboard entries to the disk where they would be stored in a file named KEYIN/NOW. If the file already exists it will be written over. Because the characters that are typed are going directly to the file, they will not appear on the screen. To view the characters, specify the **ECHO** parameter. To terminate this copy you should depress the **<BREAK>** key. The file will then be closed and DOS Ready will appear.

COPY ASCII/TXT:0 TO *PR

This command will copy the contents of the file ASCII/TXT to the line printer. Although this command is functional, it would give the same output as would the LIST command with the "P" parameter. The copy in this example will terminate automatically when the end of the file is reached.

Examples of COPYing with the X parameter:

The command **COPY filespec:d (X)** is similar to a regular copy, except that the "X" parameter will allow transferring a file from one diskette to another without requiring an DOS system present on any disk involved in the copy.

The colon and drive number are optional so that you can choose to copy a file on some drive other than drive :0. This command requires swapping diskettes several times in order to utilize the DOS system modules to perform the transfer. The number of disk swaps can be kept to a minimum by having system modules 2, 3, [and 8 for Model I/III users] resident in memory (see the **SYSTEM (SYSRES)** library command).

You will be prompted for the correct diskette and when to insert it into the drive doing the copying. The prompts are as follows:

Insert DESTINATION disk in drive :d and press <ENTER>

This references the diskette to receive the file. You must have enough space left on that diskette to contain the entire file to be copied. Under certain conditions, this prompt may appear twice in a row.

Insert SOURCE disk in drive :d and press <ENTER>

This references the disk that contains the file to be copied.

Insert SYSTEM disk <ENTER>

This is any DOS SYSTEM diskette. If the diskette which is currently in drive :0 contains the complete system, just press <ENTER>. If the proper system modules (1, 2, 3, [and 8 for Model I/III users] and 10) are resident in memory, you may press <ENTER> without actually inserting a system disk.

The disk swap prompts will be repeated as many times as necessary until the copy is complete.

You cannot COPY (X) logical devices, only disk files. The disk files can be any type file made with any DOS compatible operating system. Note that the source and destination disks must have different pack IDs (disk name and/or master password or date).

After the COPY (X), the destination file's attribute will be visible, whether the source file initially was or not. If a password is entered on the command line, the destination file will automatically have this password set as its owner password. If no password is specified for the source or destination file, then the destination will have no password protection set. The correct attributes for the destination file may be re-applied with the ATTRIB command.

COPY23B/BAS

This DOS version 5 utility is provided to copy files from Model I TRSDOS 2.3B to a DOS disk. The syntax is:

```
BASIC RUN"COPY23B"
```

Since this is a BASIC program, BASIC/CMD be on a disk in the system. Also, Model III users will have to use the REPAIR utility on the 2.3B disk before using COPY23B.

When the program starts, you will be prompted to enter the source and destination file names. The source name is the name of the file on the 2.3B disk. The destination file is the name you wish to give the file on the DOS disk. These two names can be identical except for the drive number, and usually will be. If the file is password protected on the 2.3B disk, you must use the proper password in the source file name.

After both names are entered, the information will be read from the 2.3B disk and written to a new file on the DOS disk. When the copy is complete, the BASIC "Ready" prompt will appear. At this time, you may type **CMD"S"** to return to DOS Ready, or type in **RUN** to copy another file.

CREATE

This command allows for the creation of a file of the type and size that is requested by the parameters. The syntax is:

CREATE filespec (Fill=f,Lrl=l,Rec=r,Size=s,...)

Fill=f	The created file will be populated with this byte. If omitted, the file is not initialized with any byte.
Lrl=l	This is the Logical Record Length to be used. It must be an integer in the range 1 to 256 (default LRL=256).
Rec=r	This is the number of records of length LRL to be allocated to the file.
Size=s	This is the amount of space in K (1024 byte) blocks that the file is to be able to hold. SIZE may not be specified if LRL or REC are.
SHRINK	If specified, the file is marked for future deallocation of space.

CREATE is used to pre-create a file of a specified type and size. This allows the file to be as contiguous as possible on the disk and limits the number of disk accesses that must be performed when dealing with the file. This also assures that the expected amount of space on the disk will be available for use by this file. The file will be dynamically expanded if the created size is exceeded, but it will never decrease in size below its current size unless the SHRINK parameter is specified. A file cannot be created that would require more space than is available on a disk. Remember, if a drivespec is not used, the system will attempt to create the file on the first available drive.

The *FILL* parameter allows you to specify a particular character to propogate throughout the created file. *SHRINK* is used to flag the file for deallocation of unused space once data is written to the file. If SHRINK is not specified, the file's create flag is turned on in the directory.

Note: If a file has been created, a DIR command will show:

For a CREATEd file	S: nnK
For a normal file	S= nnK

The normal equal sign "=" will be replaced by a colon ":" to indicate that the file length is the result of a CREATE rather than the actual size of the data in the file.

CREATE NEWFILE/DAT:0 (LRL=128,REC=100)

This command will create a file named NEWFILE/DAT on drive :0. It will have enough space allocated to accommodate 100 records of 128 bytes each.

CREATE ASCII/DAT:2 (LRL=1,REC=5120)

This command will create a file named ASCII/DAT in which records will have a length of one byte, and there will be space taken on the disk for 5120 (5K) of these one byte records.

CREATE MAIL/DAT:3

This command will create the file MAIL/DAT on drive :3. There will be no space assigned to the file at this time. The file name is merely placed in the directory. This is very useful as it allows the placement of a yet-to-be-used file on a designated drive. Since it was not specified, the LRL of this file will be 256.

CREATE GOOD/DAT (REC=50)

This will create a file named GOOD/DAT on the first available drive. There will be space taken for 50 records of 256 bytes each, since the default LRL is 256.

CREATE SMALL/FIL:1 (SIZE=1)

This command will create a file SMALL/FIL on drive :1 and take 1,024 bytes of space for the file (in actuality one granule will be taken as this is the smallest unit of allocation the system can deal with).

It is acceptable to create a file larger than it already exists. This is the way that you would permanently assign additional space to a file. Once a file is created, any attempt to create it smaller will not be allowed. For example, a file named TEST/DAT was created with space for 100 records, and a LRL of 256. This file will take up approximately 25K of space on the disk.

As long as no records are written to the file, trying to create the file smaller will merely be ignored. Once records have been written to the file, any attempt to create the file smaller than the space used by the records will give a "File exists larger" error. For example, 50 records have been written to the file, and are using approximately 12.5K of space. Trying to create the file to a size of 10K will show the error message and the file will be untouched. Trying to create the file to some size between 12.5K and 25K will simply be ignored, as the 25K has already been set aside by the original create command. Creating the file to more than 25K will be allowed, and will permanently assign the extra space allocation to the file.

DATE

The DATE command is used to set the month, day, and year for use with your applications programs and by DOS as it creates and handles your disks. The acceptable range of dates is from January 1, 1980 through December 31, 2011; years 1980 through 1999 are entered as "80-99" while years 2000-2011 are entered as "00-11". The syntax is:

DATE (mm/dd/yy)

mm	is the 2 digit month, 01 to 12.
dd	is the 2 digit day, 01 to 31.
yy	is the 2 digit year, 80 to 99 [1980-1999] or 00-11 [2000-2011].

DATE with no parameter specified will display the current date.

It is important to set the date with DOS, because DOS uses the date when creating and accessing files, and when making backups and formatting disks. If the date is not set, DOS will make no date entries in the directory, when it would be useful to see the actual date. When looking at the directory you are able to see the *mod date* when a file was created or last written to. If the date was not set, this "last-written-to date" will not be updated, and the file would not show the true last-written-to date. When doing backups or formatting, DOS will use "00/00/00" if the date has not been set. Because the date is so important in the DOS system, you will be prompted for it on power-up.

Because DOS will generate the day of the week and day of the year, memory restrictions to hold this data limit the date to the range 01/01/80 - 12/31/[20]11. Entering a date outside of this range will not be permitted.

In the lower center of the screen at power-up the system will prompt for the date to be entered. You should answer this prompt with a valid date string. The prompt and the date you entered will then be erased and replaced by a display showing the day of the week in the standard three character abbreviation, the name of the month (also abbreviated), the day of the month, and the year expanded as "xxxx". If desired, the date prompt

can be removed from the boot sequence with the **SYSTEM (DATE=NO)** command. If you do disable the date prompt, the date will not be initialized although it may later be set.

It should be noted that DOS will store two other numbers calculated from the current date. These are day of the year, and day of the week. These values will be placed in memory, and will remain constant unless the date is reset. The date will stay set as long as the computer has power applied to it, providing the date storage area is not overwritten by user applications. Therefore when re-BOOTing after the date has been set, the system will automatically recover the date that was last set and will not bother with prompting for the date. The date that had been set will be displayed and the system will continue.

Should you wish to examine the date that is set in the system simply type **DATE** and press **<ENTER>**. If the date has been properly set, the system will return the currently set date in day-of-week, month, day-of-month, year format. The current date will also be sent to the Job Log, if active. If you have disabled the initial date prompt, the message "Date not in system" will be displayed.

Should you wish to set the date to one other than the system is currently using, simply enter: **DATE mm/dd/yy**. The new date will be set by the system. **DATE 01/04/95** sets the date for the first month (January), the 4th day and the 95th (1995) year.

On the Model I, the **SYSTEM (UPDATE)** command will allow the date to change when the system's real time clock rolls past midnight. Due to hardware restrictions, this is not possible on the Model III. Because the real time clock may be turned off during certain I/O functions (most notably during the BACKUP and FORMAT utilities, and sometimes during other disk I/O), the time and date may not remain accurate. If the computer is kept in a constant power on state from day to day, do not depend on the system clock for exact timing functions unless you have a hardware clock installed.

The SETDATE program may be provided to allow you to set or read a hardware clock. Invoke SETDATE with a command such as: **SETDATE MM/DD/YY**. Any additional instructions needed will be provided after SETDATE invokes.

DATECONV Utility

This command converts pre-x.3 version (non-system) disks to use the time and date directory format of DOS version x.3 or later. The syntax is:

DATECONV :d [(CS)]

:d	Designates the disk drive to convert.
CS	An optional version 6 parameter used to force conversion of some disks detected as system disks.

There are no parameters for this program. If conversion of a TRSDOS 6.2, LDOS 5.1, or earlier SYSTEM disk is attempted, an appropriate error message will be displayed. To convert this type of disk, you must first use BACKUP to move x.3 system files onto it before using DATECONV. For example,

BACKUP :s :d (S,I,OLD)

Data disks from versions earlier than x.3 will be converted in all cases and marked as an x.3 type disk.

DATECONV does not normally need to be used on an x.3 disk, since the COPY and BACKUP commands automatically adjust the time and date storage when moving from LDOS 5.1 or TRSDOS 6.x to x.3. For this reason you should **never** move files to an x.3 disk when booted up under any earlier version. If you do use DATECONV on an x.3 disk, only those files that have an old style blank userlaccess password will be changed. This is to protect the time stamp on proper x.3 files.

Note:
If you are using a hard disk drive which has its own formatting routine that does not support extended dating, you will have to DATECONV your hard drive partitions after you re-format them.

DEBUG

The DEBUG command turns the DOS system's debugging utility on or off. The syntax is:

DEBUG (ON|OFf,Ext)	
Ext	optionally turns on the extended debugger
ON|OFf	turns the ghost debugger ON or OFF. If not specified, ON is assumed.

Unlike the other library commands, the **DEBUG** command does not immediately produce a visible result. It loads the system debugger into memory and then waits to be activated. The extended debugger also loads a separate block into high memory, and protects this area by decrementing the HIGH$ value.

Once the debugger has been turned on, it will be entered when one of the following occurs:

- The <**BREAK**> key is pressed.
- After a program has been loaded, before the first instruction in the program is invoked, as long as the file's protection level is not execute only or higher.

The debugger may also be automatically activated by holding down the <D> key during the bootstrap operation.

The debugger will be disabled during the execution of any programs with an invoke only (or NO access) protection level.

Once the debugger is turned on, it will remain active until it is turned off, until an execute only program is invoked, or until the system is booted. When you wish to turn off the debugger and remove the extended debugger module from resident memory, use the command:

DEBUG (OFF,Ext)

The debugger will be de-activated; the extended debugger will be removed provided the memory-resident module was the most recent module installed into memory.

Detailed examples of interaction between the debugger and program modules will be given later in this section.

Once in the debugger, you can return to the DOS Ready prompt with the command **G402D<ENTER>**. From the extended debugger, the command **O<ENTER>** may be used instead. If you entered the debugger from Model I/III BASIC with a **CMD"D"**, a **G<ENTER>** will return you to BASIC.

Following is a sample display of the DOS version 5 debugger screen.

```
AF = 0D 2C --1-1P--1
BC = 0D 61 ->  79 9E 77 23 05 20 F9 C9  71 E5 D6 08 38 0E E1 E5
DE = 01 04 ->  1A 4D 45 4D 20 53 49 5A  45 00 52 2F 53 20 4C 32
HL = 00 54 ->  01 01 5B 1B 0A 00 08 18  09 19 20 20 0B 78 B1 20
AF'= 00 54 -Z-H-P--1
BC'= 51 B0 ->  29 29 29 29 B5 6F CD 8A  51 20 EF 1F CE 81 C9 D6
DE'= 06 01 ->  09 28 42 FE 19 28 39 FE  0A C0 D1 77 78 B7 28 CF
HL'= 51 00 ->  02 C7 C6 02 FF CB 02 F7  10 32 E7 20 32 01 C7 43
IX = 40 15 ->  01 9C 43 00 9A 00 4B 49  07 C2 FE 31 3E 20 44 4F
IY = 00 00 ->  F3 AF C3 74 06 C3 00 40  C3 00 40 E1 E9 C3 9F 06
SP = 41 CA ->  52 04 C3 4B DD 03 15 40  5D 45 18 43 3F 3F 4C 00
PC = 00 62 ->  B1 20 FB C9 31 00 06 3A  EC 37 3C FE 02 D2 00 00
     3E04 ->  20 34 30 20 31 35 20 3D  3E 20 20 30 31 20 39 03
     3E14 ->  20 34 33 20 30 30 20 39  01 20 30 30 20 34 02 20
     3E24 ->  34 39 20 20 30 37 20 03  32 20 06 05 20 33 31 20
     3E34 ->  33 05 20 32 30 20 34 34  20 34 06 20 09 19 20 3D
```

The debug display contains information about the Z-80 microprocessor registers. The DOS 6 version will simultaneously display the ASCII equivalents of displayable characters in the right hand portion of the screen. The display is set up in the following manner:

The register pairs are shown along the left side of the display, from top to bottom. The current contents of each register pair is shown immediately to the right of the register labels.

The AF and AF' pairs are followed by the current status of the flag registers to the right of the register contents. The other register pairs will be followed by the contents of the 16 bytes of memory they are pointing to.

The PC register will show the memory address of the next instruction to be invoked. The display to the right of that address shows the contents of that address to that address + X'0F'.

The bottom four lines of the screen show the contents of the memory locations indicated by the address at the left of each line. Refer to the list of debug commands for information on use of the debugger.

Note that in all examples, any parameter dealing with an address or a quantity must be entered as a hexadecimal number. The debugger will not accept the backspace key. If an incorrect command has been entered it may be cancelled by pressing the <X> key until the command line is cleared. Also, debug only looks at the last four numbers entered in response to any address question.

In the following examples, the first line following a command will give the syntax of the command. There are three ways the debug commands can be entered. The first requires the <ENTER> key be pressed. The second requires the <SPACE> bar be pressed. The third type is immediate and will invoke whenever the command key is pressed as the first character in the command. In the following commands listed alphabetically, the notation "[#]" indicates a Model I/III extended debugger command, while the notation "[$]" indicates a Model 4 extended debugger command.

Command: ; (Memory Page Advance)

The command syntax is: ;

This command advances the memory display 64 bytes in the register mode and 256 bytes in the full screen mode.

Command: - (Memory Page Decrement)

The command syntax is: -

This command decrements the memory display by 64 bytes in the register mode or by 256 bytes in the full screen mode.

Command: A (Alpha Mode or ASCII Modify)

The command syntax is: **A**

This command will set the display to the alphanumeric mode. Characters outside of the displayable range (X'20'-X'BF') will be displayed as periods. DOS version 6 ASCII entry is performed as noted for the **M** command.

Command: B (Move Block of Memory) [#]

The command syntax is: **Baaaa,bbbb,nnnn<ENTER>**

This command will move a block of memory from one location to another. ***aaaa*** is the starting address of the memory block to move; ***bbbb*** is the destination address of the memory block; ***nnnn*** is the number of bytes to move. Entering a 0 will move 65535 bytes, and will cause the extended debugger to function improperly.

Command: C (Call Address)

The command syntax is: **C**

This command will single step through the instructions pointed to by PC (the Program Counter). If any CALL instruction is encountered, the routine called will invoke in full. The destination of any jump instruction encountered must be in RAM memory, or the C command will be ignored.

Command: D (Display Address)

The command syntax is: **Daaaa<ENTER>**

This command starts the memory display from the address aaaa.

Command: E (Enter Data) [#][$]

The command syntax is: **Eaaaa<SPACE>**

This command allows you to enter data directly into memory, starting at address *aaaa*. The contents of memory address *aaaa* will be displayed and you may then type in two hexadecimal characters to replace the current contents. Pressing the SPACE BAR will then enter the new characters into memory. This operation will automatically advance to the next memory location, and allow you to continue entering characters until the <ENTER> or <X> key is pressed. If *aaaa* is omitted, the current memory modification address will be used.

Command: F (Fill Memory) [#]

The command syntax is: **Faaaa,bbbb,cc<ENTER>**

This command will fill a block of memory with a specified byte. The parameter *aaaa* is the first address to be filled; *bbbb* is the last address to be filled; and *cc* is the specified byte to fill the locations with.

Command: G (Go to an Address and Invoke)

The command syntax is: **Gaaaa<ENTER>**

Gaaaa,bkp1,bkp2<ENTER>

This command goes to a specified address and begins execution. The parameter *aaaa* designates a specified address. If it is omitted, the PC contents are used.

Bkp1 and *bkp2* are optional breakpoint addresses. They will cause execution to stop at the specified breakpoint. One or both may be specified. The breakpoints must be in RAM memory. All breakpoints are automatically removed whenever you return to debug. No more than two breakpoints are allowed at the same time. If you have entered DEBUG from Model I/III BASIC, typing **<G><ENTER>** will return you to BASIC. For Model I/III use, typing **<G402D><ENTER>** (or **<O><ENTER>** from the extended debugger) will return you to the DOS ready prompt. For Model 4 use, type **<O><ENTER> to exit the debugger.**

Command: H (Hex Mode or Hexadecimal Modify)

The command syntax is: **H**

This command sets the display to show hexadecimal format. This is the format used in the earlier debug example display. DOS version 6 hexadecimal entry is performed as noted for the **M** command.

Command: I (Single Step Execution)

The command syntax is: I

This command causes the debugger to invoke the command at PC and single-step to the next instruction. This command is identical to the C command except that any CALLS encountered will not automatically be invoked in full; they must be stepped through instruction by instruction. Any jump or call instruction encountered must have its destination in RAM, or the I command will be ignored.

Command: J (Jump Byte) [#]

The command syntax is: J

This command will Jump over the next byte, in effect incrementing PC by 1.

Command: L (Locate Byte) [#][$]

The command syntax is Laaaa,dd<ENTER>

This command will locate the first occurrence of the byte *dd*, starting the search at address *aaaa*. If *aaaa* is not specified, the current memory modification address will be used. If dd is not specified, the last byte given in a previous L command will be used.

Command: M (Memory Modify)

The command syntax is: Maaaa<SPACE>

This Model I/III command will allow modification of a specified memory address *aaaa*. If the display is set to include the specified address, you will see vertical bars around that byte of memory. The address and current byte will appear in the lower left of the screen. To modify the byte, enter in the desired characters. Pressing the <SPACE> bar will modify the byte and

move to the next address. Pressing the <ENTER> key will modify the byte and exit from the M command. Pressing the <X> key will exit from the M command without modifying the current byte. If *aaaa* is omitted, the current memory modification address (shown by the vertical bars) will be used.

Command: N (Next Load Block) [#][$]

The command syntax is: **Naaaa<ENTER>**

This command will position the vertical cursor bars to the next load block. This instruction is used to move logically through a block of memory that has been loaded directly from disk using DEBUG. To use this instruction you must position the location bars over the file type byte at the beginning of any block. Press **N<ENTER>** and DEBUG will advance to the next load block header. The position of this header is determined from the length byte of the load block.

Command: O (Return to DOS) [#]

The command syntax is: **O<ENTER>**

This command is the normal way to return to DOS Ready.

Command: P (Print Memory) [#][$]

The command syntax is: **Paaaa,bbbb<ENTER>**

This command will Print a block of memory from address aaaa to bbbb inclusively. The output will contain both HEX and ASCII formats in the following manner:

aaaa bb bb ... bb cccccccccccccc

aaaa represents the current line address; ***bb bb*** represents a line of 16 locations in HEX notation; and ***cccc*** represents the ASCII equivalents of the locations.

Command: Q (Query port) [#]

The command syntax is: **Qii<ENTER> or Qoo,dd<ENTER>**

The **Qii** command syntax will display the byte at the input port ii. The **qoo,dd** command syntax will send the data byte **dd** to output port **oo**.

Command: R (Register Modify)

The command syntax is: **Rrp dddd<ENTER>**

This command will modify the contents of a specified register pair. ***rp*** represents the register pair to be modified; ***dddd*** represents the new register contents. The contents of the registers can be seen while in the register display mode. The PC register may not be altered with this command.

Command: S (Full Screen Display Mode)

The command syntax is: **S**

This command changes the display format from the register display mode to the Full Screen mode. The full screen mode will display a page of memory (256 bytes) with the current display address being contained in the display (see the D command). The register pairs will not be shown.

Command: T (Type ASCII) [#][$]

The command syntax is: **Taaaa<SPACE>**

This command will allow you to type ASCII characters directly into memory, starting at address ***aaaa***. The current contents of the address will be shown, and the command will wait for the next keyboard character. After the character is entered, you will advance to the next memory location. To exit this command, use the **<ENTER>** key. The **<SPACE>** character cannot be entered with this command. Pressing the **<SPACE>** will advance one memory location without changing the contents of the current location. If ***aaaa*** is omitted, the current memory modification address will be used.

Command: U (Update Screen Display)

The command syntax is: **U**

This command will dynamically update the display, showing any active background tasks. It may be cancelled by holding down any key.

Command: V (Compare Memory Blocks) [#][$]

The command syntax is: **Vaaaa,bbbb,nn<ENTER>**

This command will compare the block of memory starting at ***aaaa*** to the block of memory at ***bbbb***. The compare will be for ***nn*** bytes. If the display is in the register mode, the first byte of memory displayed will be set to the first location in the block starting at ***aaaa*** which does not match the block at ***bbbb***. The current memory modification address used by the M, E, and T commands will be reset to the corresponding byte in the second block.

Command: W (Word Search) [#][$]

The command syntax is: **Waaaa,dddd<ENTER>**

This command will search memory for the WORD specified with ***dddd*** (entered in LSB, MSB format). The search will start at memory location ***aaaa***. If ***aaaa*** is not specified, the current memory modification address will be used. If ***dddd*** is not specified, the last word given in a previous W command will be used. The memory display will automatically be set to show the address where ***dddd*** was located, with the vertical bars one byte before the requested word.

Command: X (Cancel)

The command syntax is: **X**

This command will return the display to the normal register display mode.

Disk Read/Write Utility [#]

The command syntax is:
drive,cylinder,sector,op,address,count

op is the single character operation code: R for Read; W for Write; * for a Directory Write. ***address*** is the starting address in memory where the information read from the disk will be placed, or where information written to the disk will be taken from. ***count*** specifies the number of sectors to read or write.

CAUTION
Be sure to log in the disk to be accessed with either the LOG or DEVICE commands before using this option! If the disk will not log in properly, insert another disk of the same density and number of tracks, and log that disk instead.

If the cylinder is not specified, the directory track will be used. If the number of sectors is not specified, a full cylinder will be read. If the starting sector is not specified, sector 0 will be the default.

If an error is encountered during a disk function, the error number will appear on the screen, surrounded by asterisks. The error indication will repeat each time another error occurs. To abort the disk function, hold down the <ENTER> key.

Example: 0,,,R,7000

This example would read the directory track from the disk in drive :0. The information read would be stored in memory starting at location X'7000'.

Example: 0,,,*,7000

This command would write the directory track on drive :0, using the information stored in memory starting at location X'7000'. The "*" assures that the proper Data Address Mark will be written to the directory cylinder.

DEVICE

This command will display all logical devices which are in use and the devices and files to which they are currently pointing and/or attached to. It will also "log on" the diskettes currently in the available disk drives by updating the Drive Code Table to show the number of cylinders, sides, the density, and the location of the directory track. The syntax is:

DEVICE (Byteio=sw,Drives=sw,Status=sw,P=sw,N=sw)

Byteio=	Displays the "byte I/O" portion of the data. The default is ON [V5] OFF [V6].
Drives=	Displays the drive portion of the data. The default is ON.
N=	Set non-stop display mode. Assumed if P is selected.
P=	If ON, duplicates the display to the screen and the printer. The default is OFF.
Status=	Displays the options status portion of the data. The default is ON.

A typical device display might look like this:

```
:0   5" Floppy #1 Cyls= 40, Dden, Sides=1, Step= 6 ms, Dly=.5 s
:1WP 5" Floppy #2 Cyls= 35, Sden, Sides=1, Step= 6 ms, Dly=.5 s
:2   5" Floppy #4 Cyls= 80, Dden, Sides=2, Step= 6 ms, Dly=.5 s
:3   5" Rigid  #0 Cyls=153
*KI <-  X'FC11'
*DO <-> X'4DC2'
*PR <-> X'41E5'
*JL  -  Nil
*SI  -  Nil
*SO  -  Nil
*UD <-> TEXTFILE/TXT:1
Options: Type, KI, JKL
System modules resident: 2,3
```

For reference purposes, the display will be considered as having three parts. The first is the DRIVE section, showing the current configuration of the disk drives. Second is the BYTE I/O section, showing the devices (in this example, *KI through *UD). Last is the STATUS section, displaying the status of the user selected options.

The information on the type and configuration of each drive in the system is found at the top of the device display. There are several fields in each line which explain what the system sees for disk storage. Here is a typical line of information pertaining to one drive and the explanation of the fields it contains.

```
:1WP PACKNAME 5" FLOPPY #1 CYLS= 40, DDEN, SIDES=1, STEP= 6 MS, DLY= 1 S
---- --------- - ------ -- -------- ---- ------- ---------- --------
aabb [cccccc] d  eeeeee ff gggggggg hhhh iiiiiii jjjjjjjjjj kkkkkkkk
```

aa This is the logical drive number the line deals with.

bb This is the diskette write protect status, with WP = Write Protected. See the SYSTEM (DRIVE=,WP) command.

cc This is the disk pack name originally applied when the disk was formatted or subsequently changed with the ATTRIB command.

dd This is the size of the floppy or hard disk. 5.25" or 8" will appear in this field (3.5" is shown as 5.25").

ee This is the type of drive, floppy or rigid (hard) shown.

ff For floppy drives, this is the physical binary location of the drive on its cable. 1, 2, 4 or 8 will appear here. For hard drives, this may be the starting head number.

gg This is the number of cylinders on the disk that was in the drive when it was last accessed.

hh This shows the density of the last disk accessed in the drive and will show either DDEN or SDEN (double/single density).

ii This shows the number of sides on the last disk accessed by the drive and will be a 1 or a 2.

jj This shows the step rate in "ms" (milliseconds) for the drive.

kk This shows the delay time that will be imposed when accessing a 5.25" minifloppy drive. It refers to the time the system will wait after starting the drive motor before it attempts to access the disk. It does **not** refer to the time the drive will stay on after an access.

The following briefly describes what each of the *xx devspecs refer to. Each device will be shown as an asterisk followed by the 2 letter device name, its I/O symbol, and an address. The DEVICE command will use special symbols to show the I/O (input/output) paths for each device.

<= will indicate an input device.

=> will indicate an output device.

<=> will indicate a device capable of input and output.

*KI This device is the Keyboard Input which is controlled with the keyboard driver.

*DO This device is the Display Output (video).

*PR This device is the line printer, normally accessed with your computer's parallel printer port.

*JL This is the JOBLOG control device which is shown NIL on power up, and must be set to its driver (JL/DVR) to be used.

*SI This is the Standard Input device which will normally be pointed (NIL).

*SO This is the Standard Output and will normally be pointed (NIL).

***UD** This may be any user created device. It must be an asterisk followed by any two alpha characters. Setting up "phantom" or "real" devices is a very important part of the device independence of DOS (see the FILTER, LINK, ROUTE, and SET commands).

These device relationships and specifications will change from time to time depending on the use of the FILTER, ROUTE, SET, LINK, SYSTEM, and RESET library commands. Note that the *UD device shown in the display is a "phantom" device that has been routed to a disk file. A phantom device refers to a device specification that is not an actual piece of hardware - it is merely a way to link to another file or device. The filename of the file is shown after the device that is providing the input to that file. After you have routed or set a device and/or driver, you may use the DEVICE library command to see how the different devices have been affected. After any changes are made to the system, the DEVICE command will show the memory address where each of the hardware devices is going to enter the first driver or filter dealing with that device, as well as the interaction between devices and/or files.

Issuing the DEVICE command will also update disk information in the drive code table in the following manner. Each diskette in a currently enabled disk drive will be examined for number of cylinders, sides, density, and location of the directory cylinder. If a drive is enabled but contains no diskette, the device table entry for that drive will not change. The LOG utility program will perform the same function, but for a single drive only.

If the device command should "hang up" or return totally incorrect drive information, it is because the drive code table does not contain the proper size and location of your drives. The physical location and size (5.25" or 8") must be correct for the device command to log on the drives. If hard drives are enabled in your system, the number of heads and the starting head number for each logical drive must have been properly set with the appropriate driver program. Should you have a problem invoking the **DEVICE** command, reboot DOS with the **<CLEAR>** key held down. If using special hardware, use the **SYSTEM (DRIVE=,DRIVER)** command to set the proper drive configuration for your system. After assuring the drive information is correct with the device command, save this configuration with the **SYSGEN** or **SYSTEM (SYSGEN)** command.

The "Options:" line will show you the system options currently active. These options are usually established with the FILTER, LINK, ROUTE, SET, SPOOL, and SYSTEM library commands. These options vary with the system implementation, and can be:

Fast/Slow Indicates whether the computer is switched to a fast or slow speed.

Forms I PR Indicates that the forms filter is resident.

Graphic Indicates that the screen print function will pass graphic characters to the printer instead of translating them to periods.

JKL Indicates that the DOS version 5 screen-print module is resident.

KI Indicates that the DOS version 5 enhanced keyboard driver is resident.

KSM Indicates that the DOS version 6 KeyStroke Multiply filter is resident.

Memdisk Indicates that the DOS version 6 RAM disk is active.

MiniDOS Indicates that the DOS version 5 minidos filter is resident

Smooth Indicates that the floppy disk smooth option is active.

Spool Indicates that the system printer spooler is resident.

Type Indicates that keyboard type ahead is active.

Verify Indicates that the DOS version 6 read-after-write verify function is active for disk writes.

You will also see which system overlays are currently resident in high memory. See the **SYSTEM (SYSRES=)** command for details on how to reside these overlays.

DIR

This is the command which allows the examination of a disk directory. Several parameters are allowed to set the type of data that will be displayed. The syntax is:

DIR [-][partspec | filespec][:d1][-:d2] (parm,parm,...)

-	Exclude files which match.
partspec	Partial filespec with possible wild card characters.
:d1	Optional drive specification.
-:d2	Optional Model 4 drive designation to request a directory display of drives d1-d2 inclusive.
A=	View in full Allocation format, or in brief format.
Date=	Used to specify files within a range of dates. Selection format is: "M1/D1/Y1-M2/D2/Y2"; "M1/D1/Y1"; "-M1/D1/Y1"; and "M1/D1/Y1-".
Inv	Include view of the Invisible files.
Mod	View modified files ony.
N	Non-stop display mode (will not pause after each 15 lines). Assumed if "P" is selected.
P	Direct output to the Printer.
sOrt=	Specify sorted or unsorted view; YES is the default.
Sys	Include view of the System files.

DOS reserves a certain portion of every disk to keep information about the files and free space available on a disk. This space is called the disk's directory. The maximum number of files a directory can hold, as well as the available free space will be determined by the number of sides, the density, and the number of cylinders on the disk. DOS will always reserve certain portions of the directory space to store its own operating system

files. There are 16 file spaces in the directory of a SYSTEM disk that are reserved for system (/SYS) files used by DOS; only two file spaces are reserved on DATA disks.

The maximum storage on a disk is determined by two things - the amount of free space and the number of directory records. Reaching the maximum on either will prevent any more information from being written to the disk. It will be necessary to remove existing files before anything more can be written to that disk. Both the number of remaining files and the free space on a disk can be seen with the FREE library command. Files can be removed with either the REMOVE or KILL, or PURGE commands.

There will be three main classifications of files used when discussing a disk's directory; they are SYStem files, INVisible files, and VISible files.

System files contain the instructions used by DOS to perform most of its basic operations. They are identified by the extension /SYS. These files will not normally be seen when issuing a DIR command.

Invisible files can be any files that you do not wish to normally see with a DIR command; that's so your display is not cluttered. Most DOS utility files are set to invisible on your master disk. The ATTRIB command allows changing the visibility of a file. Visible files are those seen when doing a simple DIR command. These are usually your program and data files.

The reason for having methods of keeping files from being displayed by a simple DIR command is one of readability. It is much easier to find program and data files on a disk if you do not have to search through all the different system and utility filenames. Parameters are provided to allow all files on a disk to be displayed.

Directory parameters

The first parameter discussed will be the "***drivespec***". It is generally entered as a colon followed by the desired drive number. The command **DIR :0** would display the directory of logical drive :0, and **DIR :5** would do the same for drive :5. The command DIR with no drivespec would display the directories of all enabled drives. Specifying a drive that is not enabled will cause an "Illegal drive number" error message to appear. If you are doing a DIR command, you may omit the colon if you are not specifying a filespec or partspec. The DOS version 6 command **DIR 0**

would be the same as **DIR :0**. The parameters to include the system and invisible files are S and I. Visible files will always be included in any display. The command **DIR :0 (I)** would display all visible and invisible files on drive :0, the command **DIR :0 (S)** would display all visible and system files, and the command **DIR :0 (I,S)** would display all files.

For DOS version 6 operation, a drive range may be specified as follows:

d1-d2 displays the directory information for drives d1 through d2 inclusive.

d1- displays the directory information for drives d1 through the maximum assigned drive;

-d2 displays the directory information for drives 0 through d2 inclusive.

The directory display will normally show files sorted alphabetically in one column. The normal display will show a disk's directory in allocation format. The command **DIR :0 (I,S)**, which varies slightly according to machine implementation, may produce a typical display as follows:

```
Drive # diskname diskdate cyldh Free-fffff.f/sssss.s  Fi-mmm/nnn
Filespec      Attrib  LRL #Recs EOF DE File Size  MOD Date Time
filename/ext SIP*+ pp lrl rrrrr eof de s-sssss.s dd-mon-yy hh:mm

BACKUP/CMD    IP   EX 256    23 247  1 S-    6.2 26-Jul-91 17:47
BASIC/CMD     IP   EX 256    22  61  1 S-    5.0 26-Jul-91 17:48
BOOT/SYS     SIP   EX 256     5 255  1 S-    1.2
CMDFILE/CMD   IP + EX 256    12 114  1 S-    3.8 26-Jul-91 17:48
DIR/SYS      SIP   RE 256    18 255  1 S-    2.5
FORMAT/CMD    IP   EX 256    19 236  1 S-    5.0 26-Jul-91 17:47
KI/DVR        IP   EX 256     8  94  1 S-    1.2 10-Dec-91 16:02
------------ -------- --- ----- --- -- --------- ---------- -----
         xx files ouyt of yyy selected, Space - zzz.00K
```

As you can see from this display, every directory command shows more than just the filenames. The first line gives the drive number, the disk name and date, the disk configuration, the amount of free and total space on the disk, followed by the quantity of free and total files on the disk. Specifically, these fields are:

#	drive number
diskname	name of the disk
diskdate	date of the disk
cyl	number of cylinders on the disk
d	disk density: S=single, D=double, H=hard
h	Number of heads on the disk
fffff.f	Amount of free space in K
sssss.s	Total formatted space in K
mmm	Number of free file slots
nnn	Total number of file slots

The lines which contain filenames describe all information about a directory entry. These fields are described as follows:

filename This is the name field of a file.

ext This is the extension field of a file.

SIP*+ Attributes indicating: Sys, Inv, Prot, PaDS, and Mod. S indicates the file is a system file; I indicates the file has been declared invisible; P indicates the file has an OWNER password; * indicates that the file is either a Partitioned Data Set or a SubDISK file; + is called the "mod flag", and indicates the file has been modified since it was last backed up. The characters I, P, S *, and + may appear separately or in any combination to show the file's actual status.

C? DOS version 6 may display a "C" in the attributes field to indicate that a file was established with the CREATE library command. A question mark "?" will indicate that the file is either open or was left open by a program (see the RESET command).

pp This field shows the protection level of the file, and can be set or changed with the ATTRIB command. The protection level displayed could be: NO, EX, RE, UP, WR, NA, KI, FU; DOS version 6 operation would display a 4-character string.

lrl This is the length of each logical record in the file.

rnr This is the number of logical records in the file.

eof This is the position of the last byte in the last sector.

de This is the number of extents (non-contiguous blocks of space) in which the file is stored.

sssss.s This is the amount of space in K (1K = 1024 bytes) that the file takes up on the disk. Under DOS version 5, if the file has been created with the CREATE library command, the "S=" will appear as "S:".

dd-mon-yy This is the date that the file was created or last written to. If you have used the SYSTEM command to disable the initial DATE prompt when powering up the system, this date cannot be established or updated. DOS can use dates between 01/01/80 and 12/31/2011.

hh:mm This is the time that the file was created or last written to. The time will be in 24-hour format: 00:00-23:59. For Model 4 users, the time could be in 12 hour format, with AM or PM indicated depending on the setting of the **SYSTEM (AM|PM)** mode. Disks from earlier versions of DOS (prior to x.3) will display a blank time field.

If you do not wish to see all of the directory allocation information, you may abbreviate the display by use of the A parameter. A typical display of a DOS disk done with the command **DIR :0 (A=N)** may appear as follows:

```
Drive 0 diskname diskdate cyldh Free-fffff.f/sssss.s  Fi-mmm/nnn
KI/DVR P          KSM/FLT P          MINIDOS/FLT P
PATCH/CMD P       FDUBL/CMD          PR/FLT P
RS232R/DVR P
```

Including the S and I parameters will show all files in the directory. The command **DIR :0 (I,S,A=N)** may produce a display such as:

```
Drive # diskname diskdate cyldh Free-fffff.f/sssss.s  Fi-mmm/nnn
BACKUP/CMD IP        BOOT/SYS SIP         CMDFILE/CMD IP
DIR/SYS SIP          FORMAT/CMD IP        KI/DVR P
KSM/FLT P            LBASIC/CMD IP        LBASIC/OV1 IP
LBASIC/OV2 IP        LBASIC/OV3 IP        LCOMM/CMD IP
MINIDOS/FLT P        PATCH/CMD P          RDUBL/CMD
PR/FLT P             RS232R/DVR P         RS232T/DVR P
SYS0/SYS SIP         SYS1/SYS SIP         SYS10/SYS SIP
SYS11/SYS SIP        SYS2/SYS SIP         SYS3/SYS SIP
SYS4/SYS SIP         SYS5/SYS SIP         SYS6/SYS SIP
SYS7/SYS SIP         SYS8/SYS SIP         SYS9/SYS SIP
```

The directory information will normally appear on the video display (the *DO device). It will automatically pause when the display screen is full. Pressing <**BREAK**> will terminate the display, while pressing any other key will continue with screen. The display may be made to scroll without pause if the *N* parameter is specified in the DIR command. If the DIR command is invoked from a JCL file, the *N* parameter will automatically be set. You may use the <**SHIFT**><**@**> keys to pause the display.

The *P* parameter will send the display to the line printer (*PR device) as well as the video. The P parameter will automatically set the *N* parameter, and will print the entire directory without pause.

The **Mod** and **Date** parameters will allow you to see only those files that have been modified since their last backup, or fall within a specified range of dates. The DOS requires dates to be within the range 01/01/80 to 12/31/2011 [enter only the last two digits of a year]. The **Date** parameter accepts four formats to provide for selecting specific ranges of dates; note that the parameters are character strings and must be enclosed in quotes. These formats are:

- **DATE="M1/D1/Y1-M2/D2/Y2"** copies only those files whose mod dates fall between the two dates specified, inclusive.
- **DATE="M1/D1/Y1"** copies all files with mod dates equal to the specified date.
- **DATE="-M1/D1/Y1"** copies all files with mod dates less than or equal to the specified date.

- **DATE="M1/D1/Y1-"** copies all files with mod dates greater or equal to the specified date.

The directory display will normally be shown sorted in alphabetical order. To disable this feature, specify **SORT=NO** as a parameter when issuing a DIR command. The PURGE library command and the BACKUP utility access files in their unsorted order. You may see the same order of unsorted access by specifying the **SORT=NO** parameter in a DIR command.

Using filespecs and partspecs

Along with the previous parameters, DOS provides other methods for locating files in a disk directory. Three terms will be used when discussing these parameters - *"filespec"*, *"partspec"*, and *"wcc"* (WildCard Character). ***Filespec*** refers to a file's name and extension. For example, the filespec **BACKUP/CMD** has the filename *BACKUP* and the extension */CMD*. A ***partspec*** would be any part or parts of a filespec. *Wcc* means a special symbol (the dollar sign "$") used in place of characters in a filespec or partspec. For example, a command using a partspec is:

DIR /CMD:0

This would show only visible files with the extension /CMD on drive :0. You can always include any of the *A*, *I*, *N*, *P*, *S*, *DATE*, or *SORT* parameters whenever using any filespec, partspec, or wcc.

You may use a filename, a file extension, or both together in any DIR command. It is not necessary to use the complete name or extension. The wcc mask character ($) can be used to mask out certain groups of characters when using a filespec or partspec. Using a partial filename or extension provides the opposite function of using a wcc. Refer to the following:

Using a partial filename will display all files whose name starts with those characters, regardless of how many other characters follow. The commands:

DIR BA:0
DIR BACK:0

```
DIR BA/C
DIR BACK/CMD
```

would all display the file BACKUP/CMD, although any other files matching the partspecs would also be displayed.

The *wcc* mask ($) is used to mask out leading characters in a filename or extension. The commands:

```
DIR $$$$UP:0
DIR $$CK:0
DIR BACK/$$D:0
```

would again all display the file BACKUP/CMD, along with any other files that match the criteria. Using wcc's after a partspec will have no effect on the command. All files that meet the specified leading criteria will be displayed, regardless of the number of other characters in the filename or extension. A wcc may also be used in the middle of a partspec. For example, the commands:

```
DIR B$CK:0
DIR B$$$$P:0
DIR BA/C$D:0
```

would all display the file BACKUP/CMD, along with other matching files.

Using -filespecs and -partspecs

Entering the "***not***" symbol (the minus sign) in front of a filespec or partspec declares it to be a "***not filespec***" or "***not partspec***". The ***-specs*** are used to exclude files from a directory display. The same rules concerning filespecs, partspecs and the wcc mask apply exactly the same for -specs as for normal file and part specs. For example, the commands:

```
DIR -BA:0
DIR -B$$K:0
DIR -/CMD:0
DIR -/$$D:0
```

would show all files on drive :0 except for BACKUP/CMD, and any other files that match the -spec criteria.

DISKCOPY & QFB

This utility is designed to allow for a backup with format to be performed. Only floppy drives may be used (the source may be a MemDISK), and the backup performed must be mirror image. The syntax is:

DISKCOPY :s :d [V6]
QFB :s :d (parm,parm,...) [V5]

:s, :d	Is the Source and Destination drives. The colon is optional for DOS version 5.
All=sw	DOS 5 parameter used to specify whether all cylinders of the source disk will be read and copied to the destination disk, or only allocated cylinders will be used. The switch ON or OFF may be specified, with the default being OFF.
Query=sw	Query for parameters not specified. Switch ON or OFF may be used. The default is OFF
V1=sw	Parameter used to specify whether or not a verify of the destination disk is to be performed on the 1st pass. The switch ON or OFF may be used, with the default being ON.
V2=sw	Parameter used to specify whether or not a verify of the destination disk is to be performed on the 2nd pass. The switch ON or OFF may be used, with the default being OFF.

Both DISKCOPY and QFB perform a single-pass format and copy of a 3.5" or 5.25" double density floppy disk. They will duplicate both single and double sided disks. QFB will also duplicate single density and other media formats and sizes permitted by DOS version 5. Attempting to copy unsupported disk types will abort the program with an appropriate error message. The :s and :d are the *source* and *destination* drive numbers, respectively, and cannot be the same drive. After starting DISKCOPY or QFB, you will be prompted to insert the source and destination disks.

Consider the results of entering the following example command.

DISKCOPY :1 :2

Drive :1 will be used as the source drive, while drive :2 will be the destination drive. Prior to DISKCOPY performing any action, a prompt will appear to load the diskettes. Once the proper diskettes have been installed, press <ENTER>, and the backup will begin. The following actions will take place.

- The source diskette will be logged in, to determine the type of format.
- Cylinder 0 of the destination diskette will be formatted.
- If cylinder 0 of the source disk contains data, it will be read into memory.
- If cylinder 0 of the source diskette contains data, the information stored in memory (see third step) will be written out to the destination diskette.
- Cylinder 0 of the destination diskette will be verified.
- The previous steps will be repeated for all remaining cylinders.
- The following message, which may vary with the DOS release, will appear after the last cylinder has been verified:

```
Duplication complete     1 disk created
Replace destination disks and press <ENTER> to repeat
  ..<R> to restart with new parameters
    ...or....<BREAK> to exit program.
```

- Press <ENTER> in response to this prompt to make another mirror image backup. Press <BREAK> to abort the DISKCOPY utility. The following prompt may appear:

```
Load SYSTEM diskette and hit <ENTER>
```

Place a system diskette in drive :0 and press <ENTER>, to return to the DOS level.

If it is desired to use QFB again with different parameters, press <R> in response to the prompt. Doing so will cause the drives to be prompted for, and prompts will appear for all parameters.

If QFB is to be restarted, or the command **QFB (Q=Y)** is entered, the following prompts for the parameters will occur:

```
Duplicate unallocated tracks? (Y/N)
Verify on same pass? (Y/N)
Verify on second pass? (Y/N)
```

The first prompt relates to the ALL parameter. If it is answered with <Y>, all cylinders will be read from the source diskette and written to the destination diskette, regardless of whether or not the cylinder contains information. If this prompt is answered <N>, only cylinders containing information will be read and written.

The next prompt relates to the V1 parameter. If it is answered with <Y>, all cylinders on the destination diskette will be verified immediately after all writes. If answered <N>, no immediate verify will be done.

The final prompt corresponds to the V2 parameter. If it is answered with <Y>, all cylinders on the destination diskette will be verified upon completion of all writing to the diskette. If answered <N>, there will be no second pass verification.

These utilities perform no check on the destination diskette with respect to the existence of data. Any existing information on a destination diskette will **always** be destroyed. Also, the Mod Flags of files on the source diskette will **not** be cleared.

DO

The DO command invokes a user created JCL (Job Control Language) file. The syntax is:

DO char filespec[/JCL] (@LABEL,parm,parm...);	
char	is an optional DO control character "$", "=", "*".
filespec	is a valid filespec - default extension /JCL.
@LABEL	is an optional LABEL indicating a start point in the JCL file.
parm	optional parameter(s) to be passed to the filespec (JCL program) during execution.
;	optional semi-colon used to allow a DO command line greater than the maximum line length.

Note: Please refer to the *Batch Processing (Job Control Language)* section of the manual for the creation of a JCL file. PAGE 179

The DO command will compile and/or invoke a series of commands that have been created by the user and stored in an ASCII disk file. The default file extension of the filespec is /JCL. No line in a JCL file may exceed one less than the maximum command line length of the DOS; 63 characters in length for Model I/III; 79 characters in length for Model 4. The DO command will also pass optional parameters and variables to the program being done.

The DO function is normally a two step operation - the compile phase and the invoke phase. During the compile, a line is read from the specified file and then written to a file named SYSTEM/JCL. If this file does not exist, it will be placed on the first available drive which is not write protected. Once the line is compiled, it is then invoked directly from the SYSTEM/JCL file. There must be at least one available (enabled and not write protected) drive in the system to compile and invoke a JCL file. When generating the SYSTEM/JCL file, DO will automatically search for the first available non-write-protected drive to create the output. However,

an invoke only option is available with a DO control character, and will be explained later.

Please note that the occurrence of any error will terminate the DO execution. The **<BREAK>** key, if not disabled, will allow you to manually abort the DO.

The three control characters ("$", "=", "*") will change the compile and execution phases of the DO command. When using these characters, a space character is **mandatory** between the word "DO" and the character. If the space is omitted, the character will be ignored. Note that if no character is specified, both the compile and execution will be done.

The ***@LABEL*** parameter will allow you to create JCL files with multiple entry points. Each entry point can indicate a different location at which processing will begin. **Note:** If the @LABEL function is used, the compile phase must be done or the DO will abort with an error message. If the @LABEL parameter is specified, the JCL file will be scanned without execution up to the specified LABEL. Once the LABEL is reached, execution will begin and continue until the next LABEL, or until the end of the JCL procedure/file has been reached. The primary reason for the @LABEL parameter is to allow many different functions to be built into one large file. This will greatly conserve disk space, as a series of small JCL files would take up a minimum of one granule apiece. For complete definitions of JCL LABELs, refer to the JCL section of the manual.

If the @LABEL and parameters cause the DO command line to exceed the maximum length permissable, the semi-colon character (;) will allow you to continue passing parameters once the DO has started. The proper use of the semi-colon is as follows:

- Terminate the DO command line by enclosing as many parameters as you can in the parentheses. Close the parentheses, then insert the semi-colon character and press **<ENTER>**.

- A question mark will appear on the screen. At this point, you may enter the remaining parameters, making sure they are enclosed in parentheses.

Refer to the following examples and descriptions as a guide to the uses of the DO function.

Character: "$"

The "$" character will DO the compile phase only, without actually executing the commands. The DO will compile your JCL file to the SYSTEM/JCL file. This will test if the syntax of a new JCL file will compile properly. Use the LIST library command to examine the SYSTEM/JCL file to see the resultant JCL lines that will be invoked.

Character: "="

The "=" character will skip the compile phase and directly invoke your JCL file. Be aware that some of the JCL features will be ignored if the compile phase is skipped. Refer to the Job Control Language section of the manual for a complete list of these features and limitations.

Character: "*"

The "*" character will rerun the last DO command that was compiled, by using the existing SYSTEM/JCL file. If this file does not exist, nothing will be done and an error message will be generated.

DO DRIVE/JCL

This command will compile and invoke a file named DRIVE/JCL. The system will search the drives for a file named DRIVE/JCL and compile it to a file named SYSTEM/JCL. After it has been compiled, the resultant SYSTEM/JCL file will be invoked.

DO = DRIVE/JCL

This command will invoke the file DRIVE/JCL without compiling it to the SYSTEM/JCL file.

DO $ DRIVE

This command will compile the file DRIVE/JCL to the SYSTEM/JCL file. The file will not be invoked. Note that the filespec DRIVE will use the default extension of /JCL.

DO MY/JCL:0 (@THIRD)

This command will compile and invoke the program MY/JCL. All instructions in the program will be ignored up to the LABEL (@THIRD). Compilation will begin at the line following the label and will continue until the next LABEL or End of File is reached.

DO *

This command will invoke the SYSTEM/JCL file. If the file does not exist, an error will be generated.

DO TEST/NEW:2 (D=5,E=6)

This command will compile and invoke the file TEST/NEW on drive 2. The file will be compiled to the SYSTEM/JCL file and each line will be invoked from this file. The variables D=5 and E=6 will be passed as needed during the compilation.

The following examples show what will happen if the space is omitted in a DO command.

DO=TEST/JCL

The use of the = character normally tells the DO command to skip the compile phase and directly invoke each line of the JCL file. If the space between the DO command and the = is omitted, the compile phase will be done! This means that the TEST/JCL file will compile to the SYSTEM/JCL (creating the SYSTEM/JCL file if none exists).

DO$TEST/JCL

The "$" character normally tells the DO to compile the TEST/JCL file without executing it. If the space between "DO" and the "$" character is omitted, the execution will be done!

DO*

This command will ignore the asterisk (*) and generate the error message Filespec required!

DUMP

This command dumps a specified block of memory to a disk file. The dump may be in load module or ASCII format. The syntax is:

DUMP filespec (Start=a,End=a,Tra=a,Ascii,ETX=d)

filespec	is any valid filespec.
Ascii	is an optional parameter for an ASCII dump.
End=a	is the ending address of the memory block.
ETX=d	optional End of Text marker for ASCII dump.
Start=a	is the starting address of the memory block
Tra=a	is the transfer address or execution point.

The DUMP command writes an exact image of the specified memory locations to a disk file in load module or ASCII format; default file extensions are /CIM for non-ASCII dumps, and /TXT for ASCII dumps. The following restrictions are placed on the DUMP command addresses.

START=a The memory block must start above address X'5500' for Model I/III use or above address X'2FFF' for Model 4 use..

END=a The ending address must be greater than or equal to the starting address.

TRA=a The transfer address (TRA) may be any valid address. If not specified, the TRA will be a return to DOS.

Addresses may be entered in either decimal or hexadecimal format. Hex addresses must be in the form X'aaaa'.

The ASCII and ETX parameters are used to dump memory to a pure ASCII file. Address loading information is not present in the file and the file cannot be loaded by the system loader. The file is identical to the file

structure of most word processor systems such as SCRIPSIT. Following the last dump character, an End of Text (ETX) character is written. This character is normally an X'03', but may be changed with the ETX parameter to a character of your choice. For example, a Scripsit file will normally have an X'00' as the ETX character.

Here are some examples of using the DUMP command.

```
DUMP ROUTINE/CMD (S=X'7000',E=X'8000',T=X'7000')
DUMP ROUTINE/CMD (S=28672,E=32768,T=28672)
```

These two commands will create identical files. The first uses hex notation for the addresses, while the second is in decimal format. The result will be to writte the block of memory starting at X'7000' and ending at X'8000' to a disk file named ROUTINE/CMD. If the file already exists it will be overwritten. If it does not exist, it will be created on the first available drive. The transfer address of the program will be X'7000'.

```
DUMP TEST:1 (S=X'9000',E=X'BC0F')
```

This command will DUMP the specified block of memory to a disk file named TEST/CIM on drive 1. Since the file extension was not specified, it defaulted to /CIM. The transfer address was not specified and will be written to the file as a return to the system.

```
DUMP WORD/TXT:0 (S=X'7000',E=X'A000',A)
```

This command will dump the specified memory range to a disk file named WORD/TXT. Since the A (ASCII) option was specified, no load module information will be written to the file, and the EXT (End of Text) character will be the normal X'03'.

```
DUMP WORD:0 (S=X'7000',E=X'A000',ETX=X'FF',A)
```

This command is identical to the last one except that the End of Text marker will be written as an X'FF'. In addition, the file's extension was not specified and will default to /TXT.

FED

FED is a DOS version 5 sector-oriented file editor. Its wide range of capabilities make it excellent for the advanced user, but its simplicity makes it easy to use for the novice. The editor supports both Model I and III, upper and lower case, and all drive types and sizes supported by DOS.

FED is a file editor, not a file copier, text editor, or word processor. It is for displaying, printing, and modifying existing files. Fed works on a file level, not a track and sector level. FED was not designed to repair damaged disks or recover lost files, but it could be used to do so by the experienced DOS user. You cannot create or extend files with FED, only modify existing ones.

The following is a brief description of FED's capabilities:

- Complete editing capabilities are supported, including Hexadecimal and ASCII modifying. Direct disk patching is a simple matter; small changes in files can be made instantly.

- FED allows for record advancing, backspacing and positioning. You may page through a file quickly, either forward or backward.

- ASCII and Hex string searching can be performed. FED allows searching for upper or lower case ASCII strings and Hex strings.

- You can locate a Hex load address in a load module format file, and calculate the load position of a specified byte. FED can also calculate where in memory a specific byte pointed to by the cursor will load.

- Complete listing of a file or individual record to a printer is supported.

- FED has both a 256 byte display mode and an extended 128 byte display. By having a 128 character display mode, the extra space makes it more visually appealing. The filespec, drivespec, record number, input and output can be displayed horizontally instead of vertically.

Here is a sample display of the 256 byte mode:

```
 ASCII representation   Hexadecimal representation    Current Record

!.h..=.UX.S @..!.00> 21D8 6811 003D CD55 58ED 5320 4006 1721.0.F
.`.@...zx. .>..3.10> ED60 CD40 00DA 945A 78B7 2005 3E13 C333.0.E - filespec
`. X..`..D .!.b..20> 60CD 2058 11ED 60CD 1C44 20F0 210D 6206.0.D
..$D.3`:.`.0*.Rw.30> 00CD 2444 C233 603A F360 C630 2A15 5277.D./
.K.`..C.R....C.R.40> ED4B F960 0BED 4313 5201 0000 ED43 0F52...C
!.a".R>..3..UU.:.50> 210D 6122 0A52 3E1C CD33 00CD 5555 C93A.  M - Ext
.R...E....Z..`.B.60> 0152 B7CC 105A FE04 D410 5A11 ED60 CD42.  D
D.3`..`.6D.3`!.b.70> 44C2 3360 11ED 60CD 3644 C233 6021 0D62.  :
..a.......[..[U..80> 110D 6101 0001 EDB0 CDE3 5BC9 CD28 55CD.  5 - Drive #
.]..K.R...`..BD..90> DE5D C9ED 4B0F 52C5 11ED 60D5 CD42 44C2.
3`.6D.3`!.b..a...A0> 3360 CD36 44C2 3360 210D 6211 0D61 0100.
.....*.R..B...BD.B0> 01ED B0D1 C12A 1352 B7ED 42C8 03CD 4244.
.3`.6...UD.3`.G:.C0> C233 60CD 3601 028E 5544 C233 60C9 473A.
.R.!.!.?" @.!.-6.D0> 0E52 B728 0721 CA3F 2220 40C9 21BD 3D36.    - Relative
.#6..@..6...6.+6.E0> 8C23 36AC 1140 0019 36AA 10FB 3683 2B36.>82 - Byte
.!.-" @.:.R..!.-.F0> 8321 FD3D 2220 40C9 3A0E 52B7 C021 BD3D.C:R

              Index                                        Command
```

Here is a sample display of the 128 byte mode:

```
.K.`..C.R....C.R!.a".R>..3..UU.:.R...Z....Z..`.BD.3`..`.6D.3`!.b   - ASCII
..a.......[..[U..]..K.R...`..BD.3`.6D.3`!.b..a.......*.R..B...BD   - Rep.

          00 01 02 03 04 05 06 07   08 09 0A 0B 0C 0D 0E 0F
          ................................................
     40>  ED 4B F9 60 0B ED 43 13   52 01 00 00 ED 43 0F 52        - Hex
     50>  21 0D 61 22 0A 52 3E 1C   CD 33 00 CD 55 55 C9 3A        - Rep.
     60>  01 52 B7 CC 10 5A FE 04   D4 10 5A 11 ED 60 CD 42
     70>  44 C2 33 60 11 ED 60 CD   36 44 C2 33 60 21 0D 62
     80>  11 0D 61 01 00 01 ED B0   CD E3 5B C9 CD 28 55 CD
     90>  DE 5D C9 ED 4B 0F 52 C5   11 ED 60 D5 CD 42 44 C2
     A0>  33 60 CD 36 44 C2 33 60   21 0D 62 11 0D 61 01 00
     B0>  01 ED B0 D1 C1 2A 13 52   B7 ED 42 C8 03 CD 42 44
          ................................................
FED/CMD   Drive 5   Record 13  X'000D'         Relative Byte >82
Command:R                                     Values X'61'=97
```

Entering FED

To enter FED, simply type **FED <ENTER>** at the DOS Ready prompt. The first prompt you will see will ask you to enter a filespec. Answer this prompt by giving the filespec of the file you wish to examine or modify. If you wish to exit FED at this point, press the **<BREAK>** key. If an illegal or improper filespec is given, the appropriate error message will appear, and you will be allowed to re-enter the filespec. The filespec prompt may be bypassed by entering FED using the syntax:

FED filespec<ENTER>

After a valid filespec has been given, the FED 256 character mode will appear on the screen, and the first record (record 0) will be contained in the "edit buffer" (The term "edit buffer" will refer to the record of the file currently in the computer's memory. The edit buffer will contain one 256 byte record at any given time). There will be two cursors flashing within the record; one cursor will be in the "ASCII" portion of the screen, the other cursor will be in the "Hex" display portion. Upon initially accessing a file, these cursors will be positioned over relative byte X'00' of record X'0000'. Throughout this documentation, the term "relative byte" will be used, and will indicate the byte number (0-255) relative to the sector in question. Also, hexadecimal notation (X'nn') will be used to represent the current record number and relative byte number.

There will also be an input cursor located on the bottom right portion of the screen, following the message "Command". This will be referred to as the "*command buffer*", and will be the place on the screen where commands are entered. The current command in use will always be displayed. When in the 128 character mode, the command buffer will appear on the lower left portion of the screen.

Also shown on the screen will be additional information such as current record number, filespec, relative byte within the sector, etc. The previous screen illustrations show where this information will be displayed. For certain commands, inputs of several characters will be required.

Depending on the mode you are in (256 or 128 character mode), these inputs will be taken in a different manner. When in the 256 character mode, these types of inputs will be taken in an input box, and the input box will be positioned vertically along the right hand edge of the display. When in the 128 character mode, these types of inputs will be taken directly to the right of the command buffer. No input box will appear, but a flashing cursor will be present, indicating that an input is requested.

It is advised that when using FED, the **<BREAK>** key should always remain enabled, as some FED commands are exitted by the use of the **<BREAK>** key.

	FED Commands
<A>	Enter ASCII character modify mode
<B>	Position to the Beginning record
<C> cccccc	ASCII Character string search for cccccc
<D>	Dump Disk File to printer (from current position)
<E>	Position to the Ending record
<F> nnnnnn	Find Hex string nnnnnn
<G>	Go to the next occurrence of last search (Hex or ASCII)
<H>	Enter Hex modify mode
<L> nnnn	Locate Hex load address nnnn
<M>	Memory location of a specified byte
<N><ENTER>	New File request (open a different file)
<O>	Output a top-of-form to printer (X'0C')
<P>	Print current record in edit buffer
<R> nnnn	Position to Record nnnn
<S><ENTER>	Save current record (sector) in edit buffer
<T>	Toggle between 256 and 128 display mode
<X><ENTER>	eXit FED and return to LDOS Ready
<Z>	"Zip" through File Load Blocks
<BREAK>	Cancel current FED command
<ENTER>	Display FED instruction set (Menu)
<;> <+>	Advance one record in the file
<->	Backup one record in the file
<SHIFT><=>	Display binary representation of byte (128 byte mode only)
	FED Cursor Movement
<⇐>	Move cursor left.
<⇒>	Move cursor right.
<⇑>	Move cursor up.
<⇓>	Move cursor down.
<SHIFT><⇑>	Position cursor to relative byte X'00' of the current record.

```
                MENU DISPLAY OF FED INSTRUCTION SET

<;>   Forward ONE Record          <BREAK>     Cancels command
<->   Backward ONE Record         <N><ENTER>  New File
<B>   Beginning Record of File    <S><ENTER>  Save Record
<E>   Ending Record of File       <X><ENTER>  Exit FED
<R>   Position to Record          <H>    Hexadecimal Modify
<Z>   Go to next Load Block       <A>    ASCII Modify
<M>   Calculate Load Address      <T>    Toggle Display modes

<C>   Find ASCII String           <F>    Find Hex string
<L>   Locate Hex Load Address     <G>    Go next occurrence

<D>   Dump File to Printer        <O>    Output top-of-form
<P>   Send Buffer to Printer      <=>    Display Binary Value

           Press <ENTER> to Return to Display Mode
```

FED Manipulation Commands

<;>

Advances one record sequentially in the file. For example, if FED was currently displaying record X'000C' and <;> was pressed, the contents of record X'000D' would be displayed (provided that a record X'000D' existed in the file). An "*" will be displayed directly below the record number when pointing to the last record in the file. Issuing the <;> command will not change the position of the relative byte cursors. A "+" will be shown in the command buffer to show positive motion in the file.

<->

Backs up one record in the file. If FED was currently displaying record X'0087' and <-> was pressed, the contents of record X'0086' would be displayed. Issuing the <-> command does not change the position of the relative byte cursors. The <-> command will be ignored if it is issued when record 0 is being displayed. A "-" will be shown in the command buffer to show negative motion in the file.

<B>

Positions to the beginning of the file (record X'0000') and points cursors to relative byte X'00'.

<E>

Positions to the ending record of the file. An "*" will appear directly below the record number, indicating that the record being displayed is the last record in the file. The relative byte cursors will be positioned on the last byte in the file (not necessarily relative byte X'FF'). Since DOS uses sector I/O, the whole sector will be displayed, and any byte in the sector may be modified. Realize that any modifications made to bytes beyond the last byte will not cause the EOF marker of the file to be updated to reflect these changes.

<R>nnnn

Positions to record X'nnnn', provided record X'nnnn' exists in the file. If the record does not exist, an "*" will appear in the command buffer. After entering **<R>**, a box will appear below the record number display box. The input for the record number to retrieve will be taken in this box. Hex digits (0-F) must be entered, as any other characters will be ignored. You may press **<BREAK>** to cancel this command. The user may enter the record number without using the standard four digit (X'nnnn') format. Simply type in the record number and press **<ENTER>**. For example, if the desired record number is X'0021', type **<R> <2> <1> <ENTER>**. To position to record X'0007', type **<R> <7> <ENTER>**. The position of the relative byte cursors will remain unchanged after the new record is retrieved.

<Z>

Points the cursors to the next "Type" byte (X'01', X'02', X'05', X'07', X'10', X'1F') of a Load Module File. This feature is designed to allow the user to ZIP through machine language files quickly. Place the cursors on a "Type" byte and press **<Z>**. After this has been done, the cursors will be positioned over the next "Type" byte. Encountering a X'02' will terminate a **<Z>**ip. Any string searching, address locating, or address calculating will disable an active **<Z>**ip.

FED Modification Commands

<A>

Enters the ASCII Modify Mode. In this mode, modifications can be made in ASCII. Anything you can type in from the keyboard (with the exceptions of the **<BREAK>** key and the arrow keys) can be sent to the

edit buffer. Modifications can be made by positioning the cursor over the bytes to be changed. After the A command is issued, the command buffer will display an "A". From this point on, any characters entered will be taken as modifications to the bytes in the record. The arrow keys may be used to position the cursor for additional edits. To exit the ASCII modify mode, the <BREAK> key must be pressed.

To modify a byte:

- Position the cursor to the desired byte to change.
- Type in the ASCII character to replace the original.

After making a modification, the relative byte cursors will move to the next byte of the record. Note - no changes are made to disk, only to the edit buffer. To make changes to disk, see the *Save* command.

<H>

Enter the Hex Modify Mode. In this mode, the user can modify bytes in the currently displayed record. Modifications can be made by positioning the cursor over the bytes to be changed. After the "H" command is issued, the command buffer will display an "H". From this point on, any characters entered will be taken as modifications to the bytes in the record. The arrow keys may be used to position the cursor for additional edits. To exit the Hex modify mode, the <BREAK> key must be pressed.

To modify a byte: 1) Position the cursor over the desired relative byte in the record. 2) Enter the hex digits that you wish to overwrite the current information with. As digits are entered, the previous hex digits will be replaced by the digits entered from the keyboard. The first hex digit entered will modify the first hex digit in the byte, and the second hex digit entered will modify the second hex digit in the byte. After an entire byte has been modified, the cursors will move to the next byte in the record. Note - no changes are made to the disk, only to the edit buffer. To make changes to disk, see *Save*.

<S><ENTER>

Save the contents of the current edit buffer to disk. The current record pointed to by FED will be overwritten by the contents of the edit buffer.

Any changes made after the initial read of the record will be written to disk.

FED Search Commands

Note: The search commands described below may cause the information in the edit buffer to be overwritten by information contained in subsequent records of the file. If edits have been made to the information in the edit buffer, they should be saved to the disk prior to issuing a search command. In most cases, you should issue a "B" command prior to performing a search. This will assure that the entire file will be searched, and no occurrences of the search string will be missed.

<C>ccc

Finds ASCII string "**cccccc**". Issuing the <C> command will cause a search to be performed for the string (cccccc). The search will start at the relative byte pointed to by the cursors. The search is identical to the <F>ind Hex string command, except that the search criteria is an ASCII string of 1 to 30 characters (depending on the display mode being used). Also, the number of characters to be searched for may be an even or an odd number. See the <F> command for further information.

<F>nnn

Finds hex string "nn nn nn". The <F> command will perform a search for the hex string nn nn nn, starting at the relative byte pointed to by the cursors. (If in the 256 byte display mode, the length of the hex string may be from 2 to 6 characters long, and must be represented as an even number of characters. If in the 128 byte display mode, the length of the hex string may be from 2 to 30 characters long, and must be represented as an even number of characters). The search will begin from the byte over which the cursor is positioned, and will scan all records past the current record until the first occurrence of the string is encountered. If a match is found, the record containing the match will be displayed, and the cursors will be positioned over the first character of the record which matches the search string. To terminate any search, you may press the <BREAK> key. This will cause the record which was contained in the edit buffer prior to the search to be read back in from the disk. If a match is not found, an "*" will appear in the command buffer, and the cursor will be positioned over relative byte X'FF' of the last record. Only hex bytes can be entered, not hex digits. An "*" will appear in the command buffer if an odd number of

hex digits are entered. If there are multiple occurrences of the specified string, you can "go" to each occurrence by means of the <G>o command.

<G>

Goes to the next occurrence of current search criteria (string or "L" address). The <G>o command performs a continuation of the last search. If the last search was for a string, it will go to the next occurrence of that string. If the last search was for an "address", it will <G>o to the next occurrence of that address. Note that <G>o works in conjunction with the last search! If the data searched for is not found, one of two things will happen. If the <G>o command is issued after an <L> command and the address is not located, the current record will be read in from disk, and the position of the relative cursors will be unaffected. If the <G>o command is issued after any other search command and the search criteria is not located, the last record will be displayed with the cursor pointing at relative byte X'FF'.

<L>nnnn

The <L> command allows the user to find load address X'nnnn' in a load module file. The *Locate* command starts its search at record X'0000', rather than at the current cursor position. If the address is located, the record containing the byte at that load address will be displayed, and the cursors will be positioned over this byte. If the address is not located, an error message will be displayed, and you will be prompted to press <ENTER> to continue. After <ENTER> is pressed, the record which was in the edit buffer prior to issuing the <L> command will be retrieved, and the position of the cursors will be unaffected. If a *Locate* is performed on a non-load module file, the appropriate error message will be displayed. The *Go* command may also be used in conjunction with the <L> command to locate multiple occurrences of the same load address.

FED Output Commands

<D>

List the file to the printer, in the same format as the *Print* command. The <D> command will print all records in the file, starting from the current record number; all records will be read in from the disk. To halt the printing prior to its completion, depress the <BREAK> key. After the printing has been completed (or terminated), the record which was in the

edit buffer prior to printing will be retrieved from disk and stored in the edit buffer, and the cursor position will remain unaffected. Realize that if changes have been made to the record in the edit buffer, these changes should be saved to the disk prior to issuing the <D> command.

If the printer goes off-line during printing, FED will continue printing after the printer has been re-enabled. Please note that the DOS spooler will work in conjunction with the printing operations of FED. Also note that all records will be printed in 20 lines, with a spacing of 2 lines between records. This will allow 3 records to be printed on 66 line/page paper.

<O>

This command will output a top-of- form character (X'0C') to the printer.

<P>

The **<P>** command will print the contents of the edit buffer in ASCII and Hex. After the **<P>** command has been issued, the record display on the screen will be sent to the printer. To terminate printing at any time, depress the **<BREAK>** key. See the sample output produced by the <P> command on page 148.

FED Miscellaneous Commands

<ENTER>

Displays FED instruction menu.

<X><ENTER>

Exits FED and returns to DOS Ready.

<N><ENTER>

Opens a New file for editing. A prompt for the filespec will be displayed. If you input an invalid or improper filespec, an error message will appear, and you will be allowed to re-enter the filespec. Note that FED will never close files, as files need not be closed with this type of editor.

<BREAK>

Clears the command buffer. Pressing <BREAK> will cancel any partial command, and will cause the termination of any command being invoked. It is also the only way to exit the ASCII and Hex modify modes. Anytime there is any doubt as to the operation being performed by FED, you may press <BREAK>, and the command buffer will be cleared.

<SHIFT><=>

Displays the binary representation of the byte pointed to by the cursors. This command may only be used when in the 128 character mode, and will be ignored if issued in the 256 character mode. After depressing <SHIFT><=>, 8 binary digits will be displayed next to the command buffer. For example, if the cursors were positioned over relative byte X'27', and this byte of the edit buffer contained a X'F3', the binary digits 1111,0011 would be displayed.

<M>

Calculates the address in memory where the byte pointed to by the cursors will load. This command works with load module format files only. If the byte is contained in a load block, the load address will be displayed below the record number. If the byte is not in a load block (e .g. a comment line, file header, etc.) the error message "Byte not in load block" will be displayed.

<T>

Toggles between the regular 256 byte mode and the extended 128 byte mode. By pressing <T>, FED shifts to "the other" mode. The 128 character mode has all of the same commands as the 256 character mode. The display is a window of the 256 byte record, and 128 bytes will be displayed. By moving the cursors (usually with the <⇑> and <⇓> arrows) you will notice a scrolling effect. The ASCII display will be at the top of the screen instead of the 16 leftmost columns. The current record number is displayed in decimal as well as hexadecimal. All inputs will be taken horizontally instead of vertically. ASCII and hex search inputs will allow 30 characters instead of 6.

SAMPLE <P> COMMAND OUTPUT

```
SPACE/CMD DRIVE 1 RECORD 22 X'0016'

0123456789ABCDEF BYTE 00 01 02 03 04 A5 06 07  08 09 0A 0B 0C 0D 0E 0F
-----------------------------------------------------------------------
<.2<.:..<2....D. <00> 3C 09 32 3C 7F 3A 04 7F  3C 32 04 7F C3 F6 44 A5
GAME OVER PLAYER <10> 47 41 4D 45 20 4F 56 45  52 20 50 4C 41 59 45 52
 < >NEW HIGH SCO <20> 20 3C 20 3E 4E 45 51 20  48 49 47 48 20 53 43 4F
REEN....TER NAME <30> 52 45 45 4E 01 00 B4 97  54 45 52 20 4E 41 4D 45
               !. <40> 20 20 20 20 20 20 20 20  20 20 20 20 20 20 21 C4
W:`...Ww:a...Ww: <50> 57 3A 60 7F CD F1 57 77  3A 61 7F CD F7 51 77 3A
a...Ww:b...Ww:b. <60> 61 7F CD Et 51 11 3A 62  7F CD F7 57 11 3A 62 7F
..Ww....0#.././. <70> CD F1 57 77 C9 E6 0F C6  30 23 C9 CB 2F CB 2F CB
 /./...0#. PLAYE <80> 2F CB 2F 18 F0 C6 30 23  C9 20 20 50 4C 41 59 45
R < >  ......... <90> 52 20 3C 20 3E 20 20 FF  FF FF FF FF FF FF FF FF
................ <A0> FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF
................ <B0> FF FF FF FF 80 88 B7 B7  B7 B7 9D 80 AE BB BB BB
......     .... INTRU <C0> BB 84 80 80 80 80 80 80  80 80 20 49 4E 54 52 55
DERS............ <D0> 44 45 52 53 AE 9D AE 9D  88 9B A7 84 88 9E AD 84
....0xH.x.0|H0.H <E0> A0 99 A6 90 30 78 48 B4  78 84 30 7C 48 30 F8 48
POINTS20 POINTS1 <F0> 50 4F 49 4E 54 53 32 30  20 50 4F 49 4E 54 53 31
```

FILTER

The FILTER command establishes a program to filter (modify) the I/O path of a specified device. The syntax is:

FILTER devspec [USING] filespec (parm,...)	[V5]
FILTER devspec [USING] devspec2	[V6]

devspec any valid DOS device.

devspec2 the device spec of an installed DOS 6 filter module.

filespec the filespec of a DOS version 5 FILTER program, with the default extension being /FLT.

parm optional parameters for the filespec program.

The FILTER command is used to filter or modify data as it passes between the specified device and its driver program. DOS is structured so that any device may be easily filtered to provide modification of standard I/O paths.

You will find that filter programs are usually written to provide other than "standard" functions for available devices. This ability is provided since the standard device driver programs may not meet your particular needs. DOS provides different filter programs for different devices.

A filter program can provide many useful functions during I/O processing. Lines and/or characters could be counted, with certain actions taking place when pre-set limits are reached. Character conversions could be performed, such as simply changing each line feed to a null, or a complete conversion from ASCII (normal TRS-80 character set) to EBCDIC (IBM character set). Keyboard entries may be intercepted and acted upon, as is done in the KSM and MiniDOS filters.

This Model I/III example shows the use of the FILTER command with the DOS forms filter program PR/FLT (the Printer filter).

```
FILTER *PR USING PR/FLT (CHARS=80,INDENT=6)
FILTER *PR PR (C=80,I=6)
```

These two filter commands will produce identical results. Note that the use of the word "USING" is optional. Also, the default extension for the filespec is /FLT. This example will filter I/O directed to the line printer through the PR/FLT program, described in the *DEVICE DRIVERS and FILTERS* section of the manual. As a result of this filter routine, printed output will be limited to 80 characters per physical line. Also, any single line which is greater than 80 characters in length will wrap around, and be indented 6 spaces on the next line. **Note:** These parameters are determined totally by the PR/FLT program, not the FILTER command.

DOS version 5 users may use the following equivalent format:

```
FILTER *PR USING PR/FLT
FORMS (C=80,I=6)
```

An equivalent format usable for DOS version 6 would be the following:

```
SET *FF USING FORMS/FLT
FILTER *PR *FF
FORMS (C=80,I=6)
```

Another filter routine provided on your DOS diskette is called KSM/FLT. It provides the KeyStroke Multiply feature of DOS.

```
FILTER *KI USING KSM/FLT USING MYKEYS/KSM
FILTER *KI KSM MYKEYS
```

The above DOS version 5 examples would produce identical results, and are illustrations of how to establish a KSM filter program. The DOS version 6 equivalent would be:

```
SET *KS KSM MYKEYS
FILTER *KI *KS
```

The KSM feature will now be enabled, and would use the file MYKEYS/KSM to provide the KSM phrases. From the example, you should see that the filtered device would be *KI (the keyboard), and the filter program used would be called KSM/FLT.

FILTER *CL REMLF

This command would filter the *CL device's I/O using the filter routine found in a user developed filter program called REMLF/FLT. For example, if *CL had been set with a SERIAL driver, this command would filter I/O to and from the RS-232 interface. From the name of the program it may be assumed that the filter program may do something such as removing a linefeed after a carriage return.

FORMAT

This command formats a diskette with cylinders (tracks), sectors, and a directory, so that it may be used by the system. The syntax is:

FORMAT [:d (parm,parm,...)]

Abs	Will format the disk even if the disk is already formatted and contains data.
Cyl=	The number of cylinders (tracks) that are to be placed on the disk, up to 96.
Dir=	Specifies a cylinder for the directory which will be used instead of an automatic selection.
Mpw="s"	The master password assigned to the disk.
Name="s"	The name that will be given to the disk.
Query	Will prompt you for density, sides, step, and number of cylinders.
SDEN **DDEN**	The density that will be used to FORMAT the disk, DDEN (double) or SDEN (single).
SIDES=d	The number of sides to be formatted, either 1 or 2.
STEP=d	The boot track step rate that will be put on track 0, either 0,1,2,3.
SYSTEM	Will add system information to a previously formatted hard disk. Also used in creating Model I dual-density system BOOT disks.
Wait	Adds a delay time after track stepping.

FORMAT is the program that will create the proper information on a diskette so the DOS system can read and write to that diskette. A disk to be formatted may be blank, or it may have already been formatted. Note that if the FORMAT command is to be used in a JCL file, the disk to be formatted must be blank unless the ABS parameter is specified.

FORMAT generates a DATA disk after formatting. DATA disks reserve only two file slots out of the total number of directory slots available. SYSTEM disks, configured by the BACKUP utility, reserve 14 additional directory slots for /SYS files.

For DOS version 5 Model I systems, the SYSTEM parameter is used for 5.25" floppy drives to designate cylinder 0 as single-density when the diskette is specified for double density formating. This creates a dual-density configuration suitable for subsequent use as a booting system disk once the operating system has been moved and the SOLE utility applied. The SYSTEM parameter can also be used with hard disks to add system directory information to a previously low-level formatted hard drive.

Typing in the format command with no parameters will prompt you for them in the following order. If the drivespec, disk name or master password were specified on the command line, their prompts will not appear. Additionally, entering any one of the remaining **xDEN, SIDES, CYL**, or **STEP** parameters will cause format to use the defaults for the other parameters, and you will not be prompted for them.

```
Which drive is to be used ?
Diskette name ?
Master password ?
Single or double density <S,D> ?
Enter number of sides <1,2> ?
Number of cylinders ?
Boot strap step rate <6, 12, 20, 30/40> ?
```

If you are formatting in drive :0, the following prompt will appear after you have answered the step rate question:

```
Load DESTINATION disk and hit <ENTER>
```

The first prompt will ask for the drive number to use. If you are going to format a disk in drive :0, do not remove the system disk and insert the disk to be formatted until prompted to do so.

The next prompts after the disk number will be for the disk name and master password. These two pieces of information are used by several of the DOS library commands and utilities. They will be referred to as the ***Pack ID*** throughout the manual. You will be allowed up to eight characters for either entry. Characters used for the password must be either alphabetic or numeric. Using any other characters will cause an error, and the format will abort. Pressing only **<ENTER>** will use the default values.

The density prompt will always appear on the Model III and DOS version 6. It will not appear when using a Model I unless you are using a double density controller board and the FDUBL driver program. Pressing **<ENTER>** in response to this prompt will use the default density value.

The side prompt must be answered **<1>** or **<2>** for single or double sided. If you are using double sided, you must have the proper hardware setup. Pressing **<ENTER>** will default to 1 side. Typically, if your hardware does not support 2-sided drives, all tracks will be locked out during verification. **Most TRS-80 floppy drive cables do not support 2-sided floppy drives.**

The cylinder prompt will not appear for 8" drives, as they always have 77 cylinders. For 3.5" or 5.25" drives, any number up to 96 may be entered. Pressing **<ENTER>** will use the default cylinder value.

The bootstrap step rate is important only if you will be using the disk in drive :0 to boot up the system. Be aware that too fast (lower value) a step rate may keep the disk from booting.

Before the actual formatting begins, the target disk will be checked to see if has been previously formatted. If it has, the following message will appear:

```
Disk contains data -- Name=diskname  Date=mm/dd/yy
Are you sure you want to format it ?
```

If the disk contains an incomplete or non-standard format, one of the following messages may appear in place of the "NAME=diskname".

Unreadable directory
Non-standard format
Non-initialized directory

You will see the disk's name and date, and can abort the format at this point. Press <N> to abort the format, or <Y> to continue. If you have specified the ABS parameter, you will see this message but will not be prompted to abort the format.

FORMAT parameter default values

The **NAME** and **MPW** parameters may be specified in the command line followed by the desired string enclosed in parentheses. If either parameter is specified without being followed by a string, you will be prompted for it before the formatting begins.

Parameters not passed in the format command line will default as follows:

- **NAME** will default to "~~DOSDISK~~". "DATADISK"
- **MPW** will default to "PASSWORD".
- **DENSITY** will use different default values depending on the hardware. Model III and MAX-80 will default to double density. Model I Radio Shack interface will default to double density if there is a doubler board and driver program in use. Otherwise, it will default to single density. DOS version 6 defaults to DDEN.
- **SIDES** will default to 1 side.
- CYLinders will default to the value set with the **SYSTEM (DRIVE=,CYL=)** command and stored in the system information sectors on drive :0. If no value has been set, the default will be 40 cylinders for the Model III and 4, 35 cylinders for Model I 5.25" drives, and 77 cylinders for Model 8" drives.
- **STEP** rate will default to the value set with the **SYSTEM (BSTEP=)** command, also stored in the system information sectors. If no value has been set, the defaults will be 30/40 ms on the Model I, and 6 ms on the Model III and 4. These values represent the step

rates in milliseconds: 5" 0=6ms, 1=12ms, 2=20ms, 3=30/40ms; 8" 0=3ms, 1=6 ms, 2=10ms, 3=15/20ms.

The **QUERY** parameter defaults to **YES**, and you are normally prompted to enter all parameters. If you are sure that the default values will produce the exact format you desire, you may specify the parameter **Q=N** to bypass all parameter prompts.

The **ABS** parameter is primarily useful when FORMAT is invoked from a JCL file. As explained in the JCL section, an unexpected prompt from an executing program can cause the JCL processing to abort. Using the ABS parameter assures that there will be no prompt from the FORMAT utility to abort the formatting if the target disk already is formatted.

FORMAT cylinder verification

When the formatting begins, you will see the cylinder numbers appear as the necessary information is written to them. After all cylinders are written, format will verify that the proper information is actually on each cylinder. If the verify procedure detects an error, an asterisk and the cylinder number will be shown on the video display. This space will be locked out, so that no files will be written to the defective area. The **FREE** command provides a method to see the locked out tracks on a diskette.

The WAIT parameter

This parameter is not normally used when formatting. It can compensate for hardware incompatibilities when using certain types of disk drives. The only time it should be used is when all tracks above a certain point are locked out when verifying. To use this parameter, specify:

WAIT=nnnn

The value for **nnnn** will normally be a number between 5000 and 50000. The exact value can vary depending on the particular disk drive. It is recommended that a value around 25000 be used at first. This value can be adjusted higher if tracks are still locked out, or lower until the bottom limit is determined.

FORMS

This command allows you to display or alter the operating parameters of the forms filter (PR/FLT for Model I/III; FORMS/FLT for Model 4) once the filter has been installed. The syntax of the FORMS command is:

FORMS [(parms)]

Default	Sets all parms to defaults.
Addlf	Adds a line feed after a carriage return.
Chars=n	The number of characters per printed line.
Ffhard	Use 12D for form feed, rather than line feeds.
Indent=n	Indents n spaces on lines longer than chars [0].
Lines=n	The number of lines printed per page [66].
Margin=n	Sets the left margin [0].
Page=n	Sets physical page length [66].
Query	A Model 4 parameter which when used will have FORMS prompt you for each parameter entry.
Tab	Expands tab characters to tab stops [OFF].
XLATE=X'aabb'	Translates aa to bb [00 00].

If the *ADDLF* parameter is specified, a linefeed will be issued after every carriage return.

The *CHARS* parameter sets the number of characters that will be printed on each line. It may be any integer between 1 and 255.

If the ***FFHARD*** parameter is specified, any form feed determined by the PAGE and LINES parameters will be sent as an X'0C' character rather than a series of linefeeds. If you use this parameter, be sure your printer will recognize the X'0C' character.

INDENT sets the number of spaces a line is to be indented if the line length exceeds (CHARS=) characters. The default value for this parameter is zero (0).

LINES sets the number of lines that will be printed on each page. It may not exceed the PAGE parameter, and if not specified, it will default to the PAGE parameter of 66.

The *MARGIN* parameter sets the width of the left margin. It is especially useful for printers with fixed position tractors, such that you want printing to actually begin some distance from the printers physical left margin.

PAGE sets the physical page size in lines. It should be set to the particular form size you are printing on (66 for normal printer paper, 6 for mailing labels, etc.). The default value is 66 lines per page.

If the *TAB* parameter is specified, any X'09' character will be expanded to a standard 8 column tab.

The *XLATE* parameter will translate a specified character to another character. The format is **X'aabb'**, where ***aa*** is the character to be translated, and ***bb*** is desired character result. Both aa and bb must be hexadecimal values. This parameter may be useful to translate printer control characters when using more than one type of printer on the same system.

FORMS (C=80,I=6,P=51,L=45,F)

This command will change the resident FORMS filter's operating parameters to the following:

(CHARS=80) will allow a maximum of 80 characters per printed line. If a line contains more than 80 characters, the excess will be printed on the next line(s).

(INDENT=6) will indent 6 spaces the remainder(s) of any line that exceeds 80 characters (determined by the CHARS=80 parameter).

(PAGE=51) sets the physical page size to 51 lines.

(LINES=45) will allow for 45 lines to be printed on a page. Since the page length is 51 lines (determined by the PAGE=51 parameter), the filter program will normally send six linefeeds after the 45th line has been printed. These linefeeds are determined by the formula (PAGE minus LINES). If no linefeeds are required, do not specify either PAGE or LINES.

(FFHARD) will cause an X'0C' to be sent rather than 6 linefeeds when the line count reaches 45.

```
FORMS (M=10,C=80,I=6)
```

This example will cause all lines to start 10 spaces in from the normal left-hand starting position (MARGIN=10). Any line longer than 80 characters will be indented six spaces when wrapped around, and will be printed starting at position 16.

```
FORMS (TAB,ADDLF)
```

This example will cause expansion of all X'09' characters to their normal eight-column tab position. Also, a line feed will be sent every time a carriage return is sent.

```
FORMS (XLATE=X'2A2E')
```

This example will translate all X'2A' characters (asterisks) to an X'2E' characters (periods). This may be useful to change the appearance of a report format.

FREE

This command will show the used and available space and files on each disk in the system, or display a space map of a disk drive. The syntax is:

FREE [:d] (P)

:d	An optional drivespec, specifying a free space map of a specified floppy drive.
P	An optional parameter that directs output to the printer as well as the video display.

To invoke the free command simply type **FREE** at the DOS Ready prompt. The display format of the FREE command will display the identical disk configuration information as displayed by the header of the DIR command. DOS will respond with a display similar to this, which may vary slightly in appearance depending on the system (the top line is shown here strictly for illustration only):

```
Drive # diskname diskdate cyldh Free-fffff.f/sssss.s  Fi-mmm/nnn
Drive 0 DOS531   11/15/91  40D1 Free-   87.0/  180 0  Fi- 97/128
Drive 1 DATADISK 06/01/91  35S1 Free-   19.0/   88 0  Fi- 39/ 64
------- -------- -------- ----- ------------ -------      --- ---
aaaaaaa bbbbbbbb cccccccc ddddd      eeeeeee fffffff     ggg hhh
```

The first line gives the drive number, the disk name and date, the disk configuration, the amount of free and total space on the disk, followed by the quantity of free and total files on the disk. Specifically, the information given in each of the fields is:

aa This # field shows the drive number that the rest of the information in the line will pertain to.

bb This **diskname** field shows the name of the disk, established with the FORMAT utility program.

cc This **diskdate** field shows the date of creation or the date of the last Mirror Image backup to the diskette.

dd This field shows the disk configuration: **cyl** = the number of cylinders on the disk; **d** = the disk density: S=single, D=double, H=hard; and **h** = the number of heads on the disk

ee This **fffff.f** field shows the amount of free space in "K" (1024 byte blocks) that remains available for use on the disk.

ff This **sssss.s** field shows the total amount of space the disk will support in "K".

gg This **mmm** field shows the number of directory entries that are available for use (number of files that may be added).

hh This **nnn** field shows the total number of directory entries that the disk will support.

The FREE command without a drivespec is global in its nature. It will search all active drives, and may not be confined to a single drive. The free space available on a diskette is also shown in a DIR library command.

Using the FREE command with a drivespec will bring up a free space map as shown below.

FREE :0

```
Drive 0 DOS531  11/15/91  40D1 Free=   87.0/  180 0  Fi= 97/128
-------------------------------------------------------------------
  0- 5   x..      xxx      xx.      ...      xxx      xxx
  6- 11  xxx      ...      xx.      ...      ***      ***
 12- 17  ...      ...      ...      ...      ...      ...
 18- 23  x..      ...      DDD      xxx      xxx      ...
 24- 29  ...      ...      ...      ...      ...      ...
 30- 35  ...      ...      ...      ...      xx.      ...
 36- 39  ...      ...      ...      ...
```

The top line will display information about the diskette size and type; this information is in the same format as the global FREE command. DOS version 6 will display some of this following the granule mapping.

The inner display area contains the details of the space allocation on the disk. The numbers on the left represent the cylinders. The granules per cylinder will be shown across each line, with six cylinders per line. This disk has three granules per cylinder, as it is a 5.25" double density disk. The granules per cylinder will vary according to diskette size, density, and number of sides.

A granule will be represented as one of four characters:

"."	(period) will represent an unused gran.
"*"	(asterisk) will represent a locked out gran (floppy disks only).
"X"	will represent a used gran.
"D"	will represent a gran used for the Directory.

If the drive has more cylinders than can be displayed on a single screen, you must press <**ENTER**> to advance to the next screen. Also, if the screen completely fills with the granule information, the system will pause; you must press <**ENTER**> to return to the DOS Ready prompt. This will prevent the top line of the display from scrolling off the screen.

This display may also be sent to *PR (your printer) by using the (P) parameter.

HELP Utility

This command displays help information. It uses the DOS/HLP file to present information about DOS commands. Information is retrieved by entering commands of the form:

HELP [filespec[/HLP] [*][keyword] [(P,V,R,S)]]

filespec	Is the name of an available help data file.
*	Designates a global search of all on-line /HLP files for the specified keyword.
keyword	Is the name of one help file entry; if omitted, the list of entries is displayed.
B	Cancels the blink feature when displaying information on the screen [V5].
P	Sends the help display information simultaneously to the printer.
R	Cancels the use of reverse video when displaying information on the screen [V6].
S	Causes the search mode to be entered.
V	Cancels the automatic video screen restoration.

The DOS/HLP file provides on-line help information for DOS commands; DOS version 5 also includes a BASIC/HLP file which contains information on disk BASIC statements and functions. The help screen is composed of:

- Brief description
- Command Syntax
- Parameter descriptions

The HELP displays use the following notations:

[ ]	Optional arguments or parameters; note that some Model I systems will display these characters as ↑ and ⇒
|	Alternative selection: "ON|OFF" means "ON" or "OFF".
< >	Key combination you press.

For example, to obtain HELP about the DOS LIB command, at DOS Ready type:

HELP DOS LIB <ENTER>

This will now access the file called DOS/HLP, and display the information filed under the keyword "LIB". The video display will remain until any character generating key is depressed. If there is more information about the keyword than would fit on one screen, pressing a key will cause more information to display and, if necessary, pause repeatedly until the information is exhausted. At that time, the video display is restored and control will be returned to DOS.

To inspect all the keywords contained within a file, type:

HELP filespec <ENTER>

This will list all of the keywords within the named file. If the previous example had been "**HELP DOS <ENTER>**", a list of the available keywords would have been displayed. Once again, the display will pause if necessary. After each screen full of keywords, the prompt "<ENTER>, <BREAK>, or type keyword?" will appear. At this time, if **<ENTER>** is pressed, the next screen full of keywords will be displayed. If **<BREAK>** is pressed, HELP will abort, and control will return to DOS. A keyword may be entered, and the information relating to that keyword will be displayed.

If the specified keyword was not in the called file, the list of all keywords would display again, to indicate what was available within that file.

To list all of the help files presently available on the system, merely type:

HELP <ENTER>

This will search all drives on line for files ending in the /HLP extension, and list them to the video display. For example,

```
Help Categories presently on line are:

DOS/HLP:1 BASIC/HLP:1

Press ENTER to exit or enter category
```

The function of the HELP command may be altered by specifying one or more of the following optional parameters: P, V, B, R, or S.

The *P* parameter sends the output to the *PR device as well as the video. While using this option, the display will not pause if filled. Since all characters are being sent to the printer, no pause is required.

The *V* parameter causes the video restoration feature to be cancelled. If not specified, the screen will be returned to the same condition as it was when HELP was invoked, less the help command itself,

The DOS version 5 *B* parameter causes the blink feature to be cancelled. Various characters can be made to flash in the video display by specifying them as blinking characters during creation of the data file. However, if HELP were invoked while in a communications mode, a continuous stream of characters would be sent from the host machine to the terminal. The B parameter alleviates this difficulty.

The DOS version 6 *R* parameter causes the reverse video option to be cancelled. Various phrases can be displayed in a reverse video mode if so specified in the creation of the data file. However, certain terminals utilize the characters involved and unpredictable results can occur while in the communications mode. The *R* parameter alleviates this difficulty.

The *S* parameter causes the Search mode to be entered. Typing:

```
HELP DOS D (S)
```

would cause a listing of all keywords starting with "D" to be displayed rather than the entire list. The potential match should be the left most characters of a keyword. By specifying "DI", all keywords starting with "DI" would be displayed.

HELP also allows a "global" scan for any on-line keyword. If the keyword "MEMORY" was known, but the file is unknown (or to save typing in the filename), then enter an asterisk "*" followed by the keyword in the help command line. For example,

```
HELP *MEMORY
```

would find the first occurrence of the keyword "MEMORY" in any /HLP file. The top of the screen displays the category being scanned while a global search is in progress. If the key is found, the text displays normally. At the end of the text, the prompt "Press <BREAK> to exit or <ENTER> to continue global scan" appears. Pressing **<ENTER>** will look for the same keyword in another file until all /HLP files have been examined. Upon completion of the scan, or if no match is found, the normal prompt for category selection will appear. Continue as desired by pressing **<BREAK>** or **<ENTER>** to return to DOS Ready or by typing in a category name to obtain the directory for that file.

Besides use at DOS Ready, the same command sequence may be employed within DOS version 5 BASIC by utilizing the CMD"exp" function. Your application could be written to invoke help as an operator choice from a menu or command line. For example, **CMD"HELP DOS FILESPEC"** might be invoked by the application program if it detected an invalid filespec entry by the operator. HELP requires about 5K of free memory to function. All system memory guides are followed, and the HELP system will abort if sufficient memory is not available.

Another example of a HELP request from DOS version 5 BASIC might be: **CMD"HELP BASIC LSET"** which will function (memory permitting) as described above, and would return control to BASIC.

HITAPE

The HITAPE utility is for DOS version 5 Model III only, and will permit the use of high speed (1500 baud) cassette I/O in the BASIC and CMDFILE programs. The syntax is:

```
HITAPE
```

Due to space constraints and a desire to provide a high level of sophistication through the proper use of interrupt tasks, it was necessary to disable the use of 1500 baud cassette loading in the resident DOS system. To restore that 1500 baud tape capability in the system when it is needed, the HITAPE/CMD utility adds an additional routine to the DOS. The utility is called and is invoked by simply typing **HITAPE <ENTER>** at the DOS Ready prompt. You may then use 500 or 1500 baud tapes in the normal manner.

If HITAPE is in when a **SYSTEM (SYSGEN)** is performed, it will be saved with the configuration file. Both CMDFILE and BASIC allow the use of high speed cassette only if HITAPE has been invoked. If HITAPE has NOT been invoked and a 1500 baud tape load is attempted, the tape will not load. It may be necessary to depress the **<BREAK>** key to regain control of the system.

KILL and REMOVE

KILL is a DOS version 5 command used to delete one or more specified files or devices from the system. The syntax is:

```
KILL filespec | *devspec [filespec | *devspec]      [V5]
REMOVE filespec | *devspec [filespec | *devspec] [V6]
```

filespec The name of the file to be deleted.

devspec The name of the device to be deleted.

See *REMOVE Library Command* for further information.

LIB

This command will display the DOS command libraries. The syntax is:

```
LIB
```

After invocation of this command, the DOS command libraries would be displayed as shown below, with the exact set of commands shown varying according to the DOS version:

```
Library <A>
 Append      Cat       Cls       Copy
 Device      Dir       Do        Filter
 Kill        Lib       Link      List
 Load        Memory    Remove    Rename
 Reset       Route     Run       Set
 Tof         Spool

Library <B>
 Attrib      Auto      Boot      Build
 CLOCK       Create    Date      Debug
 Dump        Free      Purge     Time
 Verify

Library <C>
 Forms       Setcom    Setki     Spool
 Sysgen      System
```

Library <A> is the primary DOS command library, and is located in the SYS6/SYS system module. Library <B> is the secondary command library, and is located in the SYS7/SYS system module. You may delete either system module containing the libraries if the commands included in it will not be used. The Version 6 DOS provides the machine specific Library <C> commands in a third library located in SYS8/SYS.

LINK

This command joins together multiple logical I/O devices. The syntax is:

> **LINK devspec1 [TO] devspec2**
>
> **devspec** is any currently enabled logical device.

This command is used to join together two logical devices. Both devices must be currently enabled. Once linked, any output sent to devspec1 will also be sent to devspec2. Any input requested from devspec1 may also be supplied by devspec2.

The user is cautioned about making multiple links to the same device(s), as it is possible to create endless loops and lock up the system.

The order of the devices in the link command line is important, since output to devspec2 will not be sent to devspec1, nor can input requested from devspec2 be supplied by devspec1. Also, using the ROUTE library command on devspec1 will destroy its link to devspec2, but routing devspec2 is perfectly acceptable.

Once linked, devices can be un-linked by the command "**RESET devspec**". A global RESET or a reboot will also un-link devices. See the RESET and BOOT commands for further information.

Following are some examples of the use of LINK.

LINK *DO *PR

This command will link the video display to the line printer. All output sent to the display (devspec1) will also be sent to the line printer (devspec2). Once linked, the line printer must be enabled if it is physically hooked to the system (i.e. the cable is connected to both the printer connector and the printer). If the printer becomes de-selected or faults (out of paper, etc.) the system may lock up depending on the use of the forms filter and the setting of the **SYSTEM (PRTIME)** mode. Remember that both linked devices must be enabled. Note that any output sent individually to

the printer, such as a PRINT from Basic, will not be shown on the video display.

```
LINK *PR *DO
```

This command will link the line printer to the video display. All output sent to the printer will also be sent to the video display. The line printer must be on line and enabled if any printing is to be done. This link will not send any characters from the video to the line printer.

Although files may not be directly linked to a device, it is still possible to accomplish the same results through the use of "phantom" devices. Follow this two-step procedure to accomplish a devspec to filespec link:

- Use the ROUTE command to create a "phantom" device and route it to the file;
- then use the LINK command to link the device to the phantom device.

CAUTION

Do not use the **SYSTEM (SYSGEN)** command if you linked a device to a file. The file will be shown as being open every time you power up or boot the system. You could very easily overwrite other files if you happened to switch disks with the file open.

This example will show how to link your line printer to a disk file named PRINT/TXT on your drive :0 diskette. First, a *phantom* device must be created. For this example we will create a device named *DU. To do this, use the ROUTE command in the following manner:

```
ROUTE *DU TO PRINT/TXT:0
```

This will create a device named *DU and ROUTE it to a disk file named PRINT/TXT on drive :0. If the file does not exist, it will be created and dynamically expanded as needed. If the file already exists, any data sent to the file will be appended onto the end of the file since the REWIND parameter was not specified.

Next, the printer can now be linked to the file with the following command:

```
LINK *PR *DU
```

The printer is now linked to the device *DU, which in turn is routed to the disk file PRINT/TXT. All output sent to the line printer will also be sent to the device *DU (in effect, written to the disk file PRINT/TXT).

Please note that the file PRINT/TXT will remain open until a **RESET *DU** is done. If you wish to break the link between the printer and the file without closing the file, then use the command **RESET *PR**. For further information, please refer to the ROUTE and RESET commands.

LIST

The LIST command will send a listing of a file to the video display or line printer. The syntax is:

LIST filespec[/TXT] (parm,parm,...)

Ascii8	Will allow full 8-bit output.
C=OFF	Turn off compressed display mode for hexadecimal listing output. [V5].
Hex	Sets hexadecimal output format.
LINE=d	LINE in text file where ASCII list is to begin.
Lrl=d	The Logical Record Length to be used to display a file when in the hex mode.
Ns=ON	Sets non-stop display mode; assumed if "P" is selected.
NUM	Sets line numbering mode for ASCII text.
Tab=n	Set tab stops and expansion for an ASCII listing (n=1-32) [8].
P	Directs output to the line printer.
Rec=	Record number where hex list is to begin.

If no parameters are specified, the LIST command will list the file in ASCII format, and the logical record length (LRL) of the file will be read from the directory. Normal ASCII format will strip the high bit from each character, in effect displaying only those characters in the range X'00' to X'7F'. The ASCII8 parameter may be used to see all characters, including graphics characters.

The display output defaults to paged display mode; 23 lines [V6] or 15 lines [V5] of ASCII information (16 lines of hexadecimal) will be displayed, then the listing will pause. **<ENTER>** will resume the paged display, or type **<C>** to continue in non-stop mode . You may specify **LIST ... (NS)** to force the listing to display in non-stop mode. NS may be abbreviated to "N". If you select the (P) output mode (i.e. output to a printer), (N) is automatically specified.

The DOS version 5 hexadecimal display mode produces a combined hexadecimal and ASCII display of a complete 256-byte record via a compressed display mode. The separate display format is available by turning off the "compressed" display mode via the "*C=OFF*" parameter.

The parameters shown may be entered in the same command line, such as

LIST TESTFILE:0 (HEX,REC=5,LRL=80,P).

If an extension is not used in the filespec, a default of /TXT will be used. If no file with the /TXT extension is found, LIST will search for a file with an extension of all blanks.

Here are some examples of how LIST handles the "default" file extension of /TXT.

LIST TESTFILE:0

The system will first search drive :0 for a file named TESTFILE/TXT. If not found, it will then search for a file named TESTFILE.

LIST TESTFILE

The system will search all active drives for a file called TESTFILE/TXT, and list the first file named TESTFILE/TXT it encounters. If this file is not found, it will search all active drives for a file named TESTFILE, again listing the first TESTFILE it encounters.

LIST TESTFILE/SCR

The system will search all drives for a file called TESTFILE/SCR and list the first file named TESTFILE/SCR encountered. If the file is not found, the LIST command will not search for TESTFILE/TXT.

The parameters of the LIST command will determine the output format of the information in the listed file. Refer to the following section for a complete explanation and the proper use of these parameters. Note that the *NUM* and *LINE* parameters are for ASCII listings only and will be ignored if the *HEX* parameter is specified.

Parameter: ASCII8

This parameter will allow all 8 bits of each character to be displayed during an ASCII list. Normally, any character above X'7F' (decimal 127) will have the high bit reset. The *ASCII8* parameter will be useful if you wish to see graphics characters in a file.

Parameter: NUM

NUM will number the lines of the file as they are sent to the video display or printer during an ASCII list. Line numbers will start with one (1) and be in the format 00001. Lines are determined by the occurrence of a carriage return. Line feed characters will not generate a new line number.

Parameter: LINE

The *LINE* parameter is used with ASCII files. It will start the listing with the specified line of the file. Lines are determined by the occurrence of a carriage return character in the file. An example of the proper syntax would be **LIST TESTFILE/TXT (LINE=14)**. This would list the ASCII file TESTFILE/TXT, starting with the line of the file after the 13th carriage return.

Important: The *NUM* and *LINE* parameters will always be ignored if the *HEX* parameter is specified.

Parameter: HEX

The *HEX* parameter will cause the file to be listed in the following format.

```
aaaa:bb - cc cc cc cc cc cc cc cc  cc cc cc cc cc cc cc cc
          d  d  d  d  d  d  d  d   d  d  d  d  d  d  d  d
```

aaaa represents the current logical record of the file in hex notation, starting with record 0.

bb represents the offset from the first byte of the current logical record (bb will be in hexadecimal notation).

cc will be the hexadecimal representation of the byte listed.

d will be the ASCII representation of the byte. A period (.) will be used for all non-displayable bytes.

For example, the command **LIST BASIC/CMD.BASIC (H)** would produce a display as shown here:

```
0000:00 - 05 06 4C 42 41 53 49 43  1F 32 43 6F 70 79 72 69
          .  .  L  B  A  S  I  C   .  2  C  o  p  y  r  i
0000:10 - 67 68 74 20 28 43 29 20  31 39 38 31 20 62 79 20
          g  h  t     (  C  )     1  9  8  1     b  y
```

This is a listing of the file BASIC/CMD.BASIC in hex format. The logical record length was not specified in the command, and was found from the directory to be 256.

Parameter: REC

REC is used for listing hex files starting from other than record 0. It is entered as a decimal value, and starts the LIST with the specified logical record number of the file. The first record in a file is record 0; **REC=1** would list the second record of the file. The command **LIST MONITOR/CMD (H,R=5)** would start the listing with the sixth record.

Parameter: LRL

This parameter tells the LIST command to format the output using the specified LRL for each record. If the ***LRL*** parameter is not specified, the LIST command will use the record length in the file's directory entry. The ***LRL*** parameter is valid only when used with the ***HEX*** parameter.

Parameter: P

This parameter directs the output to the line printer rather than the video display. It may be used in conjunction with any of the other LIST parameters.

Parameter: TAB

This parameter will cause the expansion of any TAB characters (X'09') encountered during ASCII listings to the video display or line printer. If a tab= parameter is not specified, tabbing is fixed at every eight columns: 8, 16, 24, 32, 40, 48, and 56. The ***TAB*** parameter accepts a tab column number between 1 and 32 to set the actual tab position to a multiple of other than every 8 columns.

The following examples will show some different LIST commands.

```
LIST MONITOR/CMD (HEX,LRL=8,REC=0)
LIST MONITOR/CMD (H,L=8)
```

These two commands will produce identical results - listing a file called MONITOR/CMD to the video display, using an LRL of 8, and starting with the first record of the file. The second example has merely substituted the abbreviations for the ***HEX*** and ***LRL*** parameters, and let the REC parameter default to 0. This listing display will be only 8 bytes wide, as the LRL is also the display width (for LRL's = 1 to 16). Maximum display width is 16 bytes per line. The same line width applies to listings sent to the printer with the (P) option.

```
LIST REPLY/PCL (NUM,TAB,P)
```

The result of this command would be to send a listing of the file REPLY/PCL to the printer, using ASCII format, expanding all tab characters encountered and numbering each new line that is printed.

```
LIST TESTFILE/OBJ (NUM,HEX,REC=5)
LIST TESTFILE/OBJ (HEX,REC=5)
```

These commands produce identical listings of the file TESTFILE/OBJ. Remember that the ***NUM*** and ***LINE*** parameters are always superceded by the ***HEX*** parameter.

LOAD

The LOAD command will load a load module format file (such as a /CMD or a /CIM) into memory without invocation. The syntax is:

LOAD (X) filespec

filespec	is any valid DOS filespec that is in load module format.
(X)	is an optional parameter for a LOAD from a non-system diskette.

The LOAD command allows you to load into memory a disk file that is in the proper format. The default file extension for the LOAD command is /CMD. The following address restrictions exist when loading programs:

LOAD	Program must reside at or above X'5200' [V5] or X'3000' [V6].
LOAD (X)	Program must reside at or above X'5300' [V5] or X'3000' [V6].

After a program is loaded, control is returned to the system without invoking the loaded program.

The *(X)* parameter allows the loading of files that reside on an unmounted system or non-system disk. DOS will prompt you to insert the diskette with the desired file on it with the message:

```
Insert SOURCE disk <ENTER>
```

After the load is complete, you will be prompted to place the system diskette back in drive :0 with the message:

```
Insert SYSTEM disk <ENTER>
```

LOG

LOG is a program that will log in the directory cylinder and number of sides on a diskette. The syntax is:

LOG :d	
:d	Is any currently enabled drive.

The LOG utility will provide a way to log in diskette information and update the drive's Drive Code Table (DCT). It will perform the same log in function as the **DEVICE** command, except for a single drive rather than all drives. It will also provide a way to swap the drive :0 diskette for a double sided diskette.

LOG :0 will prompt you and allow you to switch the drive :0 diskette. The "LOG :0" command must be used when switching between double and single sided diskettes in drive :0. Otherwise, it is not needed.

Note: Double sided single density disks cannot be used to boot the Model I system.

MEMORY

The MEMORY command allows you to reserve a portion of memory, see the current HIGH$ (highest byte of unused memory), modify a memory address, or jump to a specified memory location. The Version 6 display output includes the status of switchable memory banks known to the DOS. It also displays a map of modules resident in I/O driver system memory [V6] and high memory. The syntax is:

MEMORY (CLEAR)
MEMORY (High=a,Add=a,Word=dd,Byte=d,GO=a)

a	Any address in hex or decimal notation
dd	Any hex "word" other than X'0000'.
d	Any byte in hex notation, other than X'FF'.
Add=a	Displays the word at the specified address. Also specifies the address for WORD and BYTE. For DOS 6, **a** can be a flag specification "A-Z", entered as a string: "A", "S",...
Byte=d	Changes the contents of ADD.
CLEAR	Will fill memory from X'5200' [V5] or X'2600' [V6] to HIGH$ with X'00'.
GO=a	Transfers control to the specified address. This parameter is always invoked last.
High=a	Will set the specified address as HIGH$. addr must be less than the current HIGH$.
P	Allows for printing the memory map.
Word=a	Changes the contents of ADD and ADD+1.

In all MEMORY commands, the GO parameter, if used, will be the last parameter to be invoked, regardless of its physical position in the

command line. All other parameters will be acted upon before the actual GO is done. The following restrictions are placed on the WORD and BYTE parameters:

WORD	Cannot = X'0000' or decimal value 0.
BYTE	Cannot = X'FF' or decimal value 255.

Refer to the following examples and descriptions.

All MEMORY commands will display the contents of system low memory ([V6] DOS only) and modules loaded into high memory, if any. The version 6 DOS will also display the status of all memory banks known to the DOS. This display will take the following form:

```
32K Banks avail = 11/11 In use=<---------->

 Low Memory Directory   Start=X'08F0'   Length = 2156
Module      Start Address        End Address       Length
$KI           X'08F0'              X'0B87'           664
$DO           X'0B88'              X'0E00'           633
$PR           X'0E01'              X'0E3C'           60
$FD           X'0E3D'              X'1014'           472
$MSC0         X'1015'              X'115B'           327
```

Banks	this shows the number of memory banks free and the total number of memory banks installed; the in-use field displays for each bank from 0 to n a "+" for in use and a "-" for not in use.
Module	is the name string of the resident module.
Start	is the lowest memory address used by the module.
End	is the highest memory address used by the module.
Length	is the number of bytes used by the module.

MEMORY with no parameters will display HIGH$ (highest unused memory location) in the format X'nnnn'.

MEMORY (HIGH=X'E000')

This command would set HIGH$ to memory address X'E000', as long as the existing HIGH$ was above this location. The MEMORY command will only move HIGH$ lower in memory. The RESET command will allow you to RESET HIGH$ to the top of memory.

MEMORY (ADD=X'4049')

This command will display the contents of memory locations X'4049' and X'404A'. The display will be similar to the following format:

```
X'4049' = 16457 (X'00E0')  HIGH = X'E000'
  ----    -----    ----          ----
  aaaa    bbbbb    cccc          dddd
```

aaaa is the address specified in hex notation.

bbbb is the decimal equivalent of ADD.

cccc is the contents of address and address +1, in LO-HI format.

dddd is the current HIGH$ address.

MEMORY (ADD=X'E100',WORD=X'3E0A')

This command will modify memory locations ADD (X'E100') and ADD+1 (X'E101'), changing them to the value of WORD. The following would be displayed after this command:

```
X'E100' = 57600 (X'0000' => X'3E0A')  High = X'E000'
```

All of the display is identical to the last example, except the contents of the WORD changed will be shown, represented in the display as XXXX.

MEMORY (ADD=X'E100',BYTE=X'0D')

This command will change the BYTE of memory at the specified address (X'E100') to X'0D'. The display after executing this command would be:

```
X'E100' = 57600 (X'0000' => X'0D00')  High = X'E000'
```

All of the display is identical to the last example, except for the modified BYTE change shown here with the XX.

MEMORY (GO=X'E000')

This command would transfer control to memory address X'E000'.

Note that the GO parameter may not be abbreviated.

PATCH

The PATCH utility is used to make minor changes or repairs to existing program or data files. The syntax of the PATCH utility is:

PATCH filespec1[/CMD] [Using] filespec2[/FIX] (parms)
PATCH filespec [USING] (information in patch format)

filespec1	Any valid filespec. The default extension will be /CMD.
filespec2	Any valid filespec for a "PATCH format" file. The default extension will be /FIX.
Option=N	Match F records before installing D patch [OFF].
Remove	Nulifies the effects of a previous D&F-type patch.
Yank	Will remove the PATCH specified by filespec2 from filespec1. The PATCH to remove must have been in the X'nnnn' type format.

The PATCH utility will allow you to change information in a file in one of two ways. If the file is in load module format (/CMD type files), it may be patched by memory load location. Any type of file may be patched by the direct disk modify method.

A patch is applied to a file either by typing in the patch code directly from the command line or by creating an ASCII file consisting of the patch information.

Either the BUILD command or the TED text editor may be used to create PATCH files. A word processor, such as SCRIPSIT, may be used provided the file is a pure ASCII file (with SCRIPSIT use the S,A type of save). Also, SCRIPSIT sometimes leaves extra spaces after the last carriage return in a file. To prevent this, position the cursor just after the last carriage return and do a delete to end of text to remove any extra spaces.

It is desirable to use some logical method of naming patch files. The filename of the patch code file could be followed by a letter or a number

that would be advanced as different patches become available for the same program. For example, BASIC1/FIX, BASIC2/FIX, SYS7A/FIX and SYS7B/FIX. Although not required, it is strongly suggested that all patch code files use the extension /FIX. This will make it easier to use these files as that is the default file extension that the PATCH utility will use.

Patch Line Syntax:

For PATCH to install properly, a definite structure and syntax must be observed when creating the file. All lines in a patch file must start with either a period or one of the four patch code verb identifiers: **D**, **F**, **L**, or **X**. A period indicates that the line is a comment line, and should be ignored by the patch utility. Comments in patch files are very useful for documenting the changes you are making.

The actual patch code lines will start in one of four ways:

X'nnnn'=	Change file starting at address **nnnn** to data
Drr,bb=	Change file starting at record **rr** offset **bb** to data.
Frr,bb=	Finds data at record **rr** byte offset **bb**.
Lnn	Designates DOS library command ISAM number **nn**.
data	hexadecimal values of the form: **xx xx xx ...** or ASCII values of the form: **"abcdefg..."**

The **Lnn** line is used to identify a particular library command module, and is not normally needed by the user. The **X** and **D** verbs are use to identify the patch line as either a patch by memory load location or a direct disk modify patch, respectively. The **F** verb is used simultaneously with the **D** verb to require a match of current file bytes prior to installing the patch code; this provides a safeguard which ensures that the correct file and location is being patched. Information following the "=" sign will be the actual patch code. It must be entered in one of two ways:

- It may be entered as a series of hexadecimal bytes separated by a single space.

- It may be entered as a string of ASCII characters enclosed in quotes.

No matter which method is used, there is never a space between the "=" sign and the start of the patch code. A physical line may contain two patch verbs. These are separated by a colon under DOS version 5 and a semi-colon under DOS version 6. For example,

```
D02,45=17 62;F02,45=15 50
```

would be a properly formatted DOS version 6 patch line; use a ":" separator character for DOS version 5.

DOS Patch Modes

```
X'nnnn'=nn nn nn nn nn nn .....
X'nnnn'="String"
```

The **X**-verb will patch a file by memory load location. The patch code will be written into a load module added to the end of the file being patched. This ending module will then load with the program and overlay or extend the code at **X'nnnn'**, where **nnnn** is the memory load address for the patch code. The patch code can be entered either as hexadecimal bytes, or may be represented as an ASCII string. It must be noted that this patch mode will extend the disk file, even if all of the patching is to the *inside* of the program. Because this type of patch will merely be added to the end of the file to be patched, it may later be removed with the **YANK** parameter.

```
Drr,bb=nn nn nn nn nn nn .....
Frr,bb=bb bb bb bb bb bb .....

Drr,bb="New String"
Drr,bb="Old String"
```

This is the direct disk modify patch mode. The **rr** represents the record number in the file to be patched, and the **bb** is the byte in that record where the patch is to begin. Normally, you will use a corresponding "**F**" verb patch line for each "**D**" verb patch line. In the "**F**" verb patch line, the data bytes entered must correspond to the exact bytes currently in the file at the record and byte position specified. The "**F**" verb patch line must immediately follow its corresponding "**D**" verb patch line. By using the

"F" verb along with the "D" verb, you will ensure that you are making alterations to the correct file record location because it requires finding of bytes matching a particular "F-verb" patch line prior to installing a "D-verb" patch. Again, the actual patch code can be either hexadecimal bytes or an ASCII string. This type of patch line does not extend the file and is applied directly to the record of the file.

Because no identification of the existence of this patch will be placed in the file, this type of patch cannot be removed by the **YANK** parameter; however, the **REMOVE** parameter can be used to un-install a "D-verb" patch that was installed with the appropriate required matching of "F-verb" patch lines. If you must attempt to install a "D-verb" patch line without using a corresponding "F-verb" patch line, you will need to specify the parameter, **OPTION=N**, when invoking the patch command. This allows you to force or inhibit the required matching (the default is to not require "F-verb" patch lines)

The **LIST** command with the (HEX) parameter can be used to display a file, showing the record number and the offset byte. This is an easy way to find the location in the file you wish to patch. Be aware that the first record in a file will be record 0, not record 1. DOS version 5 users may also use the FED utility to examine a file in hexadecimal mode display.

Lnn

This format is the indicator that the patch code that follows will be to either the SYS6/SYS, SYS7/SYS, or SYS8/SYS [V6] library command module. The "Lnn" represents the binary coded location of the desired overlay in the SYS module. The patch code that follows will be in either the "X'nnnn'" or "Drr,bb" format.

(YANK)

The patch **(YANK)** parameter will allow you to remove patches applied with the X'nnnn' format. The following rules will be in effect:

- The filespec of the patch to YANK must be identical to the filespec used when the patch was applied.
- If YANK is used without a filespec, no patch will be removed.

- **Do not patch a file more than once using the same filespec for the patch file!** It will be impossible to YANK the second patch from the file.

Here are some examples that will show the different patch formats.

```
PATCH BACKUP/CMD:0 USING SPECIAL/FIX:1
PATCH BACKUP SPECIAL
```

These commands would produce identical results. The default file extensions are /CMD for the file to be patched, and /FIX for the file containing the PATCH information. The patch information in SPECIAL/FIX might look like this:

```
. Special patch for my backup system only!
X'6178'=23 3E 87
X'61A0'=FF 00 00
```

This is an example of a patch using the X'nnnn' load location format. Note the comment line in the patch code file. This line will have no effect on the patch.

```
PATCH SYS2/SYS.SYSTEM USING TEST/FIX
PATCH SYS2/SYS.SYSTEM TEST
```

Note the abbreviated syntax of the second example. The USING and default /FIX extension are not necessary. The information in the patch file TEST/FIX might look like this:

```
.This will modify the SYS2 Module
D0B,49=EF CD 65 44
F0B,49=AF 3E 44 E5
. Eop
```

This is an example of the direct patch mode. It will patch the specified record and byte in the file SYS2/SYS. There are two comment lines in this patch file. Neither will have any effect on the patch.

```
PATCH SYS6/SYS LIB1
```

This command will patch the SYS6 library module. The patch file LIB1/FIX might contain the following information:

```
L54
X'5208'=32 20 DE AF 00 C3 66 00
```

This patch is in the memory load location mode. Library patches may also be done with the direct disk modify mode.

Patching with the command line format

Applying a patch from the command line uses the same formats for memory load location and direct disk modify already discussed. A library mode patch may not be done from the command line. It is also possible to specify more than one line of patch code from the command line. This is done by placing a colon (:) between the lines of patch code.

```
PATCH MONITOR/CMD:0 (X'E100'=C3 66 00 CD 03 40)
```

This command would patch the file MONITOR/CMD, creating a load module to replace the six bytes starting at X'E100' with the patch code specified in the command line. Since there is no filespec used for the patch code, the name CLP (Command Line Patch) will be assigned to the patch code. You may use this name if you wish to YANK the patch at a later date. However, if more than one command line patch is applied, only the first one can be yanked.

```
PATCH MONITOR/CMD:0 (O=N:D01,13=4C:D02,3E=66)
```

This command would patch the file MONITOR/CMD in two places. It uses the direct mode to apply the patches to the file's disk sector 1, relative byte 13, and disk sector 2, relative byte 3E. Since the **OPTION=N** parameter was specified in the command string, this "D-verb" patch can be applied without the verification provided by using the "F-verb".

PURGE

The PURGE command allows for controlled multiple deletes of disk files. The syntax is:

PURGE (-)(partspec w/wcc):d (parm,parm,parm)

:d	is the mandatory drivespec.
-partspec	as described in the DOS glossary.
wcc	Wild-Card Character <$> used as necessary for masking characters.
Date=	Used to specify files within a range of dates. Selection format is: "M1/D1/Y1-M2/D2/Y2"; "M1/D1/Y1"; "-M1/D1/Y1"; and "M1/D1/Y1-".
Inv	Specifies Invisible files.
MPW=	The disk master password.
Query=	ON or OFF. The default is ON.
Sys	Specifies System files.

The PURGE command allows the user to delete multiple disk files without the need to specify the individual filespecs. You will be prompted for the disk's master password if it is a password other than "PASSWORD" and not provided with the *MPW* parameter. The PURGE command allows several parameters to be set, providing for a specific group or groups of files to be purged. If the *Q* parameter is not specified, or if Q is specified without a switch, "Q=Y" is automatically assumed, and you will be asked before each file is purged. The responses are <**Y**> to purge the file or <**N**> to skip it. Pressing <**ENTER**> will also skip the file. The mod flag and date will be shown for each file. PURGE defaults to visible files only. To include invisible and system files, the *I* and *S* switches must be specified.

The **Date** parameter allows you to see only those files that fall within a specified range of dates. Files without dates will never be shown if the parameter was specified. The **Date** parameter accepts four formats to

provide for selecting specific ranges of dates; note that the parameters are character strings and must be enclosed in quotes. These formats are:

- DATE="M1/D1/Y1-M2/D2/Y2" copies only those files whose mod dates fall between the two dates specified, inclusive.
- DATE="M1/D1/Y1" copies all files with mod dates equal to the specified date.
- DATE="-M1/D1/Y1" copies all files with mod dates less than or equal to the specified date.
- DATE="M1/D1/Y1-" copies all files with mod dates greater or equal to the specified date.

Note: The files BOOT/SYS and DIR/SYS are not able to be purged and will never appear during execution of any PURGE command. Following are some examples and explanations of the PURGE command.

PURGE :0 (MPW="SECRET")

This command will purge all visible files on drive :0, assuming that the master password of the disk is *SECRET*. The purge will show each file and wait until a **Y** (Yes, purge it) or a **N** (No, don't purge it) is entered. To abort the purge, press the **<BREAK>** key at this prompt. If the master password does not match the password of the disk, the purge will abort with an error message.

PURGE :1 (Q=N,I,S)

This is a very powerful command. It will purge all files, including system files, from drive :1. If the disk's master password is other than "PASSWORD", you will be prompted for it. Once the purge starts, it will remove all files from the disk, except BOOT/SYS and DIR/SYS. **You will not be asked before each file is purged - it will be automatic!**

PURGE /BAS:1 (Q=N)

This command will first ask for the master password of the disk in drive :1 (if it is not PASSWORD). Once entered, all visible files with the extension

/BAS will be purged from the disk without asking for confirmation. You will, however, be able to stop the purge by pressing the <BREAK> key.

PURGE $$EX1:0 (I)

This command will purge all non-system files whose filename has the characters EX1 as the third, fourth, and fifth characters of the filename. The wildcard character ($) masks the first two characters of the filename (the filename may be more than five characters in length, as trailing characters in the field are ignored). The file extension will have no effect on this PURGE command. You will be asked before any file is actually purged, as *Q* was not specified and defaulted to Q=Y. Invisible and visible files will both be shown, as the I switch was used.

PURGE /$$S:2

This command will purge all visible files on drive :2 whose file extension contains three characters and ends in the letter S. This would purge files with the extension of /BAS, for example. However, it would not purge the system files, as the S switch was not specified. You will be prompted before each file is purged.

PURGE -/CMD:0 (I)

This command will purge all non-system files EXCEPT those whose extension is /CMD. You will be asked before each file is purged.

PURGE :1 (D="02/01/91-02/04/91")

This command will purge all visible files on drive :1, as long as their mod date is between 02/01/91 and 02/04/91, inclusive. You will be asked before each file is purged.

PURGE /SCR:2 (Q=N,D="-06/02/91")

This command will purge all visible files with a /SCR extension, provided their mod date is 06/02/91 or earlier. You will not be asked before each file is purged.

REMOVE and KILL

This library command is used to delete one or more specified files or devices from the system. The syntax is:

```
REMOVE filespec | *devspec [filespec | *devspec] [V6]
KILL filespec | *devspec [filespec | *devspec]    [V5]

filespec   The name of the file to be deleted.

devspec    The name of the device to be deleted.
```

The REMOVE [V6] or KILL [V5] command serves two purposes. It removes unwanted files from a diskette, thereby freeing up the space previously allocated to that file. It also removes devices that are no longer needed from the device table.

Deleting files

To delete a file, type in the following command at the DOS prompt:

REMOVE filespec

If the file is password protected, you must supply the proper password or the file will not be killed. To deal with deleting more than several files it is often easier to use the PURGE library command, which in effect is a "controlled" mass-delete. This may be the case if the files to be deleted contain a common filename or extension, as the PURGE command can deal with these files as a group. The PURGE command also ignores all file passwords as long as you know the master password of the disk.

REMOVE ALPHA/DAT:0

This command will delete the file named ALPHA/DAT that is present on drive :0. After execution of this command the file and the data in it will no longer be accessible to the system, so delete carefully.

REMOVE DELTA/DAT

This command will REMOVE the file DELTA/DAT on the first drive that it is on. Be carefull Without a drivespec you could delete a file that you did not intend to.

REMOVE MIDWEST/DAT.SECRET:0

This command will delete the password protected file MIDWEST/DAT.SECRET on drive :0. If the file's attributes include a protection level of NAME or higher, SECRET must be the password to delete the file. If the proper password is not supplied, an error message will be displayed and the file will not be deleted.

Deleting devices

DOS does not permit you to delete certain devices referred to as system devices. These devices are *JL, *KI, *DO, and *PR. Attempting to delete these devices will produce an error message, and the delete will abort. However, any other device may be deleted, as long as it is pointed NIL. The status of a device may be seen by issuing a **DEVICE (B)** library command. If a device is not pointed NIL, it must first be reset with the RESET command before it can be deleted. The command '**REMOVE devspec**' will completely remove the devspec from system device space.

REMOVE *CL

This command will in effect make *CL (the Comm Line) disappear from the system device control table, assuming the *CL was reset or pointed NIL before the delete was done.

REMOVE *SI *SO

This command will remove *SI and *SO from the device table. These two devices will appear pointed NIL in the device table upon power up. They are currently unused by DOS, and may be deleted if desired.

RENAME

This command will rename a file. The syntax is:

RENAME filespec1 [TO] partspec1filespec2
RENAME devspec1 [TO] devspec2 [V6]

filespec1 The current name of the file or device to be
devspec1 renamed.

filespec2 The new name to be assigned to the file or device.
devspec2

The RENAME command allows you to change the filename and extension of a given file. The RENAME command will use dynamic defaults for the filename, extension, and drivespec of filespec2. This means that any part of filespec2 that is not specified will default to that of filespec1. The drivespec of filespec2, if specified, must be the same as that of filespec1 or the rename will abort and an error message will be generated.

The version 6 DOS will also allow you to change the name of a device. This does not change the device's routing, linking, filtering, or setting. Devspec1 must be an existing device and devspec2 must not be the name of an existing device.

RENAME will not allow the changing or deleting of a file's password. To change or alter a password, refer to the ATTRIB command.

RENAME TEST/DAT:0 TO OLD/DAT

This command will rename the file TEST/DAT on drive :0 to OLD/DAT.

RENAME TEST/DAT:0 TO REAL

This command would rename the file TEST/DAT on drive :0 to REAL/DAT. The extension /DAT was not specified for filespec2, and defaulted to that of filespec1.

RENAME TEST/DAT:0 TO REAL/

This command will rename the file TEST/DAT on drive :0 to a file named REAL. The use of the "/" with no characters after it in filespec2 kept the extension from defaulting to /DAT.

RENAME TEST/DAT TO REAL/DAT

This command will search the active drives for the file TEST/DAT and rename it REAL/DAT.

RENAME TEST/DAT TO /OLD

This command will search the active drives for a file TEST/DAT and rename it TEST/OLD. The filename was not specified in filespec2, and defaulted to that of filespec1.

RENAME DATA/NEW.SECRET:1 TO /OLD

This command will rename the password protected file DATA/NEW.SECRET on drive :1 to DATA/OLD.SECRET. The filename and password for filespec2 defaulted to those of filespec1.

RENAME TEST/DAT TO TEST

This command is not a valid command, and will produce the error message "Duplicate file name". This is because the extension of filespec2 will default to /DAT, thereby creating the same filename for filespec2 and filespec1, which are the same file.

RENAME *FO *FF

This version 6 DOS command will rename the *FO device to the new name, *FF.

REPAIR

REPAIR is a utility program to update and correct information on certain types of diskettes to make them usable by DOS. The syntax is:

```
REPAIR :d

:d          is any currently enabled drive.
```

On the Model III or 4, REPAIR must be used to read any non-DOS disk created on the Model I, or any Model I LDOS earlier than LDOS 5.0.2. You must use the CONV utility to copy programs from TRSDOS 1.2 or TRSDOS 1.3 disks to DOS disks; you should never use the REPAIR utility on them. Disks created under other non-TRSDOS Model III operating systems may need to be repaired before being read.

On the Model I, it will not generally be necessary to REPAIR a disk to read it. However, other operating systems may mark the location of the disk's directory track in different manners. If you are having trouble reading a disk, the REPAIR command may be necessary.

REPAIR will perform the following functions.

- Update the Data Address Mark (DAM) for the directory cylinder to an X'F8'.
- Read enable the directory file, DIR/SYS.
- Check and correct the excess cylinder byte.
- Set the granules per cylinder byte (GAT + X'CC').
- Strip the high bit from the directory cylinder byte pointer at disk track 0, sector 0, byte 3; this is needed to convert certain DOSPLUS diskettes.
- Write DOS system information sectors onto the disk.

REPAIR functions

There are two types of Data Address Marks (DAMs) on every diskette - one to mark a data cylinder, and another to mark the directory cylinder. The DAM used to mark the directory track varies between operating systems, and may also be dependent on the computer hardware you are using. With DOS, this DAM has been standardized to be the same on any DOS diskette. The REPAIR command will change the directory DAM of the target disk to match the DOS standard.

Your disk directory contains information on the space allocated on the disk, as well as the names of your disk files. With DOS, the directory can be opened as a file called DIR/SYS. The REPAIR command will correct the protection level of the DIR/SYS file on non-DOS diskettes to allow them to be used in the same manner.

DOS keeps certain information in the directory about the number of cylinders on a diskette, as well as how much space is available on each cylinder. The REPAIR command will update this information on non-DOS diskettes. This will be necessary if you will be attempting a mirror image from a non-DOS to an DOS disk.

The location of the directory cylinder is stored on cylinder 0. Certain operating systems store this byte in a non-standard manner. The REPAIR command will correct this so the disk may be read by DOS.

The DOS system information sectors on cylinder 0 contain considerable information. The REPAIR command will place these sectors on a non-DOS diskette.

After the repair is complete, you should be able to copy any files off of the repaired disk.

Once a disk is repaired by DOS, it may not be readable by the operating system that created it. This is due to the directory DAM change. It is recommended that the REPAIR be used on a backup copy of the disk, if at all possible.

RESET

This command will restore logical devices to their original start-up condition, alter the logical record length of a file, allow a file's directory date to be altered, close the DOS version 6 directory open state for a file which has been left open, and provide a way to restore DOS version 5 HIGH$ (highest unused memory location) to the top of memory.

```
RESET filespec ([Lrl=n]|[Date=ON|OFF])
RESET devspec

Date=sw    sets the specified file's mod date to the current
           system date for DATE=ON, or to an un-dated
           pre-X.3 version format.

Lrl=d      sets the logical record length of the specified file to
           the new value entered.
```

There are many uses for RESET. The first is a global reset, the second is the reset of a single device. The global reset will reset all active devices, while the reset of a single device will affect only that device. A device's "default driver routine" referred to in the following explanations can be seen by doing a DEVICE command before any configuration is done. If the device has been filtered, linked, routed, or set to another driver, the new device address will be shown by the DEVICE command.

RESET of a file

Under DOS version 6, if the network flag has been enabled, open files are flagged in the directory. A file can be left in an "open state" if a program does not close it prior to terminating. The DIR command discloses this condition via a "?" character in the attribute's field of the file. A "File already open" error also may indicate such a condition. To clear this condition, issue the **RESET filespec** command.

Parameter LRL is used to alters the logical record length of the file to the value n; appropriate values of n are in the range 0-255, with 0 implying a record length of 256. The following example changes the logical record length of MYDATA/DAT file to 128.

RESET MYDATA/DAT (Lrl=128)

The *DATE* parameter is used to allow the file's directory date to be altered. Specifying **RESET filespec (DATE=OFF)** restores the file's directory entry to the old-style dating of pre-x.3 release. The old accessluser password field will be set to blanks. This may be useful for files which must be transferred to an older DOS release, such as by the TRSDOS 1.3 CONVERT utility. Specifying **RESET filespec (DATE=ON)** will establish the file's directory date as that of current system date and time. Note that when this option is specified, the MOD flag will be set to indicate a file change.

Single device RESET

A single device reset will accomplish the following. Any filtering, linking, routing, or setting done to the device will be removed. Any open disk file connected to the device will be closed. Invoking a DEVICE command will show the device pointed to it's normal default driver routine. If the device has been created by the user, it will be pointed NIL when reset, and the REMOVE or KILL command can remove it from the device display at this time. If high memory was used when this device was altered, it will not always be reclaimed by the system. However, most system-provided modules can re-use the same memory allocation if they are enabled again after being disabled or reset.

Here are some examples of the RESET *devspec command.

RESET *PR

If you had your printer (*PR) filtered with the PR/FLT routine, the RESET *PR command would restore the normal I/O path between the printer DCB and its default driver.

RESET *DU

Suppose you had a dummy device *DU routed to a disk file TEST/TXT, and had your printer (*PR) linked to *DU. This configuration would cause all output to the *PR to also go to *DU, and into the disk file TEST/TXT. If you RESET *DU, the device table would show *DU = Nil, and the file TEST/TXT would be closed. However, *PR would still be linked to *DU. Since *DU = Nil, any output sent to the *PR would be ignored by *DU. The printer (*PR) would function normally. To clear the LINK, issue a

RESET *PR command. *DU would continue to be shown in the device table until the system is rebooted, *DU is deleted, or a global RESET is performed.

Global RESET

The version 5 RESET command with no devspec will do a global reset. All system logical devices will be returned to their default driver routines. All user logical devices will be removed from the device control table. Any filtering, linking, routing, or setting will be cancelled. All open files will be closed.

The Drive Code Table will be returned to its default state, with the drive configuration coming from the system information sectors of the current system disk. Any software write protection will be cancelled. If your system drive has been set to some drive other than physical drive 0, be sure to insert a system disk into physical drive 0 before performing a global reset. Any high memory disk drivers such as FDUBL, or hard disk drivers will be removed from memory, and access to any drives requiring one of these drivers will not be possible.

The system will also attempt to set HIGH$ to the top of the available memory. If certain system functions that use the task processor are active (such as the SPOOL command or the blinking cursor on the Model I), the reset cannot restore HIGH$ to the top of memory. If this is the case, the following message will appear.

```
Can't reset memory, background task(s) exist
```

To reset HIGH$, you must turn off the particular function or reset the individual device before doing the global reset.

ROUTE

The ROUTE library command re-routes input/output for a specified logical device or creates a device. The syntax is:

ROUTE devspec1 [TO] filespec | devspec2 [(parms)]
ROUTE devspec (NIL)

NIL	Specifies that devspec1 is routed to a bit-bucket.; output sent to the device will be discarded and ignored.
Rewind	Resets the file pointer of filespec so existing data can be reread or overwritten.

ROUTE will re-route all I/O for a specified logical device to another logical device, to a disk file, or (NIL). ***NIL*** means that the device is routed to nothing. Any output sent to a device routed (NIL) will simply be ignored. A device routed (NIL) also has no input. **Note:** No more than four devices may be routed at any one time on the Model III.

Anytime a device is routed to a filespec, a File Control Block (FCB) and a blocking buffer will be dynamically allocated in high memory. The system will determine the current HIGH$ (highest unused memory location) and use the space directly below this location for its buffer. HIGH$ will then be decremented to protect this area. ROUTE will be able to reuse a previously installed but currently unused high-memory support module of a previous ROUTE of the same device.

If the designated filespec already exists, the data routed to that file will be appended to the end of the existing file. If you wish the data to be written from the beginning of the file, specify the ***REWIND*** parameter which allows you to start the routed file at its beginning. In some cases, it will be advisable to use the CREATE command to allocate file space before doing the route.

A new logical device may be created with the ROUTE command. To create a device, simply route the desired devspec to another devspec, to a filespec, or to (NIL). The new device will then appear in the device table.

To examine any currently existing routing, use the DEVICE command. The device notations shown directly below the disk drive configurations will indicate all currently recognized devspecs and any routing, among other things, that has been done.

Once a device has been routed, it may be returned to its normal power-up state or removed completely from the device table with the RESET or REMOVE commands.

ROUTE *PR *DO

This command will route any data sent to the line printer (*PR) to the video display (*DO). None of the characters will be printed by the line printer, but instead will be shown on the video display. This command is very similar to LINK *PR *DO, the exception being that the characters are not printed by the line printer with the route but are with the link. The line printer need not be hooked to the system if *PR is routed to *DO. To remove the routing, use the command RESET *PR.

ROUTE *DU TO TEST/TXT:0

This command will route a user device (*DU) to a disk file TEST/TXT on drive :0. A File Control Block and a blocking buffer will be established in high memory. The device table will show the routing with an entry of:

```
*DU  => TEST/TXT:0
```

The file TEST/TXT will remain open as long as the device *DU is not RESET. The file must be closed with the RESET *DU command prior to removing the diskette from the drive.

ROUTE *PR TO PRINTER/DAT

This command routes all data normally sent to the line printer (*PR) to a disk file PRINTER/DAT. The system will search all active drives and use the first file PRINTER/DAT it finds. Any data sent to the *PR will then be appended to the end of the PRINTER/DAT file. If the file does not exist, it will be created on the first available drive. An FCB and blocking buffer will be allocated in high memory and the file PRINTER/DAT will remain open until the *PR is reset.

Before routing any device to a disk file, it is advisable to determine the amount of free space available on the diskette. Make sure the space available on the disk is adequate to hold the amount of data you wish to route to it! A "Disk space full" error may lock up the system if encountered when writing to a file via the ROUTE command.

The constant "*EOF maintenance*" file mode may be useful to invoke when routing to disk files (see use of the "trailing ! character" with filespecs, in the Glossary). This will cause the EOF (End Of File) to be updated after each buffer is written to the file. If EOF maintenance is not invoked, then the EOF will not be written to the file until the routing is reset, which will properly close the file. If a file is not properly closed, the data written to it may not be recoverable. If a "Disk space full" error is encountered when the EOF maintenance has been invoked, all data up to the last "*full*" buffer written to the file will be intact, and the file will be readable by the system.

RUN

The RUN command will load a program into memory and then invoke it. The program must be in load module format. The syntax is:

RUN (X) filespec (parm,parm,...)

filespec	is any valid DOS filespec of a file in load module format.
(X)	is optional to invoke the program from a non-system disk for the single drive user.
parm	optional parameters to be passed to the filespec program.

The RUN command will load in a load module format program that resides above X'51FF' for version 5 or X'2FFF' for version 6, and then invoke the program. The default extension for the filespec is /CMD. If the program resides on a non-system diskette, the (X) parameter may be specified to load the program from that diskette and begin execution only after a system disk has been reinserted in drive 0.

If the (X) parameter is used the program must load above X'52FF' for version 5 or X'2FFF' for version 6.

The RUN command is identical to the LOAD command except for the fact that control is transferred to the program module transfer address rather than returning to the system. Load module format programs may also be directly loaded and invoked from the DOS Ready prompt by simply typing in the name of the program.

Following are some examples of the RUN command.

```
RUN SCRIPSIT/LC
SCRIPSIT/LC
```

Both of these commands will produce the same results. The program SCRIPSIT/LC will be loaded into memory and invoked.

RUN BASIC
BASIC

Both of these commands will produce the same results. They will load a program named BASIC/CMD and invoke it. Note that the file extension defaulted to /CMD when not specified by the RUN command.

RUN (X) INVADERS/CMD

This command is for the single drive user. It will load the program INVADERS/CMD from any disk, whether or not it is an DOS system disk. After the command has been entered, you will be prompted with the message:

Insert SOURCE disk <ENTER>

At this point, you should insert the diskette containing the program in drive :0 and press **<ENTER>**. After the program is loaded, you will be prompted:

Insert SYSTEM disk <ENTER>

You should now insert your DOS system disk back into drive :0 and press **<ENTER>**. Program invocation will begin at this point.

SET

This command sets a logical device to a driver routine. The syntax is:

SET devspec [TO] filespec[/dvr] (parm,parm,...)

devspec	Any currently enabled logical device.
filespec	Any valid "driver type" program.
parms	Optional parameters required by the driver program specified with filespec.

The SET command will set a logical device to a driver program. It does this by loading the specified driver program, which will relocate itself into high memory just below HIGH$, lowering HIGH$ to protect itself. Once a device is set, any I/O to or from the device will be controlled by the new driver routine. DOS will allow the passing of parameters to the driver program. These parameters are totally independent of the SET command, and are determined only by the needs of the driver program.

The filespec parameter is the filespec of the driver program. The default extension for this file is /DVR.

When a device is set, any previous filter, route, link, or set of that device will be destroyed. Once a device has been set, it will remain set until it is either routed or reset. The driver program will remain in high memory even if the device is reset. Some driver programs may not be able to re-use their high-memory modules when re-installed after they are reset. If such a device is reset, and then set again, the driver routine will load in below the current HIGH$. As a result, the setting, resetting, and then setting again of devices will cause the available memory to continue shrinking. Once a driver program is loaded, it will not be removed from memory or overwritten, even if the same device is reset and then set to the same driver. A global reset (if allowable) will remove this driver program and free up the memory by resetting HIGH$.

```
SET *CL TO RS232T/DVR (BAUD=300,WORD=7)
SET *CL RS232T (BAUD=300,WORD=7)
```

These two commands produce identical results. The *TO* is optional and may be replaced by a single space. Specifying the filespec RS232T/DVR produces the same results as specifying the filespec RS232T, as the default extension is /DVR. The RS232T program is for the Model III computer. The Model I version could be used instead with identical results. For Model 4 operation, the driver name is COM/DVR; no parameters are permitted with it.

These commands set the communications line "*CL" to a driver routine called RS232T/DVR. This is an actual version 5 DOS driver program, and is described in the *DEVICE DRIVER* section. The parameters BAUD and WORD are valid parameters of the RS232T/DVR program. All I/O to/from the communications line will be sent through this driver routine, and be properly dealt with to be sent out the RS-232 interface.

SET *KI INKEY (F=64)

This command sets the keyboard (*KI) to a user driver program named INKEY/DVR, and passes the parameter F=64 to the driver program.

SET *PR RS232x

This command would set *PR (the line printer) to one of the supplied RS232 driver programs. This would be the normal way to use a serial printer with DOS.

SETCOM

This command allows you to change or display the parameters of the serial device driver after the driver has been installed into memory. It's command syntax is as follows:

SETCOM	[(parms)]
DEFAULT	Set to system defaults shown below in brackets.
Baud=n	Sets the BAUD rate [300].
BREAK=n	Sets logical break character to n . The default is [OFF] for DOS 5 and <CTRL><C> for DOS 6.
Parity=	Sets parity; ON, OFF, "EVEN", "ODD" ["EVEN"]
Query	A version 6 parameter which provides prompting for each parameter noted above.
Stop=n	Sets stop bits, 1 or 2 [1].
Word=n	Sets word length, 5-8 bits [7].
CD=	Carrier Detect [IGNORE]
CTS=	Clear To Send [IGNORE]
DSR=	Data Set Ready [IGNORE]
DTR=	Data Terminal Ready [ON]
RI=	Ring Indicator [IGNORE]
RTS=	Request To Send [OFF]

In order to be able to use SETCOM, you must first install the serial driver using the SET command. Additional information is provided in the section documenting the serial driver.

SETKI

The DOS version 6 SETKI command is used to adjust the keyboard reaction of key repeat when you press a key. The syntax is:

SETKI [(parms)]

Default	Returns the parameters to their default values.
Rate=r	Sets the repeat rate as **r**. **r** is a number in the range 1-127; 2 is the default value.
Query	Prompts you to enter new values for RATE and WAIT.
Wait=w	Sets the initial delay between the time a key is first pressed and the first repeat of that key as w. **w** is a number in the range 10-127; 22 is the default.

SETKI changes the keyboard repeat key response. If you do not specify a parameter, the current delay and repeat rate settings are displayed.

SETKI (WAIT=15) <ENTER>

This example sets the delay rate to 15.

SETK I <ENTER>

This example displays the current delay and repeat rate settings in the format:

Wait= 15, Rate =2

Note: Both the RATE and WAIT parameters use modulo 128. For example, entering 138 has the same effect as entering 10.

SOLE

SOLE is a DOS version 5 Model I utility which is used to install a double-density booting driver onto a system disk which has been formatted in dual-density. It's syntax is:

SOLE [:d]	
:d	references the drive containing the dual-density disk; if not specified, it will default to :0.

The Model I Radio Shack BOOT ROM only supports booting from a diskette with a single density track 0; thus, a special dual-density system disk is needed to boot a double density DOS disk. The Model I FORMAT utility will format a double-density disk with a single-density cylinder :0 when both the **DDEN** and **SYSTEM** parameters are specified. This assumes that the double-density disk driver, FDUBL, has been installed. After the operating system has been moved to the dual-density diskette, the booting driver is applied by using **SOLE.**

If your Model I is equipped with a double density adaptor, you may wish to make a double density boot disk, either one or two sided, using the following procedure:

1. Boot the DOS system disk and issue the command,

   FDUBL (Tandy I Percom)

choosing one of the two parameters depending on the type of adaptor present in your machine. Note that **FDUBL** defaults to the "Tandy" parameter.

2. Place a blank disk in drive :1 and issue the command:

   FORMAT :1 (SYSTEM)

and respond to the queries with the desired diskette configuration data.

3. Move the files from the system disk to the newly formatted diskette with the command:

```
BACKUP  :0  :1 (SYS,INV)
```

4. Save your configuration by issuing the command:

```
SYSTEM  (SYSGEN,DRIVE=1)
```

5. Install the double density boot file by issuing the command:

```
SOLE  :1
```

6. At this point, remove both diskettes then place the double density DOS disk into drive :0. Insert the DOS diskette #2 into drive :1 and copy the files from it to drive :0 and re-save the configuration with the commands:

```
BACKUP  :1  :0 (INV)
SYSTEM  (SYSGEN)
```

7. Make additional copies of this double density system diskette onto blank diskettes in drive :1 by issuing the commands:

```
FORMAT  :1  (SYSTEM)
BACKUP  :0  :1
```

Note: You cannot use DISKCOPY or QFB to backup a dual-density diskette.

SPOOL

The SPOOL command establishes a FIFO (First In, First Out) buffer for a specified device (usually a line printer). The syntax is:

```
SPOOL [devspec] [TO] (filespec)
          (Mem=a,Disk=b,Bank=c)
SPOOL devspec (OFF | Pause | Resume | Clear)
```

devspec	Is any valid DOS device; if not entered, *PR is assumed.
filespec	Is an optional DOS filespec; if not entered, devspec/SPL is asumed.
Bank=c	A version 6 parameter in the range 1-30 which specifies that the spool buffer and main spooler module should use bank c; this requires machines with expanded memory installed.
Clear	Empties the spooling buffers without printing.
Disk=b	Disk space to be used by the spooler (0 to bb); b is the amount of disk space to be used in blocks of 1K (1024 bytes).
Mem=a	Memory to be used by the spooler. a is the amount of memory to be used, in blocks of 1K (1024 bytes). 1K is automatically used.
OFF	Turns off the spooler and resets devspec.
Pause	Allows you to temporarily suspend despooling.
Resume	Allows you to continue despooling after a pause.

Model I/III Note: For proper SPOOL operation, the KI/DVR program must have been set before turning on the spooler.

When using the SPOOL command with a serial printer, you must use the following steps for proper operation. First, SET *PR to the serial driver, FILTER *PR with the FORMS or PR filter, then turn on the spooler.

The SPOOL command will establish a first-in-first-out (FIFO) buffer for a specified device. All output sent to the device will be placed in an output buffer consisting of memory and/or disk buffers, and will be sent to the device whenever that device is available to accept this data. The spooler will free up the memory it uses when turned off, provided the spooler was the most recent module installed. If the memory it used was trapped by a subsequent module's installation, the spooler will re-use its previous memory allocation and inhibit alteration of its previous memory and disk parameters.

The minimum amount of memory required by the SPOOL command is 1K (1024 bytes) for the memory buffer. The filespec is optional, and if no disk buffers are required, it is possible to spool strictly to memory. If disk space is requested, additional memory will be used to map the spool file area. The more disk space used, the larger the required memory block will become. Using parameters that would cause the memory used to go below X'8000' will not be allowed (Note that X'8000' is an approximate value, and may vary +/- 256 bytes).

When the spooler is active, output to the specified device is treated in the following manner. Any output data which cannot immediately be accepted by the device is sent to the memory buffer. When the memory buffer is full, the data is sent to the disk buffer (if one has been specified). The stored information is sent to the device in a FIFO manner. Output of the stored data to the device is carried on as a background task even when the system is performing other functions.

Parameter: devpec

The devpec is the name of the device whose output is to be spooled. The printer is the most usual device to be spooled; thus, devspec will be assumed to be *PR no device specification is provided.

Parameter: filespec

The filespec is the name of the file the SPOOL command will write to any time its memory buffer is full. The default extension for this file is /SPL. The default filename will be the two letters of the devspec which is being

spooled. For the filespec parameter to be valid, the "DISK=" parameter must not have set the disk file allocation to zero. Refer to the following examples.

```
SPOOL *PR TEXTFILE:0
```

The filespec will be TEXTFILE/SPL:0, as the file extension was not specified and defaulted to /SPL.

```
SPOOL *PR PR/TXT:0
```

The filespec will be PR/TXT:0. Specifying the /TXT extension will override the default /TXT.

```
SPOOL *PR :1
```

This command will look on drive :1 for a file named PR/SPL. If the file PR/SPL:1 is not found, it will be created on drive :1 with a length determined by the "DISK=" parameter.

```
SPOOL *PR
```

This command will search all active drives for a file named PR/SPL. If this file is not found, the file PR/SPL will be created on the first available drive (with the file size determined from the "DISK=" parameter).

Parameter: MEM

As stated earlier, the SPOOL command will always require a minimum of 1K (1024) bytes for a memory buffer. If more memory is required, it may be allocated with this parameter. For example:

```
MEM=10
```

will allocate 10K (10,240 bytes) of memory to be used as a spool buffer. This memory will be dynamically allocated by the fully integrated spool system processor to provide the most efficient operating environment, depending on the particular configuration you have established for your DOS system. If this parameter is not specified, 1K of memory will automatically be allocated for the spool buffer.

Parameter: DISK

This parameter sets the maximum amount of file space to be allocated for the spooling. Disk space is allocated in blocks of 1K (1024 bytes), the same as memory. When this parameter is set, the system will create a file of the size specified. If this parameter is not specified, the SPOOL command will automatically allocate approximately 5K of disk space, depending on the particular disk type. The file name of this file will be determined by the filespec parameter.

To prevent the SPOOL command from using any disk space, specify this parameter:

(DISK=)
(DISK=0)

By specifying the "DISK=" parameter with no size, the system will not allocate any disk space to the SPOOL command and will not create any file.

Parameter: BANK

This parameter allows for the spool buffer and most of the spooler module code to reside in a 32K memory bank other than normal high memory (bank 0). If your computer is equipped with expanded memory known to the DOS (use the MEMORY command to determine this), then you should utilize BANK to install the spooler into a bank higher than 0 to use the minimum amount of high memory. For BANK other than 0, MEM is automatically set to utilize all of the requested bank.

The following examples will show some possible combinations of the spool parameters.

SPOOL *PR TEXTFILE:0 (MEM=5,DISK=15)

This command will allocate 5K of memory and 15K of disk space in a file named TEXTFILE/SPL on drive :0. Any output sent to the printer will be buffered and sent to the line printer (*PR) as fast as the printer can accept the characters. Even if current program printing functions exist, other functions will be carried out and the line printer will continue to receive data from the spooled buffers as fast as it can accept the data. The other

program function processing will be carried on with little noticeable interruption. If the 5K memory buffer is filled, the data will then be written to the disk file TEXTFILE/SPL on drive :0.

SPOOL *PR (MEM=10,DISK=)

This command will create a 10K memory buffer for any data that is to be sent to the line printer (*PR). If a printer command is received, the data will be immediately sent to the 10K memory buffer, and then spooled to the line printer whenever the printer can accept it (i.e whenever the printer is not printing or otherwise in a BUSY or FAULT state). Since the data is sent to the printer as a background interrupt task, normal program execution will continue to take place. Note that none of the spooled data will be sent to a disk file, as the parameter *DISK=* was specified without any size. If the memory buffer is filled, processing of the current program functions will halt until the line printer has printed enough data to bring the outstanding character count below 10k (the size of the memory buffer).

If you are running an applications program that involves output to the line printer, it is possible that the overall efficiency of the program may be inproved by activating the DOS spooler. The size of the program and the available free memory and disk space will determine the amount of spooling available for your needs.

Parameters: PAUSE, RESUME, and CLEAR

While a device's output is being spooled and despooled, the despooling operation may be temporarily suspended by using the PAUSE parameter. Regardless of the device being spooled, all you need enter to suspend despooling is:

SPOOL (P)

Note that the device's output continues to be spooled to memory and disk; it is the despooling to the printer which is suspended. Once despooling is suspended, it may be resumed by entering SPOOL (R). If you need to clear out any spooled text remaining in the spool buffers, use SPOOL (C). The buffered text remaining will then be lost.

Caution

The spool file on disk will remain open as long as the spooler is active. Do not delete this file or remove the diskette without first closing the file! You will not be allowed to do a SYSGEN if the spooler is active.

The file may be closed by turning the spooled device OFF. The proper syntax is:

SPOOL devspec (OFF)
SPOOL devspec (N)

Either of these two commands will turn off the spooler and close the associated disk file. Additionally, any other filtering, linking, setting, or routing done to *PR will be reset. Please note that the disk file will not be closed by resetting or deleting the spooled device. It must be turned off to close the file.

Once the spooler is turned off, it may be turned on again. Doing so will re-use the same memory locations allocated when it was originally turned on. The original parameters will be re-used.

SYSGEN

This command creates or deletes a configuration file. It's syntax is:

SYSGEN ([ON|OFF][,DRIVE=d])		[V6]
SYSTEM (SYSGEN[=ON|OFF][,DRIVE=d])		[V5]
DRIVE=d	Specifies the drive for the SYSGEN [default is 0].	
OFF	Removes the configuration file from drive d.	
ON	Creates a configuration file on drive d. *On* would be assumed if neither *ON* or *OFF* is entered.	

This command creates or deletes a configuration file on the disk drive specified. In the version 5 DOS, SYSGEN is a sub-command of the SYSTEM command; thus **SYSTEM (SYSGEN=switch,Drive=d)** is the required entry.

If switch is not specified, "ON" is assumed. That is, **SYSGEN** is the same as **SYSGEN (ON)**, and **SYSTEM (SYSGEN)** is the same as **SYSTEM (SYSGEN=ON)**. After a **SYSGEN (OFF)** command has been given, the current configuration of the system will not change until the system is booted again.

When SYSGEN is invoked, all current device and driver configurations will be stored on the disk in drive :0 or that drive specified by the "DRIVE=" parameter. Job Control Language, if active, will be temporarily suspended during the configuration generation; SYSGEN will not abort if within a JCL file. SYSGEN will abort with an appropriate error if the drive specified to receive the configuration file is write protected. An invisible file named **CONFIG/SYS** will be created to hold the configuration. Each time the system is booted, the configuration stored in this file will be loaded and set. You may prevent this automatic configuration by holding down the **<CLEAR>** key while the boot is in progress. Note that the system configuration will take place before any AUTO'ed command is invoked.

In addition to the options settable by the SYSTEM command, the following will be stored in the configuration file by a **SYSGEN** or **SYSTEM (SYSGEN)** command:

- All filtering, linking, routing, and device setting that has been done. This includes the serial driver and *KI settings.
- Any active background tasks (such as CLOCK, DEBUG, TRACE, etc).
- Any special utility modules or user programs loaded into high memory and protected with the MEMORY command. All memory from HIGH$ to the physical top of memory and added version 6 low memory modules will be written to the CONFIG/SYS file.
- All modules added to DOS version 6 I/O driver memory.
- The present state of VERIFY (either ON or OFF).
- All Device Control Blocks. This will include the current lines per page and line counter stored in the printer DCB.
- The state of the CAPS lock for the keyboard.

Certain DOS features should never be SYSGENed if a disk file is involved. They are any ROUTE or SET involving a disk file. SYSGENing open files can cause loss of data if the disks are switched in the drives without the files being closed. Disk switches with open files can also cause existing data to be overwritten.

SYSTEM

This command is used to select DOS features and configure the user definable areas of your DOS system. The syntax is:

SYSTEM (parm,parm,...)

SYSTEM parameters vary with DOS version. They are:

ALIVE	BASIC2	BLINK	BREAK
BSTEP	DATE	DRIVE	FAST
GRAPHIC	HERTZx	RESTORE	SLOW
SMOOTH	SVC	SWAP	SYSGEN
SYSRES	SYSTEM	TIME	TRACE
TYPE	UPDATE		

The existing configuration of your DOS system can be seen by doing the DEVICE and MEMORY commands. The SYSTEM command can set or change the disk drive configuration as well as turn on or off different keyboard, video, and hardware drivers. Each valid SYSTEM sub-command will be discussed in this section.

Once your DOS system has been configured, you may store the configuration on the disk in the drive you BOOT with using the SYSGEN command [V6] or SYSTEM (SYSGEN) parameter [V5]. Please read this section thoroughly to determine the different SYSTEM sub-command uses, and to discover exactly how other DOS commands will affect the (SYSGEN) parameter.

Certain of the SYSTEM commands must load driver routines into low [V6] or high memory to accomplish their functions. When they do this, they determine the highest unprotected memory location (referred to as HIGH$) and load directly below this location (or lowest version 6 unprotected memory address below the resident system and load the module above this). After loading, the DOS system moves HIGH$ down (or LOW$ up) to protect these routines. If you have invoked any SYSTEM commands that require the use of this high memory, be aware that your overall free memory will be decreased accordingly.

Most of the following sub-commands for the SYSTEM command may be used together in the same command line. To do this observe the syntax: **SYSTEM (parm,parm,....,parm)**. Each parameter must be accompanied by its switch or setting as required.

SYSTEM (ALIVE=switch)

ALIVE displays an alternating character in the upper right corner of the screen. It is primarily used to determine the current state of the task processor. If the alive bug is alternating, the task processor is running. Note that the character may continue moving (indicating an alive system) even when the TRACE library command display has stopped.

The switch is either "ON" or "OFF". If not specified, "ON" is assumed. The ALIVE parameter uses some RAM in high memory. The ALIVE module can now be removed from memory.

SYSTEM (AMPM) [V6]

The display of the time field in the version 6 DIR command's output can be altered to display the time in 12-hr clock time or 24-hr clock time. Specifying **SYSTEM (AMPM=ON)** will cause a 12-hr clock display; specifying **SYSTEM (AMPM=OFF)** will cause a 24-hr clock display. This option is configurable and pertains only to the DIR display.

SYSTEM (BASIC2) [V5]

This command will direct you to the ROM Basic in the TRS-80 computer. Typing in **SYSTEM (BASIC2)** while in the DOS Ready mode will function identically to pressing the reset button with the <**BREAK**> key held down. The screen will clear, and "Coss ?" (Model III) or "Memory Size?" (Model I) will be displayed in the upper left corner of the display. Any routing, linking, or driver routines set under DOS will be reset to the normal ROM Basic drivers. While in ROM BASIC, none of the disk functions are available for use and you cannot return directly to DOS or BASIC. You must press the reset button or turn off the computer and go through power up to get back to the operating system.

SYSTEM (BLINK=aaaa)

This command controls the DOS cursor character. The parameter *aaaa* can be represented as **ON**|**OFF** or as a decimal value. The cursor character

numbers in the following examples are the ASCII values (in decimal) of the TRS-80 character set. This command will use high memory on the Model I. The BLINK module can now be removed from memory.

ON Turns the blinking cursor on, with the cursor character being a graphics block (character 176, ▮) for version 5 or an underline (character 95, _) for version 6.

OFF Turns off the blinking cursor. On the Model III, the cursor character will be a non-blinking graphics character. The Model I will have the normal power-up cursor.

aaaa can also be represented as any displayable ASCII character value. For example, if the command SYSTEM (BLINK=42) were given, the blinking cursor character would be an asterisk (character 42).

SYSTEM (BLINK,LARGE)

This command turns on a large (character 143) blinking cursor.

SYSTEM (BLINK,SMALL)

This command turns on a small (character 136) blinking cursor.

SYSTEM (BREAK=switch)

This command will enable or disable the <BREAK> key. The allowable switches are ON or OFF. If switch is not specified, the default will be ON. Once the <BREAK> key is disabled by typing a **SYSTEM (BREAK=OFF)** command, pressing it will have no effect, and the system break bit will not be set. It may be re-enabled at any time by doing a **SYSTEM (BREAK=ON)** command. The **(BREAK=ON)** will also enable the <BREAK> key if it was disabled by the *AUTO* library command. No memory will be used with this parameter.

Note: Specifying **(BREAK=OFF)** will prevent routines such as the *BUILD* command from exiting when the <BREAK> key is pressed!.

SYSTEM (BSTEP=n)

This command will establish the default bootstrap step rate used with the FORMAT utility. The BSTEP value can be 0, 1, 2, or 3. These values correspond to the same step rates as described for the **SYSTEM (DRIVE=,STEP=)** command:

n	8" floppy step rate	5-1/4" floppy step rate
0	3 ms	6 ms
1	6 ms	12 ms
2	10 ms	20 ms
3	15/20 ms	30/40 ms

This value will be stored in the system information sector on the current drive :0 if the "DRIVE=" parameter is not specified. If you switch drive :0 disks or change drive :0's with the **SYSTEM (SYSTEM)** command, be aware that this default value will be taken off the new drive :0 disk.

SYSTEM (DATE=switch)

This command is used to enable or disable the initial prompting for the date on power up. The diskette may not be write protected when using this command. The switch may be "ON" or "OFF" as follows:

ON Enables the date prompt if it has been disable with the *OFF* parameter. If the switch is not specified, on is assumed.

OFF Disables the date prompt on power up or reset.

Since the date is used extensively throughout the DOS system, it is recommended that you never disable the initial date prompt with this command. The date will remain set even if you press the computer's reset button, so you will not have to re-enter it.

SYSTEM (DRIVE=d,parm,parm,...)

This command sets certain parameters for the disk drives in your system. Refer to the following for explanation of the allowable parameters. This command uses no extra memory.

DRIVE=d represents any valid drive number in your system. Only one DRIVE=d parameter can be used in any system command line.

CYL=nn This command will set the default number of cylinders (in the range 35 to 96) to be used with the FORMAT utility for the specified floppy drive. This value will be written to the system information sector on drive :0 if the "DRIVE=" parameter is not specified. If you switch drive :0 disks, be aware that this value will be taken from the new drive :0 disk when formatting.

DELAY= This command is valid only for 3.5" and 5.25" drives. The DELAY is the time allowed between drive motor start up and the first attempted read or write of the diskette in that drive. The **ON** parameter sets the delay to 1.0 second. This is the normal DELAY time for all 5.25" drives. The **OFF** parameter sets this delay to 0.5 seconds.

DISABLE This command will remove the specified drive number from the Drive Code table. Once disabled, any attempt to access that drive will cause the message "Illegal Drive Number" to appear. The drive can be re-enabled with the ENABLE parameter.

ENABLE This command will enable the specified drive number and place its configuration information in the Drive Code table. If you enable a drive that has not been previously enabled or set up with the **SYSTEM (DRIVE=,DRIVER)** command, totally unpredictable results may follow.

STEP=n This parameter will set the stepping rate for the specified drive number, where n is a number 0 to 3. The following table lists the different stepping rates in ms (milliseconds) for 8" and 5.25" (and 3.5") drives. Do not select a step rate faster than your drive can handle. If in doubt, contact the drive manufacturer.

n	8" floppy step rate	5.25" floppy step rate
0	3 ms	6 ms
1	6 ms	12 ms
2	10 ms	20 ms
3	15/20 ms	30/40 ms

The 15/20 and 30/40 are dependent on whether you are using a double/single density disk controller chip. Version 6 Model 4 and version 5 Model III owners will have the 15ms and 30ms step rate, while Model I interface owners will have the 20ms and 40ms step rate. The fastest step rate for the Model I will be 12ms when using single density disks. Using a double density board in the Model I will provide the 6ms and 30ms step rates.

SYSTEM (DRIVE=d,DRIVER="filespec")

To access the disk drives, DOS will use information stored in memory in the Drive Code Table (DCT). No special configuration should have to be done unless drives other than 3.5" and 5.25" floppy drives are used. To configure the system for other drive types, it will be necessary to use this SYSTEM command.

The floppy/DCT program supplied (MOD1/DCT, MOD3/DCT, LX-80, MAX-80, or FLOPPY/DCT) will allow you to change the logical drive numbers for your floppy drives. This may be desirable when running hard disk systems.

SYSTEM (DRIVE=d,WP=sw)

This command will allow you to software write protect any or all drives currently enabled. Only one "DRIVE=d" parameter may be entered on the command line.

The parameters for this command are as follows:

d Designates the drive number affected.

sw Specifies the switch **ON** or **OFF**. "ON" will set the write protect status, and "OFF" remove it and allow the drive to be

written to. Setting WP to OFF will not over-ride any hardware write protection in effect (i.e. a write protect tab).

The command with no drivespec specified will act globally. That is, SYSTEM (WP=ON) will write protect all drives in the system, and SYSTEM (WP=OFF) will remove any software write protection that has been done on any drive. The WP=OFF parameter will have no effect on a disk physically protected with a write protect tab. Note that if the switch ON or OFF is not specified, ON is assumed.

SYSTEM (FAST) & SYSTEM (SLOW)

These commands are used only if a suitable clock speed-up modification has been installed in your computer unit - or it is designed to run at 4 MHz. They will modify certain timing loops in the DOS to accommodate the current processor clock speed, as well as switching the software controlled clock. These two commands have precedence over any other SYSTEM command, and will always be invoked before any of the other commands are carried out. No memory will be used by these parameters.

The version 5 Model III (FAST/SLOW) parameters will utilize the Model 4 hardware clock speedup while still maintaining an accurate time clock. Model III users should not specify (FAST) without an appropriate hardware speedup.

The version 5 Model I FAST/SLOW parameters will switch speed-up modifications addressed through port 254. The clock speed is controlled in the following manner:

- FAST issues an OUT Port 254,1 command.
- SLOW issues an OUT Port 254,0 command.

SYSTEM (GRAPHIC)

This command informs the DOS system that your line printer has the capability to directly reproduce the TRS-80 graphics characters during a screen print (version 5 screen print is enabled as a parameter of the KI/DVR program; version 6 screen print is a resident feature of DOS). If this parameter is used, any graphics characters on the screen will be sent to the line printer during a screen print command, either from the DOS level

or with version 5 BASIC's CMD"*". Do not use this parameter unless your printer is capable of directly reproducing the TRS-80 graphics characters.

SYSTEM (HERTZ5|HERTZ6)

This DOS version 6 parameter is used to adjust the software real time clock task for either 50 Hertz (HERTZ5) or 60 Hertz (HERTZ6) AC line frequency.

SYSTEM (PRTIME=ON|OFF)

This sub-command can enable or disable the printer time-out when line printer output is requested and the printer is unavailable. Specifying **SYSTEM (PRTIME=ON)** establishes an approximate 10-second time-out when the printer is unavailable (DOS version 6 generates the Device not available error code). Control will then be returned to whatever module called the printer driver. By specifying **SYSTEM (PRTIME=OFF)**, you will disable any time-out; if the printer is unavailable, the DOS will lock up! The default is OFF. **PRTIME** is configurable with the SYSGEN command. This facility is applicable to version 5 only when the PR/FLT is installed as it uses the printer driver resident within the filter.

SYSTEM (RESTORE(=ON|OFf)) (V6)

This DOS version 6 parameter alters the boot-up sequence to either restore all floppy disk drives to cylinder 0 (RESTORE=ON) or leave all floppy drives other than drive :0 at the cylinder of their last access. Restoring all floppy drives will speed up the first access performed.

SYSTEM (SMOOTH[=ON|OFF])

The **SMOOTH** sub-command is provided for faster disk I/O involving floppy drives aligned precisely to 300 rpm. If smooth is turned on, it alters the floppy disk driver so that the system interrupts are disabled earlier than what would otherwise occur. This has the effect of providing faster I/O with disk drives precisely aligned to 300 rpm where extra sector retries would be necessary and the drives may appear to "go to sleep". Note that when **SMOOTH** is turned **ON**, you will not be able to type ahead during disk I/O nor will you be able to effectively use dump-to-disk **ON** with (L)COMM even at 300 baud.

SYSTEM (SVC) [V5]

This DOS version 5 command will load a Supervisory Call (SVC) table into high memory. A description of the SVC table is beyond the scope of this Reference Manual. You must have set *KI to the KI/DVR program if you wish to use the SVC table.

SYSTEM (SWAP=s,DRIVE=d)

SWAP enables the swapping of any two logical drives by switching their Drive Code Table (DCT) assignments. Specifying **SYSTEM (DRIVE=d1,SWAP=d2)** switches drive d1 for d2. The command is functional while JCL is in execution even if one of the referenced drives holds the executing JCL file. If one of the designated drives is the SYSTEM drive and JCL is in execution, the other designated drive must contain a SYSTEM. Note that **SYSTEM (SYSTEM=d)** is equivalent to **SYSTEM (DRIVE=0,SWAP=d).**

SYSTEM (SYSGEN[=ON|OFF][,DRIVE=d])

This command creates or deletes a configuration file on the disk drive specified. If switch is not specified, **ON** is assumed. That is, **SYSTEM (SYSGEN)** is the same as **SYSTEM (SYSGEN=ON).** After a **SYSTEM (SYSGEN=OFF)** command has been given, the current configuration of the system will not change until the system is booted again.

When SYSGEN is invoked, all current device and driver configurations will be stored on the diskette in drive :0 or that drive specified. Job Control Language, if active, will be temporarily suspended during the configuration generation; SYSGEN will not abort if within a JCL file. SYSGEN will abort with an appropriate error if the drive specified to receive the configuration file is write protected. An invisible file named CONFIG/SYS will be created to hold the configuration. Each time the system is booted, the configuration stored in this file will be loaded and set. You may prevent this automatic configuration by holding down the <CLEAR> key while the boot is in progress. Note that the system configuration will take place before any AUTO'ed command is invoked.

In addition to the options settable by the SYSTEM command, the following will be stored in the configuration file by a SYSTEM (SYSGEN) command:

- All filtering, linking, routing, and device setting that has been done. This includes the serial driver and *KI settings.

- Any active background tasks (such as CLOCK, DEBUG, TRACE, etc).

- Any special utility modules or user programs loaded into high memory and protected with the MEMORY command. All memory from HIGH$ to the physical top of memory and added version 6 low memory modules will be written to the CONFIG/SYS file.

- The present state of VERIFY (either ON or OFF).

- All Device Control Blocks. This will include the current lines per page and line counter stored in the printer DCB.

- The state of the CAPS lock for the keyboard.

Certain DOS features should never be SYSGENed if a disk file is involved. They are any ROUTE or SET involving a disk file. SYSGENing open files can cause loss of data if the disks are switched in the drives without the files being closed. Disk switches with open files can also cause existing data to be overwritten.

SYSTEM (SYSRES=n)

This command will allow you to reside certain DOS system overlays in high memory - not the library modules. SYS files 1-5, and 8-12 [V5] or 9-12 [V6] may be loaded using this command. Note that each system overlay will require only the amount of high memory normally used by the overlay when it is resident in the system overlay region. This command does **not** allow multiple entries on the command line. For example, **SYSTEM (SYSRES=1,2,3)** will result in only SYS3 being made resident.

Having certain of these SYS overlays resident in memory will speed up most disk I/O operations, as these modules will not have to be loaded from disk. It will also allow you to purge these overlays from your system disk, providing more room for data and programs. Overlays 2, 3, 8 [for DOS version 5], and 10 must be resident for certain types of backups.

SYS2/SYS and SYS3/SYS must remain on any booting disk if a configuration file created with the SYSGEN parameter is to be loaded.

The DEVICE command will show any overlays that are currently resident in high memory.

SYSTEM (SYSTEM=n)

This command will allow you to assign a drive other than drive :0 as your system drive. It will do this by swapping the DCT (Drive Code Table) information of the drive specified with the current system drive. Note that there must be a diskette containing the necessary system files in the drive specified! If the designated drive does not contain the DOS and JCL is in execution, the command will abort, otherwise you will be prompted to enter a SYSTEM disk. This command is equivalent to **SYSTEM (SWAP=n,DRIVE=0)**.

Once this command has been invoked, DOS will look for any needed system files on the new system drive. Also, the defaults for number of cylinders and the bootstrap step rate use by the FORMAT utility will now be taken from the new system drive. If necessary, use the SYSTEM parameters "BSTEP" and "DRIVE=,CYL=" to establish these defaults on the new system disk. The logical drive numbers will also be changed - addressing drive :0 will now access the newly specified system drive, and vice versa.

This procedure may be repeated, and a swap of the current system drive with the drive specified will occur. The logical drive numbers will also change again. Be careful when repeating this command, or you may lose track of which drive is currently assigned to what logical drive number.

Note that doing a global RESET command will reset all drive DCTs to their default configurations. Be sure to have a system disk in physical drive :0 before performing a global RESET command.

SYSTEM (TIME=switch)

This command will enable or disable the prompt for the time on power up or reset. You must not have a write protected disk in drive :0 if using this command. The switch is either "ON" or "OFF" as follows:

ON Enables the time prompt on power up or reset. If the switch is not specified, **ON** is assumed.

OFF Disables the prompt for the time on power up or reset.

SYSTEM (TRACE[=ON|OFF])

This command will display the contents of the Z-80 processor Program Counter on the video display in the upper right corner. The display will be a hexadecimal address. Any information normally displayed on the top line will be overwritten by the trace display. The display is constantly updated as a high priority background task. The TRACE command is primarily useful during debugging of assembly language programs.

The trace display will halt if an assembly language program disables the interrupts, or if a BASIC program (Model I only) does a CMD"T". Doing a CMD"R" will restart the trace display.

The allowable commands are:

TRACE (ON) Turns the TRACE on.

TRACE (OFF) Turns the TRACE off.

TRACE Turns the TRACE on which is the default parameter.

Note: TRACE, along with some other operations, may not function properly on the Model III when the display is in the wide-character mode.

SYSTEM (TYPE=switch)

This command will turn the task processing of the keyboard driver type ahead feature either on or off. DOS version 5 must first have set *KI to the KI/DVR program specifying the **(TYPE)** parameter in order to gain access to type-ahead. If you wish to temporarily suspend the type ahead feature, use the **SYSTEM (TYPE=OFF)** command. This will turn off the type ahead processing without disturbing any other filters you may have applied to the keyboard. This will just inhibit the type-ahead operation; it does not remove the type-ahead task. The type ahead task processing may be restarted with the **SYSTEM (TYPE=ON)** command.

SYSTEM (UPDATE=switch) [V5]

The UPDATE sub-command [Version 5 Model I only] will allow the system date to be updated if the real time clock passes midnight (23:59:59). The date will advance one day and the day of the week and day of the year will also change. This routine will use some high memory. Due to hardware differences, this routine will not work on the Model III. The UPDATE module can be removed from memory.

The switches are ON or OFF, to enable or disable the update function. Doing a global RESET library command will disable the update function.

TAPE100

This DOS version 6 Model 4 program is used to transfer files to or from a Model 100 cassette tape. It's syntax is:

TAPE100 [file] [(Read | Write)]
TAPE100 [file1 [TO] file2] [(Read | Write)]

file, file2	Are each either a DOS filespec or a Model 100 filename.
Read	Specifies that you want to read a file (**file**) from tape and write it to a file (**file2**) on disk. If specified, you do not have to specify file1. DOS simply reads the first text file it sees on the tape.
Write	Specifies that you want to read a diskfile (**file**) and write it to a tape as **file2**.

You can use TAPE100 to read files from cassette tape as well as from Model 4 disks. The TAPE100 command allows you to:

- Read a cassette tape file and write it to a disk file; the cassette tape must have been made with the Model 100 computer.
- Read a disk file and write it to a cassette tape

A Model 100 filename is 1-6 alphanumeric characters long and it must begin with a letter. For example, ACCT61, LETTER, and ABFILE can be Model 100 filenames.

If you do not specify file or file2, you will be prompted to enter the source and destination filespecs if the operation is a WRITE, or just the destination filespec if the operation is a READ. If you specify neither READ or WRITE, you will be prompted for the operation. If the disk or tape file cannot fit in available memory, the error message "File too large to fit in available memory" will appear.

```
TAPE100 PRNTER TO PRINT/DAT:0 (READ) <ENTER>
```

In this example, TAPE100 will read the Model 100 file ***PRNTER*** and write it to the disk in drive :0 as ***PRINT/DAT***.

```
TAPE100 ACCTING/TXT:1 (READ) <ENTER>
```

Here, TAPE100 reads the first text file it finds on a Model 100 tape and writes it to the disk in drive :1 as ***ACCTING/TXT***.

```
TAPE100 WEST/DAT:0 TO WESTRN (WRITE) <ENTER>
```

In this example, TAPE100 reads the drive :0 disk file ***WEST/DAT*** and writes it to a file on a Model 100 tape named ***WESTRN***.

TED - ASCII Text Editor

The Text EDitor (TED) is a full screen "quick" text editor with typical word-processing type features (four-directional cursor movement; bi-directional scrolling; text directional delete; large text buffer; etc); however, TED was not designed to be a full featured word processor. TED was designed for you to be able to rapidly enter a full-screen text editing environment while accomplishing many of your text file editing tasks.

Summary of editing commands

The following are the command keys and their functions as supplied by TED. Once you become familiar with the operation of TED, this section may be all you need to refer to from time to time to jog your memory.

Action	Key Entry
Move the cursor one position left	[⇐] or [CTRL H]
Move the cursor one position right	[⇒] or [CTRL I]
Move the cursor one position down	[⇓] or [CTRL J]
Move the cursor one position up	[⇑] or [CTRL K]
Move the cursor to the beginning of the line	[SHIFT ⇐]
Move the cursor to the end of the line	[SHIFT ⇒]
Move the cursor to the end of the text	[SHIFT ⇓] [V6] or [SHIFT ⇓Z] [V5]
Move the cursor to the beginning of the text	[SHIFT UP]
Toggle overstrike/insert modes	[CTRL A]
Specify a BLOCK...	[CTRL B]
Specify DELETE	[CTRL D]
FILE the text buffer to disk	[CTRL F]
GO find the next search string match	[CTRL G]
LOAD a text file into the buffer	[CTRL L]
Command confirmation or advance to next line	[ENTER] or [CTRL M]
Go to the NEXT video page	[CTRL N]
PRINT the entire text buffer	[CTRL P]
QUERY a directory	[CTRL Q]
REPLACE searched string with new string	[CTRL R]
SEARCH for a string	[CTRL S]
Go UP to the previous video page	[CTRL U]
EXIT the text editor	[CLEAR SHIFT =]

Invoking TED

TED is invoked via the command:

```
TED  [filename[/TXT]]
```

TED will display a welcome message on the bottom line of the video screen. This display line will also be used for the display of status, prompting, and error messages. TED displays three different types of messages during its operation. Error messages are indicated by a terminating exclamation point, "!". Queries which need a response are indicated by a terminating question mark, "?". Informative messages use no special character for their termination. Thus, "Marker!" is an error, "String?" is a query, and "Block" is information.

Text entry modes

TED will accept only displayable ASCII characters in the range 20H through 7FH for text entry. Any other character value will be interpreted as a command entry. If it matches a value in the command table, that command will be invoked; otherwise, the entry will be ignored.

TED operates in two text entry modes: overstrike and insert. The initial mode established when TED is first invoked is the "*overstrike*" mode. While TED is in "*overstrike*" mode, it will use an underscore as the cursor character. When you toggle to "*insert*" mode, the cursor is changed to a full graphics block. You toggle from one mode to the other via the **<CTRL A>** command.

When TED is in "*overstrike*" mode, any acceptable text entry typed character is written over the character which appears under the blinking cursor. You can overstrike a newline character (i.e. **<ENTER>**, which is displayed as small graphics block). You can also overstrike either a "*begin*" block marker or an "*end*" block marker. You can be in overstrike mode when you come to the end of the text (or starting from an empty text buffer, for that matter) and still be able to enter text in this mode.

When you switch to "*insert*" mode, anytime you enter an acceptable text entry character, the entire text will be pushed down one position starting

from the character under the cursor to make room for the inserted character. The video screen will be constantly updated as text is inserted.

The text entry mode is only changed via the <CTRL A> command. Going into "delete" mode does not change the mode of text entry.

Loading a text file

The <CTRL L> command is used to load a text file into the text buffer area. When you depress <CTRL L>, you will be prompted for the name of the file with Load Filespec?. If the extension is omitted from your entry, "/TXT" will be automatically provided.

The LOAD command will not automatically clear any text remaining in the text buffer prior to the LOAD. The new text is not inserted at the cursor position but rather is appended to the end of the current text. If you wish to load the new file over the old text, simply invoke the command sequence, <SHIFT ⇑> followed by <CTRL D> then <SHIFT ⇓> for DOS 6 or <SHIFT ⇓ Z> for DOS 5. This will delete the entire text buffer.

If the file is too large to fit into the available text buffer, the error message "No room!" will be displayed and no text will be loaded. If any disk read error is encountered while reading the text file into the text buffer, the message "I/O error!" will be displayed. The text which was loaded up to the point of encountering the error will be retained in the text buffer.

Entering text

Entering text is easy, you just type away. If you already have text in the buffer and wish to enter new text at the end, just move the cursor to the bottom (via the <SHIFT ⇓> key for DOS 6 or <SHIFT ⇓ Z> for DOS 5), then type in your text. If you wish to enter new text at some other point, just position the cursor, toggle to the "insert" mode, then type away. TED will stay in *insert* mode until explicitly toggled back to *overstrike* mode.

As you are entering text, any word which is too long to fit at the end of a video line will be split at the 64th column and continued onto the next line. These "long words" are not automatically bounced onto the subsequent line, as is the case with the typical word processor.

Cursor positioning manipulations

The ARROW keys are the primary tools to move the cursor. They will move the cursor in the direction indicated by the arrow. The shifted arrow keys will be interpreted as cursor movement requests unless TED is in the DELETE or BLOCK modes.

The <SHIFT ⇐> request will move the cursor to the first position of the current line. The <SHIFT ⇒> request will move the cursor to the last position of the current line. You can position the cursor to the first position of the text buffer by a <SHIFT ⇑> request. Finally, the <SHIFT ⇓> for DOS 6 or <SHIFT ⇓ Z> for DOS 5 positions to the end of text.

The page up, <CTRL U>, command will refresh the video screen so that the new first displayed line is thirteen lines previous to the current first displayed line. The page next, <CTRL N>, request will refresh the video screen so that the new first line displayed is the last line of the current displayed text. If the video display has incomplete screen of text displayed, the page next request will be ignored.

Text deletion

TED provides five forms of text deletion in addition to the block deletion discussed later. To delete the single character which appears under the cursor, invoke the delete command via <CTRL D>. This action will get rid of the character and all text which succeeded that character will be pulled back one position. The <CTRL D> command also puts you into DELETE mode which is made apparent by the display of the word **"Delete"** in the status line. The DELETE mode is active for only the next keyboard entry. There are only four subcommands associated with the DELETE mode: delete to beginning of line (bol), delete to end of line (eol), delete to top, and delete to bottom. TED will always prompt before performing one of these deletes.

Deletion desired	Command sequence
delete to bol	<CTRL D> then <SHIFT ⇐>
delete to eol	<CTRL D> then <SHIFT ⇒>
delete to top	<CTRL D> then <SHIFT ⇑>
delete to bottom	<CTRL D> then <SHIFT ⇓><Z>

After typing <CTRL D>, the character now under the cursor is the character which was to the right of the deleted character. Since in the case of delete to bol and delete to top, you are deleting text which is in front of the cursor, you really don't want to delete the character which is under the cursor after the <CTRL D>. Well, you don't have to worry about that because those two subcommands properly backup one position before continuing the deletion.

Block operations

The BLOCK command, <CTRL B>, has six subcommands: Begin, End, Copy, Delete, Move, and Print. These subcommands are specified by entering the first letter of the subcommand word (<B>, <E>, <C>, <D>, <M>, or <P>). The entry may be in either upper or lower case. Note that these subcommands are **not** control key combinations but normal alphabetic single-key entries. When you invoke the BLOCK command, the word Block will be displayed in the status line.

Anytime you need to deal with a block; say to copy it, move it, or delete it, you have to first mark it. The beginning and ending positions of a block are marked by first positioning the cursor over the first character of the block and then entering the two command sequence, <CTRL B> followed by <B>. This is followed up be positioning the cursor over the character immediately following the last character of the block and then entering the two command sequence, <CTRL B> followed by <E>. The beginning position will be indicated on the display by a "begin" marker which is inserted by TED into the text. The marker is displayed as a graphic left bracket. The ending position will be indicated on the display by an "end" marker which is also inserted by TED into the text. The marker is displayed as a graphic right bracket. These markers occupy ordinary text positions; thus they may be deleted or overstriked. Any remaining in the text buffer at the time a FILE command is performed will be written to the disk file just as if they were ordinary text characters.

Although you can mark as many blocks as your heart desires, TED provides no way to differentiate between marked blocks in other than the BLOCK-DELETE function. For copying and moving blocks, the first block marked in the text is the one chosen for copying or moving. On the other hand, a BLOCK-DELETE request requires that the cursor be positioned within the interior of the marked block which is to be deleted.

To COPY the first marked block in the text to some other position, simply mark the beginning and end of the block as discussed above, move the cursor to the position in the text where you want the marked block copied into, then invoke the block copy command via the sequence, <CTRL B> followed by <C>. Note that the block which will be copied is the first marked block found in the text buffer. A few things could go wrong with your request. If TED can find no properly marked block, it will display the error message Marker! and terminate the block mode. Another error which could occur is when the position you wish the block copied into happens to be in the interior of the block itself! You will be informed of this prohibition by a display of the error message Cursor!.

The successful block copy operation only copies the marked text; the markers are not copied as well. In fact, the marked text remains in its original position relative to the text which surrounds it. The cursor position relative to the text will be unchanged after the block is copied; however, the screen may be refreshed and the physical location of the cursor on the screen may be different.

A block of text may be MOVED from one position to another by a command sequence similar to the block copy. In this case, simply mark the beginning and end of the block as discussed above, move the cursor to the position in the text where you want the marked block moved to, then invoke the block MOVE command via the sequence, <CTRL B> followed by <M>. Again note that the block which will be moved is the first marked block found in the text buffer. This operation is essentially one of copying and automatic deleting without the double check prompt. As in the case of the BLOCK-COPY, the same errors are possible with similar diagnostic messages when things are not as they should be. With the BLOCK-MOVE command, the new cursor position will be the new position of the moved block. The screen may be refreshed and the physical cursor position altered to accommodate this request.

The block operation, deletion, is similar to the above functions, you first must mark the block's beginning and ending positions. You must then position the cursor to the interior of the marked block and invoke the command with the sequence, <CTRL B> followed by <D>. As a safety check, TED will prompt you before deleting the block. It is necessary to depress <ENTER> to affirm your intentions. Any other character entry (including a <Y>) will cause TED to ignore the block delete request.

The same errors as for copy and move can occur; however, the messages may not be for the same reasons. When a block delete is requested, TED will first look for an ending block marker starting from the cursor position. If none is found, the error displayed will be Marker!. This doesn't mean necessarily that a properly marked block is missing. On the other hand, if an ending marker is found past the cursor position, TED next scans forward for a beginning block marker. A Marker! error will also be posted if none is found. If a marker is found but is also past the cursor position, a Cursor! error will be posted.

The PRINT block operation will cause the first marked block to be printed.

Filing away your text to a disk file

The <CTRL F> command is used to FILE the contents of the text buffer area into a disk file. When you depress <CTRL F>, you will be prompted for the name of the file with Filespec?. If the file specification you wish to use has an extension of "/TXT", you do not have to enter the extension. If the extension is omitted from your entry, "/TXT" will be automatically provided.

The FILE command will save the entire text buffer, excluding the terminating NULL but including any block markers, into the disk file identified by your input. If any disk write error is encountered while saving the text buffer into the disk file, the message I/O error! will be displayed. In any case, the text buffer is left undisturbed.

Text search

TED provides the SEARCH command to scan the text buffer for a specified string of characters. You specify the search by invoking the command with <CTRL S>. TED then prompts you for the search string with the query message String?.

You can enter up to 23 characters to be used for the search string. Terminate your search string with an <ENTER> which is not included as one of the 23 characters. TED will then look for the string starting with the first character following the cursor. The matching is case sensitive which means that characters entered in upper case must be found in upper case and characters entered in lower case must be found in lower case. If the search string cannot be found, the message Can't! will be displayed. At this

point, the cursor location remains unchanged. If a matching string of text is found in the text buffer, it will be displayed. The display window will be redrawn starting with the line which contains that string. The cursor will be repositioned to the first character of the matching string.

If you press <ENTER> only in response to the String? query, then the search will proceed with the last entered search string, providing one was available. Using this procedure, you can advance the cursor to each occurrence of the search string in question.

Another way to find each occurrence of a search string is with the GO command, <CTRL G>. Each depression of <CTRL G> is identical to the sequence, <CTRL S> followed by <ENTER>.

Text search and replace

TED also provides the capability of replacing a text string matching the search string with a different string - the replacement string. When the REPLACE command is invoked via <CTRL R>, the query message String? will be displayed. The query, same as for SEARCH, is asking you for the replacement string. You can enter up to 23 characters to be used for the replacement. Terminate your string with an <ENTER> which is not included as one of the 23 characters. TED will then look for the currently pending SEARCH string starting with the first character **immediately** under the cursor. If the SEARCH string cannot be found, the message Can't! will be displayed. At this point, the cursor location remains unchanged. If a matching string of text is found in the text buffer, it will be replaced with the REPLACE string. The display window will be redrawn starting with the line which contained the string which was replaced. The cursor will be repositioned to the first character immediately following the replacement string.

If you wish to replace the next occurrence of text which matches up with the SEARCH string with that same REPLACEMENT string, all you need do is <CTRL R> ENTER.

The GO command, <CTRL G>, still functions to find the next occurrence of the SEARCH string. If that occurrence of the string is beyond the text currently displayed on the screen and you wish to confirm its replacement, simply GO to the next occurrence then REPLACE, as necessary.

Printing text

TED provides the <CTRL P> command to print the entire text buffer. If you want to print just a block of text, use the BLOCK PRINT command.

Obtaining a Directory

TED provides the <CTRL Q> command to obtain a directory of files. After the Drive? prompt, enter the desired drive specification as either ":d" or "d". You may also restrict the display to files matching a particular file extension by entering a 4-character sdtring preceding the drive, as in "/TXT:1"; a dollar sign in any extension character position designates a match on any character. Depressing <ENTER> after a directory query will restore ther text screen image.

Exiting from TED

To exit TED and return to DOS Ready, use the <CLEAR SHIFT => command. If the text buffer is empty, TED will immediately terminate. However, if there is any text in the buffer, you are provided an opportunity to retract your request. TED will display the prompt message,

Press ENTER to exit

Text recovery

If you exit the TED application inadvertently without saving the edited text, TED permits you to re-enter with the asterisk parameter which provides a chance to recover your text. Instead of automatically clearing the text buffer as TED does, **TED *** will display whatever is in the text buffer memory area. Thus, if you have not altered any of the information in that memory area, you can always go back and recapture it.

Since all of the text pointers normally established by TED will not be initialized when invoking TED via the * parameter, it will be necessary to scroll through the text until reaching its last character prior to doing any other operation. This may also be performed using NEXT PAGE.

TIME

This command is used to set the time for the "real time" clock.

TIME [hh:mm:ss (CLOCK[=ON|OFF])]	
hh:mm:ss	Sets time to **hh** hours (00-23), **mm** minutes (00-59), and **ss** seconds (00-59); "ss" defaults to 00 if omitted.
CLOCK=	Enables or disables the video clock display.

TIME is used to adjust the time kept by the system's real time clock. You can also be prompted on power up or reset for the time. This prompt may be enabled and disabled with the **SYSTEM (TIME=)** library command. The clock function is normally controlled by hardware circuits in the expansion interface (Model I) or by a signal developed from the AC power line (Models III and 4). This time is not an actual "real time" clock, as the clock referred to is used by many different software and hardware devices. Certain system operations require that the clock be turned off altogether. You are advised not to depend on the clock for constantly accurate time and date information but only as a relative time of day indicator. A hardware clock option us useful for keeping accurate time.

Issuing a **TIME** command with no parameters will display the current setting of the clock. The clock will be reset to 00:00:00 every time you power up, press the reset button, or issue a BOOT library command.

To set the clock, use the command **TIME hh:mm:ss**, specifying the hours, minutes, and seconds desired, with the seconds being optional. The latest time acceptable is 23:59:59, as the clock will always run in the 24 hour mode.

The time lag between pressing the **<ENTER>** key and the time that will actually be set on the clock will be approximately two seconds (the time needed to invoke the **TIME** command). It is usually best to type in the command **TIME** and then the time plus several seconds after the correct

time. Wait for "seconds-2" to come up on your watch and press <ENTER>. This will give you the correct time on the clock.

There are several ways for application programs to retrieve the current time setting of the clock. At an assembly language level, a call to the @TIME service function will return the time. When using BASIC, the time can be returned through the TIME$ variable.

The time may be constantly displayed on the video screen by issuing a **CLOCK** command or with the <C> key function of the version 5 MiniDOS keyboard filter program. Either of these commands will enable or disable the clock display in the upper right hand corner of the screen.

On the Model III, the BASIC commands CMD"R" and CMD"T" will also turn on and off the clock display. The clock on the Model III may not be accurate when running with 50 Hertz AC power.

The clock paramater is used to turn on or off the screen display of the real time clock. When you enter this command it will activate a background task and display the clock time in the upper right corner of the screen. This will take precedence over whatever DOS or BASIC may attempt to print at the screen locations occupied by the display.

The clock will only run in the 24 hour mode. The date can automatically be updated when the clock passes midnight on the Model I by using the **SYSTEM (UPDATE)** command. The Model III time and date routines are in ROM, and cannot be made to update the date automatically. The initial date value is normally prompted for when powering up the system. The time and date values may also be set with the TIME and DATE commands

The real time clock may be turned off while DOS is doing some of its critical disk I/O functions, such as when using the BACKUP and FORMAT utilities. You will be notified of this by this message which notifies you that the real time clock has lost several seconds or more:

```
Note: Real time clock no longer accurate
```

The SETTIME program may be provided to allow you to set or read a hardware clock. Invoke SETTIME with a command such as: **SETTIME HH:MM:SS**. Any additional instructions needed will be provided after SETTIME invokes.

TOF

This command will emit a form feed character (12d) to the *PR device. If the printer is currently unavailable, the command does nothing. Its syntax is:

```
TOF
```

VERIFY

The VERIFY command forces all disk writes to be verified with a read-after-write operation. The syntax is:

> **VERIFY [(switch)]**
>
> **switch** is the parameter **ON** or **OFF**, "ON" is the default.

VERIFY will determine whether or not writes to a disk file are verified with a read-after-write operation. The state of the **VERIFY** command may be saved in the configuration file with the **SYSTEM (SYSGEN)** library command. The normal power up condition is with verification turned off. To cause a read after write verify of every write operation, you must specify the command **VERIFY** or **VERIFY (ON)**. The command **VERIFY (OFF)** will disable the read-after-write verification.

The **VERIFY** command works by having the disk controller re-read the sector just written and note that the read was successful. The disk controller will normally compare the Cylical Redundancy Checksum (CRC) determined from the read with the CRC recorded when the sector was written. It does not do a byte for byte verify on the information in the disk sector. Anytime that an error is detected, the appropriate error message will be displayed.

Although having the VERIFY function turned on will provide the greatest reliability during disk I/O, it will also increase the overall processing time whenever a disk file is written to. The user must determine if the increase in reliability warrants the increase in processing time.

All disk writes will automatically be verified during any BACKUP utility function, whether the VERIFY command has been issued or not. Also, certain critical writes to system tables and any write to the directory will always be verified.

DOS Drivers and Filters

CLICK/FLT

This DOS version 6 filter can be used to produce a tone from the sound generator inside your computer whenever a keyboard key is pressed. It is installed with the syntax:

SET devspec CLICK/FLT [(Char=nn)]
FILTER *KI devspec

devspec Is the device name you assign to the filter.

Char=nn Specifies that the click should be generated on each entry of nn rather than all keystrokes. nn represents the character and is in the range 1 to 255.

After you install the filter, the sound generator produces a tone each time you press a key on your keyboard; this provides an auditory feedback.The sound that CLICK produces when the system is running at SLOW speed is different from that produced when the system is running at FAST speed.

You can change the pitch and duration of the tone by applying a patch to the values that produce the tone. This patch is:

PATCH CLICK/FLT.FILTER (D00,A0=ddpp:F00,A0=1848)

dd is a hexadecimal value specifying duration in the range 01 to FF; 01 produces the shortest duration, and FF produces the longest. pp is a hexadecimal value specifying the pitch of the tone in the range 01 to FF; 01 produces the highest pitch and FF produces the lowest pitch.

SET *CK CLICK (C=13) <ENTER>
FILTER *KI *CK <ENTER>

This command set installs the filter and associates it with the keyboard device. Each time the **<ENTER>** key is pressed, CLICK produces a tone.

When you install CLICK/FLT, DOS will place it in the low memory I/O driver region if space is available. If not, DOS will place it in high memoryand then issue the following informative message:

Note: filter installed in high memory.

If you want to use MemDISK while you are using CLICK/FLT, you should install MemDISK first.

COM and RS232x

These programs are drivers for the RS-232 serial hardware. They allow you to set your RS-232 parameters to values that match other RS-232 devices; parameters for COM/DVR are set with the SETCOM command.

SET devspec [TO] serial[/DVR] (parm,parm,...)

devspec	Is the device to be used with the RS-232, normally *CL, or the Comm Line.
serial	Model I = RS232R, LX-80 = RS232L; Model III = RS232T; MAX-80 = RS232M; Model 4 = COM
Baud=	Sets the BAUD rate to any supportable rate.
BREAK	Determines the received character, if any, acepted as a system BREAK.
Parity=	Sets the PARITY switch, ON or OFF. If ON is specified, EVEN or ODD may also be used.
PORT =	0 or 1, to select the A or B channel.
Stop=	sets the stop bits, either 1 or 2.
Word=	sets the word length, 5 to 8 bits.
DTR=sw	Data Terminal Ready
RTS=sw	Request To Send
CD=sw	Carrier Detect
CTS=sw	Clear To Send
DSR=sw	Data Set Ready
RI=sw	Ring Indicator

The RS232R driver program will accept and configure the RS-232 hardware in the Model I Radio Shack interface; RS232L is for the LX-80. The RS232T driver program will accept and configure the RS-232

hardware in the Model III. The COM driver program is for the Model 4. All are installed using the SET command

The defaults for the Model I configuration parameters will be the switch settings on the RS-232 board. If a parameter is specified when setting the driver, it will override the switch setting.

The defaults for Model III and 4 configuration parameters are: BAUD = 300; WORD = 7; STOP = 1; PARITY = ON,EVEN. The receiving side of the driver is interrupt driven and contains an internal 128 character buffer to prevent loss of characters during disk I/O and other lengthy operations.

The serial driver provides for the selection of a received logical BREAK character rather than arbitrarily using a code of 01D or 80H. For DOS version 5, **BREAK** normally defaults to "OFF"; thus, no received character will be interpreted as a system BREAK. For DOS version 6, BREAK defaults to **<CTRL><C>**. In all cases, the logical BREAK, if any, will be converted to the system BREAK for the partitcular DOS: 01D for version 5; 80H for version 6. The driver parameters may be altered after the driver is installed by using the **SETCOM** command documented earlier.

The Line Condition parameters have been provided so that you may set up the conventions required by most communicating devices. As specified by standard RS232 conventions, a TRUE condition means a logic 0, or positive voltage. A FALSE condition means a logic 1, or negative voltage. DTR and RTS may be set to a constant TRUE by specifying the ON switch. If DSR, CD, CTS, or RI are specified ON, the driver will observe the lead and wait for a TRUE condition before sending each character. If specified OFF, the driver will wait for a FALSE condition before sending a character. If not specified, the lead will be ignored.

<BREAK>, PAUSE "**<SHIFT><@>**", and **<ENTER>** characters received from the communication's line will be recognized by DOS. This would be useful in "host" type applications. With BREAK on, the system break bit will be set whenever a modem break (extended null) or logical BREAK is received. If BREAK is specified OFF, the driver will never set the break bit. Regardless of the setting of BREAK, the system pause bit will be set when the ASCII code X'60' is received, and the system "enter" bit will be set whenever a carriage return (X'0D') is received.

The following example sets the driver program for a serial printer.

```
SET *CL TO COM/DVR                                    [V6]
SET *CL RS232T/DVR                                    [V5]
SETCOM (BAUD=300,WORD=8,STOP=1,CTS)
ROUTE *PR *CL
```

This example did not specify PARITY; thus, it will use the default value. CTS was specified, so CTS must be a TRUE condition before sending a character. This would be useful when using a serial printer with its BUSY line hooked to the CTS line on the computer; characters would be sent to the printer only when the printer was ready to accept them.

The *CL device will usually be used to communicate with the RS-232 hardware. However, when using a serial printer, the *PR device would normally be used when setting the RS-232 driver.

```
SET *CL TO COM/DVR                    (V6)
SETCOM (BREAK)

SET *CL RS232T (BREAK)                (V5)
```

This example will restore the default values. Because "BREAK" was specified, certain system functions will recognize "break", "pause", or "enter" characters from the RS-232 as if they came from the keyboard.

```
SET *CL COM                           (V6)
SETCOM (DTR,CTS,BREAK)

SET *CL RS232T (DTR,CTS,BREAK)        (V5)
```

This example is identical to the previous, except that the DTR line will be held in a constant TRUE state, and the driver will not transmit any characters unless the external device raises the CTS line.

```
SET *CL RS232T (W=7,P=ON,EVEN)
```

This example will set the word length to 7, and set parity ON and EVEN.

FDUBL

FDUBL is a disk driver program for use with the Model I, 5.25" drives, and a double density modification board such as a Percom-type (Aerocomp DDC) or a Radio Shack board. The syntax is:

FDUBL (Tandy | Percom)	
Percom	Parameter to designate a Percom-type doubler
Tandy	Parameter to designate a Radio Shack doubler

This command loads a special disk driver program which allows you to use a double density hardware modification to read, write, and format double or single density 5.25 disks with the Model I. Before buying a double density board, please check with the manufacturer or DOS Support to assure compatibility with the DOS FDUBL driver.

If you have a doubler installed, after you give this command, you can use either single or double density disks in any of your 5.25" disk drives. DOS will automatically recognize whether you have a single or double density diskette in a drive, and react accordingly. Once you have installed the FDUBL driver, you will see the prompt "Single or Double density <S,D> ?" appear after you enter the disk name and master password during the disk FORMAT utility. Answer this prompt by pressing the **<D>** key to create a double density diskette or **<S>** to create a single density diskette. Pressing **<ENTER>** for this prompt will default to double density.

The FDUBL driver is loaded into high memory and protects itself by lowering the value stored in the HIGH$ memory pointer. Logical drives 0-7 are set up to use this driver in place of the normal DOS single density driver. You can use the **SYSTEM (SYSGEN)** command to save the driver in your configuration file, to be loaded automatically every time you boot. Be sure that any application programs you are using respect the HIGH$ pointer.

JOBLOG

This driver program will establish the DOS Joblog device. The syntax is:

ROUTE *JL [TO] filespec I devspec	[V6]
SET *JL TO JL[/DVR] [USING] filespec/devspec	[V5]
filespec devspec	The file or device to be sent the Joblog information.

The JL/DVR program [DOS version 5] or resident job log driver [DOS version 6] will establish the DOS Joblog device (*JL). Once set, a log of all commands entered or received will be sent to the specified file or device, along with a time stamp. Note that the time stamp will be determined from the setting of the system's real time clock (see the TIME command). If a filespec is used, the default extension will be "/JBL".

Setting *JL will use high memory. The **RESET *JL** command will terminate the JobLog, and close any associated disk file. If *JL is set again, the previous high memory allocation will be re-used.

To view the contents of a JobLog disk file, you must first **RESET *JL**, so the file will be closed. You may wish to add a trailing exclamation point "!" to the end of the filespec, so that constant EOF maintenance will be invoked (see the filespec definition in the *GLOSSARY*). The LIST command will allow you to list the contents of the file to the screen or to the printer.

Note that if an existing filespec is used when setting *JL, any information sent to the JobLog file will be appended to the end of the file.

You may wish to send the information to a device such as *PR, rather than a file. In this case, a devspec rather than a filespec would be used in the command line when setting *JL to its driver.

KI/DVR

The DOS Version 5 KI/DVR program will enable enhanced keyboard features. The syntax is:

SET *KI TO KI/DVR (Type,Jkl,Delay=d,Rate=r)	
Delay=d	Sets the delay until the first repeat to **d**.
Jkl	Ativates the screen print option.
Rate=r	Sets the repeat rate to **r**.
Type	Activates the type ahead feature.

Among other things, the KI/DVR program establishes the <CLEAR> key as a special control key for many DOS functions. This driver must be set if SPOOL, SYSTEM (SVC), KSM, MiniDOS, LCOMM, or any other program that utilizes the <CLEAR> key as a control key is to be used!

On the Model III, the keyboard repeat and debounce features are part of the ROM keyboard driver, and will be available even if this driver is not used. However, using the KI/DVR program will provide an increased key repeat rate.

On the Model I, the keyboard will use the ROM driver on power up. You will not have key repeat or debounce unless KI/DVR has been set.

The driver allows the key combination, <CLEAR><SHIFT><0>, to generate a code of 160d - the same as <CLEAR><SPACE>. Note that it will not toggle CAPS LOCK; only <SHIFT><0> will toggle CAPS LOCK.

If Model III DOS is running on a Model 4 computer, installing the KI/DVR will automatically load a driver which supports the <CTRL> key, the function keys, and <CAPS> in lieu of <SHIFT><0>.

As this driver is established with the SET Library command, it must be applied before any other *KI filters. When KI/DVR is set, the driver will reside in high memory. Once KI/DVR is set, you will not be allowed to set

it again with additional parameters without first doing a RESET *KI command. For example, if you had initially SET *KI TO KI/DVR, and later wish to initialize the type ahead feature, you must first RESET *KI, and then SET *KI TO KI/DVR (TYPE). The only additional memory used will be the amount needed for the type ahead option. The original memory used for the KI/DVR will be reused for the original KI/DVR functions.

Specifying the TYPE parameter enables the "type ahead" feature. This will provide a 128 character buffer, and will allow typing ahead even when the system is performing other functions such as disk I/O. If you make a mistake while typing ahead, pressing the <SHIFT><⇐> will erase the current line. Pressing <CLEAR><@> will empty the entire type ahead buffer. To temporarily disable the type ahead function, use the command SYSTEM (TYPE=OFF). It may be re-enabled with a SYSTEM (TYPE=ON) command.

The screen print option will send the contents of the video screen to *PR (usually a line printer) whenever the <LEFT SHIFT><⇓><*> keys are pressed on a Model I or III, or <CTRL><:> on a Model 4. Characters outside the ASCII range are normally translated to periods. However, the GRAPHIC parameter of the SYSTEM command will allow graphics characters to be sent to the line printer during a screen print.

The "DELAY" and "RATE" parameters deal with the keyboard repeat function. DELAY sets the initial delay between the time a key is first pressed and the first repeat of that key. It can be any value 10 or greater. The default is 30, and provides a delay of about 3/4 of a second. The RATE parameter sets the rate of key repeat, and can be any value 1 or greater. The default is 3, and provides a repeat rate of about 10 per second. Using DOS version 6, you can change these parameters with the SETKI command.

Keyboard equivalents

When KI/DVR is set, the TRS-80 keyboard will be able to produce the entire ASCII character set from X'00' to X'7F' (0 to 127). Keys not normally accessible can be entered as described in the tables on pages 259-260.

Extended Cursor Mode (ECM)

For DOS version 5, many applications programs use the four arrow keys to control cursor motion. However, this precludes entering an X'5B' character, as this is the value returned by the <↑>. To allow the full ASCII character set to be used by an application, the ECM will change the values returned by the four arrow keys. When in the ECM, it will be necessary to use control keys to perform the original arrow functions. <CTRL><H> will perform a backspace, <CTRL><I> a tab, et cetera. Pressing an arrow key will display the corresponding graphics character. The ECM will primarily be useful for applications programs that do their own cursor control and also require that the full ASCII character set be available.

Forcing CAPS lock or unlock

An application program can force the keyboard input to be in either the CAPS lock mode or the normal upper/lower case mode without having the operator press the <SHIFT><0>. Assembly language programs can set bit 5 of KFLAG$ (Mod 1=X'4423', Mod 3=X'429F') to force CAPS lock, or reset that bit for normal upper/lower case entry. From a BASIC program, the POKE statement can be use. For the Model I, the statement:

```
POKE(&H4423),PEEK(&H4423) OR 32
```

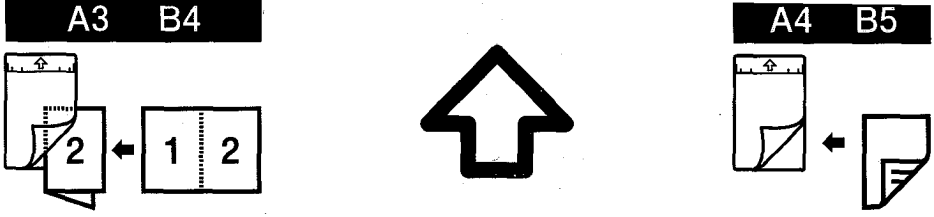
will force CAPS lock, and the statement:

```
POKE(&H4423),PEEK(&H4423) AND 223
```

will force normal upper/lower case entry. For a Model III, the address to use would be &H429F. The logical AND and OR assure that only the CAPS lock bit of the memory location will be changed.

Using DOS version 6, you will have to access the "K" flag using a DOS service call.

DEC	HEX	TAG	ENTERED BY	DEC	HEX	TAG	ENTERED BY
0	00	NUL	CTL <@>	32	20	SPA	<SPACE>
1	01	SOH	CTL <A>	33	21	!	SH <1>
2	02	STX	CTL <B>	34	22	"	SH <2>
3	03	ETX	CTL <C>	35	23	#	SH <3>
4	04	EOT	CTL <D>	36	24	$	SH <4>
5	05	ENQ	CTL <E>	37	25	%	SH <5>
6	06	ACK	CTL <F>	38	26	&	SH <6>
7	07	BEL	CTL <G>	39	27	'	SH <7>
8	08	BS	CTL <H> (1)	40	28	(	SH <8>
9	09	HT	CTL <I> (2)	41	29	)	SH <9>
10	0A	LF	CTL <J> (3)	42	2A	*	SH <:>
11	0B	VT	CTL <K>	43	2B	+	SH <;>
12	0C	FF	CTL <L>	44	2C	,	<,>
13	0D	CR	CTL <M> (4)	45	2D	-	<->
14	0E	SO	CTL <N>	46	2E	.	<.>
15	0F	SI	CTL <O>	47	2F	/	</>
16	10	DLE	CTL <P>	48	30	0	<0>
17	11	DC1	CTL <Q>	49	31	1	<1>
18	12	DC2	CTL <R> (14)	50	32	2	<2>
19	13	DC3	CTL <S>	51	33	3	<3>
20	14	DC4	CTL <T>	52	34	4	<4>
21	15	NAK	CTL <U>	53	35	5	<5>
22	16	SYN	CTL <V>	54	36	6	<6>
23	17	ETB	CTL <W>	55	37	7	<7>
24	18	CAN	CTL <X> (5)	56	38	8	<8>
25	19	EM	CTL <Y> (6)	57	39	9	<9>
26	1A	SUB	CTL <Z>	58	3A	:	<:>
27	1B	ESC	CTL <,> (7)	59	3B	;	<;>
28	1C	FS	CTL </>	60	3C	<	SH <,>
29	1D	GS	CTL <.> (9)	61	3D	=	SH <->
30	1E	RS	CTL <;>	62	3E	>	SH <.>
31	1F	VS	SH <CLR>	63	3F	?	SH </>



Set Generation

DEC	HEX	TAG	ENTERED BY	DEC	HEX	TAG	ENTERED BY
64	40	@	<@>	96	60	`	SH <@>
65	41	A	SH <A>	97	61	a	<A>
66	42	B	SH <B>	98	62	b	<B>
67	43	C	SH <C>	99	63	c	<C>
68	44	D	SH <D>	100	64	d	<D>
69	45	E	SH <E>	101	65	e	<E>
70	46	F	SH <F>	102	66	f	<F>
71	47	G	SH <G>	103	67	g	<G>
72	48	H	SH <H>	104	68	h	<H>
73	49	I	SH <I>	105	69	i	<I>
74	4A	J	SH <J>	106	6A	j	<J>
75	4B	K	SH <K>	107	6B	k	<K>
76	4C	L	SH <L>	108	6C	l	<L>
77	4D	M	SH <M>	109	6D	m	<M>
78	4E	N	SH <N>	110	6E	n	<N>
79	4F	O	SH <O>	111	6F	o	<O>
80	50	P	SH <P>	112	70	p	<P>
81	51	Q	SH <Q>	113	71	q	<Q>
82	52	R	SH <R>	114	72	r	<R>
83	53	S	SH <S>	115	73	s	<S>
84	54	T	SH <T>	116	74	t	<T>
85	55	U	SH <U>	117	75	u	<U>
86	56	V	SH <V>	118	76	v	<V>
87	57	W	SH <W>	119	77	w	<W>
88	58	X	SH <X>	120	78	x	<X>
89	59	Y	SH <Y>	121	79	y	<Y>
90	5A	Z	SH <Z>	122	7A	z	<Z>
91	5B	[	CLR <,> (8)	123	7B	{	CLR SH <,>
92	5C	\	CLR </>	124	7C	|	CLR SH </>
93	5D	]	CLR <.>	125	7D	}	CLR SH <.>
94	5E	^	CLR <;>	126	7E	~	CLR SH <;>
95	5F	_	CLR <ENTER>	127	7F	DEL	CLR SH <ENTER>

ASCII Characters

DEC	HEX	TAG	ENTERED BY
0	00	NUL	CTL <@>
1	01	SOH	CTL <A>
2	02	STX	CTL <B>
3	03	ETX	CTL <C>
4	04	EOT	CTL <D>
5	05	ENQ	CTL <E>
6	06	ACK	CTL <F>
7	07	BEL	CTL <G>
8	08	BS	CTL <H>
9	09	HT	CTL <I> (1)
0A		LE	
0B		VT	
C		FF	

DEC	HEX	TAG	ENTERED BY
32	20	SPA	<SPACE>
33	21	!	SH
34	22	"	
35	23		
36	24		
37			

Generation

ERED BY (11)
7 (12)

DEC	HEX	TAG	ENTERED BY
224	E0		CLR SH <@> [V6]
225	E1		CLR SH <A>
226	E2		CLR SH <B>
227	E3	(13)	CLR SH <C>
	E4	(13)	CLR SH <D>
			CLR SH <E>
			CLR SH <F>
			R SH <G>
			<H>

DEC	HEX	TAG	ENTERED BY	DEC	HEX	TAG	ENTERED BY
0	00	NUL	CTL <@>	32	20	SPA	<SPACE>
1	01	SOH	CTL <A>	33	21	!	SH <1>
2	02	STX	CTL <B>	34	22	"	SH <2>
3	03	ETX	CTL <C>	35	23	#	SH <3>
4	04	EOT	CTL <D>	36	24	$	SH <4>
5	05	ENQ	CTL <E>	37	25	%	SH <5>
6	06	ACK	CTL <F>	38	26	&	SH <6>
7	07	BEL	CTL <G>	39	27	'	SH <7>
8	08	BS	CTL <H> (1)	40	28	(	SH <8>
9	09	HT	CTL <I> (2)	41	29	)	SH <9>
10	0A	LF	CTL <J> (3)	42	2A	*	SH <:>
11	0B	VT	CTL <K>	43	2B	+	SH <;>
12	0C	FF	CTL <L>	44	2C	,	<,>
13	0D	CR	CTL <M> (4)	45	2D	-	<->
14	0E	SO	CTL <N>	46	2E	.	<.>
15	0F	SI	CTL <O>	47	2F	/	</>
16	10	DLE	CTL <P>	48	30	0	<0>
17	11	DC1	CTL <Q>	49	31	1	<1>
18	12	DC2	CTL <R> (14)	50	32	2	<2>
19	13	DC3	CTL <S>	51	33	3	<3>
20	14	DC4	CTL <T>	52	34	4	<4>
21	15	NAK	CTL <U>	53	35	5	<5>
22	16	SYN	CTL <V>	54	36	6	<6>
23	17	ETB	CTL <W>	55	37	7	<7>
24	18	CAN	CTL <X> (5)	56	38	8	<8>
25	19	EM	CTL <Y> (6)	57	39	9	<9>
26	1A	SUB	CTL <Z>	58	3A	:	<:>
27	1B	ESC	CTL <,> (7)	59	3B	;	<;>
28	1C	FS	CTL </>	60	3C	<	SH <,>
29	1D	GS	CTL <.> (9)	61	3D	=	SH <->
30	1E	RS	CTL <;>	62	3E	>	SH <.>
31	1F	VS	SH <CLR>	63	3F	?	SH </>

Set Generation

DEC	HEX	TAG	ENTERED BY	DEC	HEX	TAG	ENTERED BY
64	40	@	<@>	96	60	`	SH <@>
65	41	A	SH <A>	97	61	a	<A>
66	42	B	SH <B>	98	62	b	<B>
67	43	C	SH <C>	99	63	c	<C>
68	44	D	SH <D>	100	64	d	<D>
69	45	E	SH <E>	101	65	e	<E>
70	46	F	SH <F>	102	66	f	<F>
71	47	G	SH <G>	103	67	g	<G>
72	48	H	SH <H>	104	68	h	<H>
73	49	I	SH <I>	105	69	i	<I>
74	4A	J	SH <J>	106	6A	j	<J>
75	4B	K	SH <K>	107	6B	k	<K>
76	4C	L	SH <L>	108	6C	l	<L>
77	4D	M	SH <M>	109	6D	m	<M>
78	4E	N	SH <N>	110	6E	n	<N>
79	4F	O	SH <O>	111	6F	o	<O>
80	50	P	SH <P>	112	70	p	<P>
81	51	Q	SH <Q>	113	71	q	<Q>
82	52	R	SH <R>	114	72	r	<R>
83	53	S	SH <S>	115	73	s	<S>
84	54	T	SH <T>	116	74	t	<T>
85	55	U	SH <U>	117	75	u	<U>
86	56	V	SH <V>	118	76	v	<V>
87	57	W	SH <W>	119	77	w	<W>
88	58	X	SH <X>	120	78	x	<X>
89	59	Y	SH <Y>	121	79	y	<Y>
90	5A	Z	SH <Z>	122	7A	z	<Z>
91	5B	[	CLR <,> (8)	123	7B	{	CLR SH <,>
92	5C	\	CLR </>	124	7C	|	CLR SH </>
93	5D	]	CLR <.>	125	7D	}	CLR SH <.>
94	5E	^	CLR <;>	126	7E	~	CLR SH <;>
95	5F	_	CLR <ENTER>	127	7F	DEL	CLR SH <ENTER>

Extended (non-ASCII)

DEC	HEX	TAG	ENTERED BY	DEC	HEX	TAG	ENTERED BY
128	80	<BREAK>	[V6]	160	A0		CLR <SPACE>
129	81	CLR ^A	<F1> ECM <⇐>	161	A1		CLR SH <1>
130	82	CLR ^B	<F2> ECM <⇑>	162	A2		CLR SH <2>
131	83	CLR ^C	<F3>	163	A3		CLR SH <3>
132	84	CLR ^D	<F4> ECM <⇒>	164	A4		CLR SH <4>
133	85	CLR ^E		165	A5		CLR SH <5>
134	86	CLR ^F		166	A6		CLR SH <6>
135	87	CLR ^G		167	A7		CLR SH <7>
136	88	CLR ^H	ECM <⇓> (10)	168	A8		CLR SH <8>
137	89	CLR ^I	SCM CLR <⇒>	169	A9		CLR SH <9>
138	8A	CLR ^J	SCM CLR <⇓>	170	AA		CLR SH <:>
139	8B	CLR ^K		171	AB		
140	8C	CLR ^L		172	AC		
141	8D	CLR ^M		173	AD		CLR <->
142	8E	CLR ^N		174	AE		
143	8F	CLR ^O		175	AF		
144	90	CLR ^P		176	B0		CLR <0>
145	91	CLR ^Q	<sF1> ECM <s⇐>	177	B1		CLR <1>
146	92	CLR ^R	<sF2> ECM <s⇑>	178	B2		CLR <2>
147	93	CLR ^S	<sF3>	179	B3		CLR <3>
148	94	CLR ^T	<sF4> ECM <s⇒>	180	B4		CLR <4>
149	95	CLR ^U		181	B5		CLR <5>
150	96	CLR ^V		182	B6		CLR <6>
151	97	CLR ^W		183	B7		CLR <7>
152	98	CLR ^X	SCM CLR SH <⇐>	184	B8		CLR <8>
153	99	CLR ^Y	SCM CLR SH <⇒>	185	B9		CLR <9>
154	9A	CLR ^Z		186	BA		CLR <:>
155	9B	SCM CLR SH <⇑>		187	BB		
156	9C			188	BC		
157	9D			189	BD		CLR SH <->
158	9E			190	BE		
159	9F			191	BF		

DEC	HEX	TAG	ENTERED BY	DEC	HEX	TAG	ENTERED BY
192	C0		CLR <@> (11)	224	E0		CLR SH <@> [V6]
193	C1		CLR <A> (12)	225	E1		CLR SH <A>
194	C2		CLR <B> (12)	226	E2		CLR SH <B>
195	C3		CLR <C> (12)	227	E3	(13)	CLR SH <C>
196	C4		CLR <D> (12)	228	E4	(13)	CLR SH <D>
197	C5		CLR <E> (12)	229	E5		CLR SH <E>
198	C6		CLR <F> (12)	230	E6	(13)	CLR SH <F>
199	C7		CLR <G> (12)	231	E7		CLR SH <G>
200	C8		CLR <H> (12)	232	E8		CLR SH <H>
201	C9		CLR <I> (12)	233	E9		CLR SH <I>
202	CA		CLR <J> (12)	234	EA		CLR SH <J>
203	CB		CLR <K> (12)	235	EB	(13)	CLR SH <K>
204	CC		CLR <L> (12)	236	EC		CLR SH <L>
205	CD		CLR <M> (12)	237	ED		CLR SH <M>
206	CE		CLR <N> (12)	238	EE		CLR SH <N>
207	CF		CLR <O> (12)	239	EF		CLR SH <O>
208	D0		CLR <P> (12)	240	F0	(13)	CLR SH <P>
209	D1		CLR <Q> (12)	241	F1	(13)	CLR SH <Q>
210	D2		CLR <R> (12)	242	F2	(13)	CLR SH <R>
211	D3		CLR <S> (12)	243	F3		CLR SH <S>
212	D4		CLR <T> (12)	244	F4	(13)	CLR SH <T>
213	D5		CLR <U> (12)	245	F5		CLR SH <U>
214	D6		CLR <V> (12)	246	F6		CLR SH <V>
215	D7		CLR <W> (12)	247	F7		CLR SH <W>
216	D8		CLR <X> (12)	248	F8		CLR SH <X>
217	D9		CLR <Y> (12)	249	F9		CLR SH <Y>
218	DA		CLR <Z> (12)	250	FA		CLR SH <Z>
219	DB			251	FB		
220	DC			252	FC		
221	DD			253	FD		
222	DE			254	FE		
223	DF			255	FF		

Abbreviations:

^ => Notation for the <CTRL> key.
CLR => Notation for the <CLEAR> key.
CLRSH => Depression, in order, of the <CLR> and <SH> keys.
CTRL => "Control" key operated by <SHIFT><⇓>, or <CTRL> key.
ECM => Extended Cursor Mode toggled by <CLR><SH><SPACE>.
SCM => Standard Cursor Mode toggled by <CLR><SH><SPACE>.
SH, s => Notation for the <SHIFT> key.
SHCLR => Depression, in order, of the <SH> and <CLR> keys.

Notes:

1. Can also be generated with <⇐> if not in ECM.

2. Can also be generated with <⇒> if not in ECM.

3. Can also be generated with <⇓> if not in ECM.

4. Can also be generated with <ENTER>.

5. Can also be generated with <SH><⇐> if not in ECM.

6. Can also be generated with <SH><⇒> if not in ECM.

7. Can also be generated with <SH><⇑> if not in ECM.

8. Can also be generated with ⇑ if not in ECM.

9. Can also be generated with <CTRL><ENTER>.

10. Also generated in SCM with <CLEAR><⇐>.

11. Used to empty the type-ahead buffer.

12. Used by KeyStroke Multiply, if KSM is active.

13. Used by the MiniDos filter if active.

14. For DOS version 6, repeats the last DOS command.

KSM

KSM/FLT allows the use of files containing phrases associated with the unshifted alphabetic keyboard keys to be used as direct keyboard inputs. The syntax is:

SET *KS [TO] KSM/FLT [Using] filespec (Enter=n) FILTER *KI *KS	(V6)
FILTER *KI [TO] KSM/FLT filespec (Enter=nn)	(V5)
filespec	Is an existing KSM type file
Enter=n	Is an optional parameter specifying the character to be used as an embedded enter.

Because the KSM filter uses the <CLEAR> key as a special control key, the DOS 5 KI/DVR program must be set before the KSM filter is applied.

As shown in the syntax block, the SET command is used to install the KSM filter for DOS version 6; the filter is attached to the keyboard device via the FILTER command. The FILTER command is used establish the KSM filter for DOS version 5.

The KSM program will load up to 26 phrases from the specified file (filespec) into memory. These phrases will be taken as though they were typed in from the keyboard when the <CLEAR> key and the specified unshifted alphabetic key are held down together. The default file extension for the filespec is /KSM. To create a KSM file, use the BUILD command in the following manner:

BUILD filename/KSM

This BUILD command will display the alphabetic keys one at a time and allow you to input your desired phrase or command. The extension of the filespec must be specified as /KSM to see the individual key prompts. The actual display will be:

A=>

and will continue up to Z=>. Once all 26 characters have been assigned, the file will be closed and the BUILD will be terminated. The BUILD may also be terminated any time before reaching Z=> by pressing the <BREAK> key in response to any character prompt.

The following rules will govern the entry of phrases during the BUILD.

- Each phrase should be terminated by pressing <ENTER>. This does not place an <ENTER> character at the end of the phrase, but merely signifies the end of the phrase.

- The "ENTER" parameter is used to determine what character KSM will see as an embedded <ENTER> key. If not specified, it defaults to a semicolon ";". The value for enter may be entered as a decimal value between 0 and 255, or as a character enclosed in quotes, such as ":". Whenever this character is encountered in a KSM phrase, it will be translated into an <ENTER>.

- Length of phrases should be limited to 63 or 79 characters for DOS command lines and 255 characters for BASIC lines.

The BUILD (HEX) parameter may be used to create characters or strings that are not directly available from the keyboard. The KI/DVR program does allow the full ASCII character set to be generated. However, if you wish to change key assignments, or to generate characters above X'7F', the **BUILD (HEX)** command will accomplish this.

It is not absolutely necessary to use the BUILD command with the /KSM file extension to create the KSM files. Any file in ASCII format can be used by the KSM/FLT program. If you wish to use the BUILD command without the /KSM extension, a BASIC program, or a word processor to create the KSM file, observe the following format.

When the file is read in by the KSM program, it is stored in memory according to lines. This means that all characters up to the first carriage return (X'0D') will be assigned to the letter <A>, all characters up to the next carriage return to the letter <B>, etc. If a key is to be skipped or left undefined, a carriage return must be inserted for that character. Remember that the character specified with the ENTER parameter (default is the semi-

colon) will be translated into an <ENTER> by the KSM/FLT program.

Following are some examples of the KSM function in the DOS command mode.

```
A=> DIR :0
```

This string would appear when the <CLEAR> and <A> keys were pressed together. The command DIR :0 would be shown but would not be acted upon until the <ENTER> key was pressed.

```
A=> DIR :0;
```

This is the same as the last example, except the "DIR :0" command would be executed immediately as an <ENTER> was the last character of the phrase (represented by the semi-colon).

```
F=> FREE;DEVICE;
```

This phrase would be read in when the <CLEAR> and <F> keys were pressed together. The DOS command "FREE" would be executed, and the command DEVICE would execute afterward.

Following are some examples of the KSM function in BASIC.

```
F=> FOR
N=> NEXT
C=> CLEAR5000:DEFINT A-Z:DEFSTR S,U,V:DEFDBL D:DIM
S(100);
```

The keys <F> and <N> could be assigned the phrases "FOR" and "NEXT". Whenever these BASIC commands were needed, they could be entered in by pressing the <CLEAR> and alphabetic key. The <C> key would insert the entire line associated with it into the program. It is possible to assign the most common BASIC keywords and commands to a KSM file so they may be instantly inserted while programming in BASIC.

Once *KI is filtered with a KSM file, a different KSM may be utilized as follows:

- If the length of the new KSM file is less than or equal to the original KSM file, merely issue another filter command. The new KSM file will be loaded over the existing one, and will not require any additional memory.

- If the new KSM file is larger than the original, you will not be allowed to change it. The message "Request exceeds available memory" will appear, and the FILTER operation will abort. You will have to do a global RESET to remove all configuration, and then reconfigure using the larger KSM file. Be sure to set the KI/DVR program again before applying the new KSM file.

MEMDISK/DCT - RAM Disk Driver

MemDISK is a DOS version 6 driver which allows you to add a pseudo floppy drive to the DOS system, which stores its files in the regular or extended memory of the computer, rather than on a physical floppy disk. The syntax for installing MemDISK is:

```
SYSTEM (DRIVE=d,DRIVER="MEMDISK"[,DISABLE])

DRIVE=d    Specifies the logical drive number.

DISABLE    Required if the "d" slot is active.
```

With this invocation of the SYSTEM command, the following menu is displayed:

```
<A> Bank 0 (Primary Memory)
<B> Bank 1
<C> Bank 2
<D> Banks 1 and 2
<E> Disable MemDISK

Which Type of Allocation
 <A>, <B>, <C>, <D>, or <E> ?
```

Each bank of switchable memory contains 32K of RAM. If your system has only 64K, then you do not have banks 1 and 2, only bank 0. Bank 0 is the top half of user memory. It is shared by programs, drivers, filters, and MemDISK.

Selecting "A" sets up a RAM disk but uses the primary high memory area of your machine. If you have a 64K computer, then this is the only available installation. A bank 0 RAM disk can seriously reduce the amount of memory available to your programs and is not recommended. Because bank 0 is shared, if you select "A", you are prompted for the number of cylinders that are to be used for the MemDISK. You must select at least three cylinders. Selecting the number of cylinders allows you to use Memdisk but still retain some memory for the programs you want to run. The amount of memory used by each cylinder is 4608 bytes

per cylinder (4.5K) for double density (DDEN) and 2560 bytes per cylinder (2.5K) for single density (SDEN).

Selecting "B" uses one memory bank to create a 32K RAM disk in bank 1. Selecting "C" also uses one memory bank to create a 32K RAM disk, but creates it using bank 2. A 2-bank 64K RAM disk is established by selecting "D" at this prompt.

Selection "E" disables the selected drive's MemDISK driver. This allows you to remove a MemDISK from memory and possibly free its driver space for other programs.

After selecting A, B, C, or D, you will be asked:

```
Single or Double Density <D> ?
```

This allows you to cause the selected memdisk to emulate a single density floppy disk, or a double density floppy disk. By simply pressing **<ENTER>**, the system will default to single density. If you want to use the RAM disk as the system drive, it must be installed as double density.

If you selected to utilize bank 0 memory (selection "A"), you will be aked for the number of cylinders of storage space to utilize via the prompt:

```
Note: each cylinder equals xx.xK of space
Number of free cylinders: 3-12 ?
```

If MemDISK finds that the selected memory seems to be pre-formatted (as could be the case if you were re-installing the memdisk after a re-boot), you will be told:

```
Destination MemDISK contains data
```

You will be finally asked:

```
Do you wish to Format it <Y,N> ?
```

Answering "N" or **<ENTER>** will leave the memory alone and allow you to recover the MemDISK data that may have been stored there prior to a re-boot. This provides you an opportunity to re-use a previous MemDISK. Answering "Y" will force MemDISK to reformat the allocated RAM, and set up the BOOT/SYS and DIR/SYS files. Formatting is not optional upon

initial installation. MemDISK is not initially installed unless you format it.

Once the formatting is done, MemDISK will place its directory onto cylinder 1 of the pseudo floppy drive, and display a completion message.

Once you turn your machine off, afterward when you turn on the system and boot up, all information, including the boot and directory information on the pseudo disk will have been erased. Therefore, DOS will not permit you to SYSGEN while a MemDISK is active.

If you want to disable the MemDISK, you must issue the command:

SYSTEM (DRIVE=drive,DRIVER="MEMDISK") <ENTER>

Then, at the menu select the <E> option. Memdisk displays one of the following messages:

If you receive the message, "MemDISK disabled, memory now available", MemDISK was disabled and was able to totally remove itself from the memory it was using.

If you receive the message, "MemDISK disabled, Unable to reclaim high memory". MemDISK was unable to reclaim high memory (Bank 0) because another driver or filter was installed after MemDISK was set up and the other program is still in the way. This is known as memory fragmentation. If you need to use this area of memory, then you must reset the system.

If you receive the message, "MemDISK disabled, Unable to reclaim driver area", MemDISK was disabled and was able to reclaim high memory or alternate bank memory, but it could not reclaim the driver area.

Finally, if you receive the message, "MemDISK disabled, Unable to reclaim high memory and driver area", MemDISK was disabled, but it could not reclaim any memory.

Filters and drivers can be loaded into an area within DOS called I/O driver low memory. This area does not take away from the memory available for your programs. However, not all of the drivers and filters can fit into this area at the same time. If there is no room left in low memory, most of the drivers and filters can be automatically loaded into high memory. Low memory works on a first come, first served basis and MemDISK is the

only driver or filter that must load into low memory. Thus, MemDISK should be installed before other drivers and filters. This ensures that there is space available in low memory for Memdisk to reside.

If you omit the "DRIVE=" parameter in the system command when installing MemDISK, DOS displays "Logical drive number required" error message.

You cannot specify drive :0 when installing MemDISK. You must install MemDISK as another drive, then use the SYSTEM (SYSTEM=0) command to change drive :0 to MemDISK - after first backing up the system files to it. After MemDISK becomes the system drive, you can **PURGE SYS0/SYS** from the MemDISK.

MemDISK must be installed with the SET command. If you attempt to invoke MemDISK as a command, DOS displays the error message: "Must install via SYSTEM (DRIVER=)!"

If you attempt to re-install MemDISK in a different area of memory than the area that it was originally installed in, you will get the error message "MemDISK already Active". MemDISK must always be re-installed as it was initially installed.

If you specify the wrong drive number (in the **SYSTEM (DRIVE=drive,DRIVER="MEMDISK")** command) and you attempt to disable the Memdisk, then you receive the error message: Target Drive not a MemDISK.

If you attempt to disable a MemDISK and there is no MemDISK in the system to disable, then you receive the error message: "MemDISK not present".

MiniDOS [Model I/III only]

The MiniDOS filter program provides a means to access certain DOS functions without having to be at the DOS Ready prompt. The syntax is:

```
FILTER "KI [USING] MINIDOS/FLT
```

Because the MiniDOS filter uses the <CLEAR> key as a special control key, the KI/DVR program must be set before the MiniDOS filter is applied.

The MiniDOS filter allows the keyboard driver to intercept certain keyboard inputs and immediately act on them. Note that the filter will reside in high memory. If *KI is reset, the MiniDOS filter will re-use its initial memory allocation if activated again.

Once the MiniDOS filter is applied, pressing the <CLEAR><SHIFT> and the specified alphabetic key will cause the following:

<C> - Toggle the CLOCK display on or off.
<D> - Enter the system DEBUGger (if activated).
<F> - Display FREE space for all active drives.
<K> - Kill a file.
<P> - Send a character to a line printer.
<Q> - Display a disk's directory.
<R> - Repeat the last DOS command.
<T> - Issue a Top Of Form to the line printer.

When the MiniDOS filter intercepts one of these keys, it will immediately execute the associated function. These keys are active inside any program that uses the DOS keyboard driver, including BASIC. When the function has been completed, control will be returned to the calling program as though no key had been pressed. For this reason, if some of these functions are executed from the DOS level, the DOS Ready prompt will not appear on the screen when the operation is complete. However, the system is still positioned as if the prompt were on the screen, and is ready to take another input.

The full descriptions and parameters for each command will be listed here.

<C> - The <C> command will toggle the clock display on or off. This is identical to issuing a **CLOCK (ON)** or **CLOCK (OFF)** command.

<D> - The <D> command will enter the system DEBUGger or extended DEBUGger, providing it has been previously activated with the **DEBUG** or **DEBUG (EXT)** command.

<F> - The <F> command will allow you see the free space available on a drive, along with the disk's name and date of creation. After selecting this command, you will see the letter "F" enclosed within braces. At this point, enter the drive number of the target drive and press **<ENTER>**. The display will be in the following format:

```
NNNNNNNNDDDDDDDD    FFFF K Free
N = the disk name.
D = the disk date of creation.
F = the free K (1024 bytes) in Hex notation.
```

<K> - The <K> command will allow you to kill a specified file. You will see the letter "K" appear with in braces. At this point, type in the filespec you wish to kill. If no drivespec is included, all drives will be searched and the first matching filespec will be killed.

<P> - The <P> command will allow you to send a character to a line printer. The character must be in the form of two hexadecimal digits. This feature will allow you to send control characters to the line printer to switch printing modes, etc. If an invalid character is entered, an asterisk will appear. You will be allowed to re-enter the character at this point.

<Q> - The <Q> command will show the visible files on a specified drive. You will see the letter "Q" appear within braces. Type in the drivespec of the target drive, and the visible files will be displayed in four-across format. You may also specify a particular file extension. The syntax would be:

```
d/EXT
```

where "d" is the drivespec, and /EXT is the desired extension. The

wildcard character "$" may be used, but all three characters must be specified. For example, to find all file extensions starting with B, /B$$ must be specified.

<R> - The <R> command will repeat the last issued DOS command regardless of the filter's position in the keyboard device chain.

<T> - The <T> command will issue a "Top Of Form" to the line printer. This will also clear the line counter.

Entering an invalid parameter for any of the above commands will display the associated DOS error message.

floppy/DCT

The MODx [Model I/III] and FLOPPY [DOS version 6] DCT programs are used to change the logical drive numbers of 3.5", 5.25", and 8" floppy drives. The SYSTEM command is used to execute the driver program.

SYSTEM (DRIVE=d,(DISABLE,)DRIVER="name")

name	The name of the driver provided with your DOS, FLOPPY, LX80, MODx (x=1 or 3) or MAX80
d	the new logical drive number, 1 to 7.
x	1 or 3, depending on the computer model.

This driver program is provided to allow you to change the logical numbers of your 3.5", 5.25", and 8" (where applicable) floppy drives. It will primarily be used when running a hard drive. Used in conjunction with the **SYSTEM (SYSTEM=)** command, it will allow you to set up your hard and floppy drives to any desired logical number sequence. If you have already defined a drive with the SYSTEM command, you must disable the drive with the **DISABLE** option before you redefine the device.

Upon execution, the following prompt or prompts will be displayed:

```
Enter drive code (0=5", 1=8"):
Enter drive I/O address <1-4>
```

The drive code prompt will be asked if your computer supports 8" drives. The drive I/O address requested will be a number between 1 and 4, and will correspond to the drive's physical location on the drive cable. On the Model I, the first physical drive on the cable will be 1, the second will be 2, etc. On the Model III and 4, the lower built in floppy will be 1, the upper built in floppy will be 2, and the two external drives will be 3 and 4.

The driver program must be installed using the SET command. If you attempt to invoke it as a command from the DOS Ready prompt, DOS displays the error message: Must install via: SYSTEM (DRIVER=)!

FORMS and PR

The FILTER programs FORMS/FLT or PR/FLT are provided to format the data sent to the line printer. The syntax is:

SET *FF FORMS/FLT	[V6]
FILTER *PR *FF	[V6]
FILTER *PR FORMS/FLT (parm,parm,...)	[V5]

AddIf Will add a linefeed after a carriage return.

Chars The number of characters per printed line.

Ffhard Will issue an X'0C' for a form feed, rather than a series of linefeeds.

Indent Number of characters to indent from left margin on lines longer than CHARS parm.

Lines The number of lines printed on each page.

Margin Sets the left margin.

Page Sets the physical page length in lines.

PORT Sends output to a specified port (Model I only).

Tab Causes expansion of X'09' tab characters.

SLine= Adjusts the printer line counter to Model I or Model III conventions.

Xlate=X'aabb' specifies a one-character translation.
aa = the character to be translated.
bb = what "aa" will be translated to.

This filter program adds certain enhancements to the normal printer driver routine. Oncethe filter has been applied, *PR may be returned to its power up driver with the **RESET *PR** command. The forms filter's operating

parameters may be changed after the filter has been installed into memory by using the FORMS command. If you wish to re-install the filter program, it will occupy the same high memory initially allocated.

If you have entered a command that uses the printer, you will not experience "lock up" if the printer is not connected to the system. Realize that if the printer is merely in a deselected or alert state, the system will wait until printer capabilities have been re-established. However, the printer driver portion of this filter also supports a ten-second time-out enabled or disabled with the **SYSTEM (PRTIME)** command which can be used to return control to your program even if an on-line printer goes off-line due to some mechanical difficulty or paper out condition.

This filter program also adds two features to operation under BASIC. The command LPRINT CHR$(6) will reset the system line counter to top of page. This may be used when manually positioning to top of form. Also, the command LPRINT with no arguments will now cause a blank line to be generated.

The filter will allow you to determine the format of the data sent to your line printer. There are several configurable parameters used to set the format of the DOS version 5 PR/FLT output. They are:

ADDLF If this parameter is specified, a linefeed will be issued after every carriage return.

CHARS= This parameter sets the number of characters that will be printed on each line. It may be any integer between 1 and 255.

FFHARD If this parameter is specified, any form feed determined by the PAGE and LINES parameters will be sent as an X'0C' character rather than a series of linefeeds. If you use this parameter, be sure your printer will recognize the X'0C' character.

INDENT= This parameter sets the number of spaces a line is to be indented if the line length exceeds (CHARS=) characters. The default value for this parameter is zero (0).

LINES= This parameter sets the number of lines that will be

printed on each page. It may not exceed the PAGE parameter, and if not specified, it will default to the PAGE parameter of 66.

MARGIN= This parameter sets the width of the left margin. It is especially useful for printers with fixed position tractors.

PAGE= This parameter sets the physical page size in lines. It should be set to the particular form size you are printing on (66 for normal printer paper, 6 for mailing labels, etc.). The default value is 66 lines per page.

PORT= On the Model I, this parameter changes printer output from the normal memory mapped location to a user specified port. Any port between 1 and 255 may be specified.

SLINE=n The allowable values are 0 or 1. A zero will set the page length to 66 lines per page and the initial line count to 0, matching Model I conventions. A one will set the lines per page to 67 and the initial line count to 1, matching Model III conventions. If this parameter is not specified, the normal convention will be used (Model I = 66,0 and Model III = 67,1).

TAB If this parameter is specified, any X'09' character will be expanded to a standard 8 column tab.

XLATE This parameter will translate a specified character to another character. The format is X'aabb', where aa is the character to be translated, and bb is desired character result. Both aa and bb must be hexadecimal values. This parameter may be useful to translate printer control characters when using more than one type of printer on the same system.

SET *FF *PR USING FORMS/FLT [V6]
FILTER *PR *FF

FILTER *PR USING PR/FLT [V5]

These set of commands will establish the filter program in high memory (or low memory for DOS version 6 if sufficient space is available in the I/O driver region) and initiate it to filter the *PR (line printer) output. The first set illustrated is for DOS version 6 while the latter is for DOS version 5. Examples for changing the filtering parameters are illustrated in the section on the FORMS.

```
FILTER *PR PR (SLINE=0)
FILTER *PR PR (SL=0)
```

This DOS version 5 example will establish the printer line counter to start from 0 rather than 1, and use a page length of 66 lines per page as the SLINE=0 parameter was specified. The SLINE parameter will match Model I conventions, and may be necessary when running Model I software on the Model III.

Job Control Language

The DOS Job Control Language (JCL) is one of the most powerful features of the DOS operating system. It allows the user to construct a sequence of commands and statements to control the actions of the operating system or applications programs. There are many different features to JCL, providing for user prompts and alerts, allowing the input of specified variables at runtime, providing for logical branching of program control based on user inputs, and allowing for variable substitution.

How JCL works

To use JCL, it is first necessary to understand how it works. In the most basic sense, this is the procedure:

- The user creates an ASCII text file consisting of commands and statements he wishes to invoke.
- The user starts the JCL processing with the DO command.
- The JCL processor takes over control of the keyboard.
- A line is read in from the file and passed to the system **exactly as if it came from the keyboard.**
- When the end of the file is reached, keyboard control returns to the user and the JCL processing stops.

From this description, the purpose of JCL could be summarized as a method to execute a series of commands to control the computer with no input from the computer operator other than to start up the procedure.

While the above description is correct, it is by no means a complete description of JCL's capabilities. The following sections of the JCL documentation will describe how to use many different features. The layout of the sections will start with the basics of creating a JCL file, and then show how to incorporate the more advanced features. It is recommended that you read these sections in order, as later sections will refer to material presented in the earlier ones.

Creating a JCL file

As noted in the above description, a JCL file is an ASCII file. For the purposes of JCL, this means a file containing those characters normally available from the keyboard. There are many different ways to create a JCL file. The BUILD command will let you create or extend a JCL file, but does not provide a means to edit an existing file. You can use the TED text editor for that purpose. Any word processor or text editor can also be used to create or edit a JCL file, as long as it can save a file in ASCII format without line numbers. You could also create a JCL file with a BASIC program, creating the lines as strings and writing them to a sequential file.

> Note: No single line in a JCL file may be more than maximum command line length less one [63 characters for DOS version 5, 79 for DOS version 6]. Depending on the JCL method used (Execute only or Compiled), JCL will either ignore all characters after the limit, or abort the processing entirely.

Restrictions of JCL

Certain DOS commands cannot be executed from a JCL file. As the main concept of JCL is to use a pre-determined set of commands, any program with unpredictable prompts will not function properly when run from a JCL file. Also, any program which requires removing the system disk will certainly cause the JCL to abort. Among the commands **not** valid from a JCL file are certain BACKUP commands, BUILD, COPY (X), certain CONV commands, DEBUG, certain PURGE commands, RESET, and RESET *KI. As a general rule, you should not use any library command or utility program when specifying a QUERY parameter (although the global RESET and RESET *KI commands cannot be used in a JCL file, a RESET *device can be used with any device other than *KI).

However, if the order of prompts or inputs in a program is known, it is allowable to pre-arrange the proper responses in a JCL file, being careful that they remain in sync with the prompts. In this manner, you can have a JCL file totally run a program or other procedure with no operator input. This will depend on the method used by the program to normally take keyboard input.

Simple JCL execution

JCL files that contain only executable comments, commands, or execution JCL macros are very common in day to day use of the DOS system. The easiest JCL file to understand is one containing only commands. For example, let's assume that you have a program that requires the use of the printer filter program PR/FLT [Model I/III] or FORMS/FLT [DOS 6] to set the printer line length, margin and page length. You could put the following command lines in a JCL file:

```
SET *FF FORMS                                    [DOS 6]
FILTER *PR PR/FLT                                [DOS 5]
FORMS (CHARS=80,MARGIN=10,LINES=60)
```

If this JCL file were called START/JCL, using the command **DO = START** at the DOS Ready prompt would execute the lines and apply the printer filter, returning to the DOS Ready prompt.

Let us further assume that you now wish to go into BASIC and run a program. The JCL file could be expanded as follows:

```
SET *FF FORMS                                    [DOS 6]
FILTER *PR PR/FLT                                [DOS 5]
FORMS (CHARS=80,MARGIN=10,LINES=60)
BASIC
RUN"PROGRAM/BAS"
```

Now using the command **DO = START** will establish the printer filter, enter BASIC, and pass the command **RUN"PROGRAM/BAS"** to BASIC. The program would be loaded in from disk and executed. However, like the first example, you will return to the DOS Ready prompt as soon as the first keyboard input is requested by the program! To solve this problem, we must add one of the special JCL execution commands, called a "MACRO", to the end of the file.

JCL Execution Macros and Comments

JCL execution macros perform many different functions. They are always entered in the JCL file as two slashes followed by the name of the macro. An execution comment is any line that starts with a period. These

comments will be displayed to the screen during execution. Following is a list of all JCL execution macros:

JCL EXECUTION COMMENT	. COMMENT
JCL TERMINATION MACROS	//ABORT, //EXIT, //STOP
JCL PAUSE/DELAY MACROS	//DELAY, //PAUSE, //WAIT
JCL ALERT MACROS	//ALERT, //FLASH
JCL KEYBOARD MACROS	//KEYIN, //INPUT

An execution macro cannot be the first line in a JCL file!

The execution comments provide a means to display informative messages as the JCL file executes. You could label your JCL file and show other useful information as follows:

```
. Program start up JCL, last modified 01/01/92
SET *FF FORMS                              (DOS 6)
FILTER *PR PR/FLT                          (DOS 5)
FORMS (CHARS=80,MARGIN=10,LINES=60)
BASIC
RUN"PROGRAM/BAS"
```

This comment would be displayed when the JCL executes, and show the file's purpose, and the last date you made modifications to the file.

Remember from our last example that an unwanted return to the DOS Ready prompt would be made as soon as a keyboard input was requested by the program. To keep this from occurring, you can use the //STOP macro.

JCL "TERMINATION" macros

//STOP

The //STOP macro is used to halt execution of the JCL file and return keyboard control to an application requesting keyboard input. Thus, our JCL example could be expanded as follows:

```
. Program start up JCL, last modified 01/01/92
SET *FF FORMS                              (DOS 6)
FILTER *PR PR/FLT                          (DOS 5)
FORMS (CHARS=80,MARGIN=10,LINES=60)
BASIC
RUN"PROGRAM/BAS"
//STOP
```

The JCL file is now complete, and as soon as the program requested a keyboard input, the keyboard would become "alive". The response to the prompt could then be input from the keyboard.

However, perhaps the program is one that requires no keyboard input during its execution. In this case, you might want to return to the DOS Ready prompt when the program is completed. Using the //STOP macro in this case would not be correct. When the program completed, the //STOP would be executed, and the BASIC Ready prompt would appear. You would not return to the DOS Ready level.

As noted before, a return to DOS Ready will happen automatically if you do not use the //STOP macro at the end of the JCL file. Another way to force an end to the JCL execution and return to DOS Ready is to use either the //ABORT or //EXIT macros to end the JCL file.

//ABORT

The //ABORT macro is used to exit a JCL procedure and return to the program that initiated the DO command. It is quite similar to the //EXIT macro. A return to calling program will take effect after displaying the message:

```
Job aborted
```

It would be used if your JCL processing logic detected an invalid run-time condition, and wanted to display an informative message. Also, any error that the operating system detects that will result in a jump to the @ABORT DOS vector will disable further JCL processing and display the above message. Basically, this macro should be used to exit JCL execution any time an undesired condition occurred.

//EXIT

The //EXIT macro is used to end the execution of JCL processing and return to the program that initiated the DO command. If no termination macro is entered in a JCL file, the JCL processing will terminate upon reaching the end of the file as though //EXIT was the last line in the JCL file, displaying the message:

```
Job done
```

The Job done message indicates a normal conclusion of the JCL file. This type of JCL exit should be used if the conclusion of the JCL command file also represents the conclusion of the job that is running. Therefore, the following JCL could be used to run a program that did not require any keyboard input, and needed to return to the DOS Ready prompt after it finished.

```
. Program start up JCL, last modified 01/01/92
SET *FF FORMS                                    [DOS 6]
FILTER *PR PR/FLT                                [DOS 5]
FORMS (CHARS=80,MARGIN=10,LINES=60)
BASIC
RUN"PROGRAM/BAS"
//EXIT
```

As you can see from these macros, you have three different ways to end a JCL file.

//STOP	- Stop JCL execution, remain in the user's program.
//ABORT	- Stop execution, display "Job Aborted".
//EXIT	- Stop execution, display "Job Done".

Be sure to use the proper termination macro for the intended job application.

JCL "PAUSE/DELAY" Macros

The other execution macros can be used to provide special effects if you need them in your JCL files. One of the most often used is the //PAUSE macro, which provides a means to temporarily suspend JCL execution.

```
//PAUSE optional message string
```

When this macro is encountered in an executing JCL file, it will be displayed on the screen along with any optional message. The message can inform you why the pause was ordered. Pressing <ENTER> will resume JCL execution, while pressing <BREAK> will abort the JCL. For example:

```
. Program start up JCL, last modified 01/01/92
SET *FF FORMS                                    [DOS 6]
FILTER *PR PR/FLT                                [DOS 5]
FORMS (CHARS=80,MARGIN=10,LINES=60)
BASIC
//PAUSE Be sure data disks are mounted in drives 1 and 2!!
RUN"PROGRAM/BAS"
//EXIT
```

This example will suspend the JCL as soon as BASIC executes and before the program is run and loaded. You can then check that the needed disks are in your other drives, and press <ENTER> to continue the JCL.

The //DELAY and //WAIT macros are similar to the //PAUSE macro, and used to give JCL execution a specific delay period.

```
//DELAY duration
```

The //DELAY macro will provide for a definite timed pause. JCL execution will automatically continue at the expiration of the delay period. The actual delay will be approximately 0.1 seconds per count. The count may range from 1 to 256. Thus, a delay from 0.1 seconds to a delay of

25.6 seconds is possible. The following example shows the proper syntax of the //DELAY macro.

```
. This could be an informative message for the operator
//DELAY 50
basic
run"newprog/bas"
//STOP
```

This example JCL will print an informative message and then delay for approximately five seconds. After the delay, it will execute BASIC and then run the desired program.

```
//WAIT hh:mm:ss
```

The //WAIT macro is similar to //DELAY, except that the length of the delay depends on the setting of the system clock. Providing the system clock is functioning, the //WAIT macro will put the entire system in a "sleep" state until such time as the system clock matches the time specified in the macro operand. The system clock can be set with the TIME command. You can also set the time from a JCL file by using a direct execution of the TIME command, or with the //INPUT macro, which will be discussed later. Consider the following example:

```
. example JCL for running alarm program
//wait 02:15:00
basic
run"alarmset/bas"
//exit
```

Assuming that the system clock was set, this example would display the comment and then wait until the clock matched the time of 02:15:00 specified in the //WAIT macro. It would then execute BASIC, and run the program ALARMSET/BAS, exiting to DOS Ready after completion of the program.

JCL "ALERT" Macros

The //FLASH and //ALERT macros are provided to give both visual and audio alerts to the operator. The //FLASH macro will blink a message line

on the video screen, making it easy to emphasize an important piece of information. The //ALERT macro will send an audio tone to the Model I/III cassette port or Model 4 sound port, allowing an audio alert.

```
//FLASH duration message
```

This macro will flash a message on and off on the video screen. The duration can be any number from 0 to 255. This is the number of times the message will flash. If no duration is specified, the message will flash 256 times. The message string can be any comment you wish displayed. For example:

```
. TEST/JCL
//FLASH 10 Starting initialization JCL
```

When the TEST/JCL executes, the //FLASH line will be displayed. It will flash on and off 10 times, as specified by the duration count. At any time during this period, you may press <**ENTER**> to stop the flash and proceed to the next line. Pressing <**BREAK**> at this point will abort the JCL and display the message "Job Aborted".

The //ALERT macro may be used to provide an audible signal to the operator. It will generate up to eight different tones and direct its output to the cassette port on the line that normally goes to the AUX cassette jack on the Model I/III. By plugging the AUX lead of the cassette cable into a small amplifier, this macro could prove quite useful. On the Model 4, //ALERT uses the computer's sound port. You could use it to signify the end of a large JCL procedure. It could also be used during the execution of a procedure to bring attention to a specific process. The proper syntax is:

```
//ALERT (tone,silence,...)
```

The actual tone selected is controlled by a tone number. The number range is 0-7, with "7" producing the lowest tone, and "0" producing the highest tone. Any value entered will be used in its modulo 8 form. That is, if you enter the number "8", a zero value will be assumed. The value 65 will produce the tone assigned to a "1". The tone is followed by a period of silence by entering a second number. "Tone" and "silence" must be entered as number pairs (e.g. "1,0"). In fact, this can be repeated for as many number combinations as can fit on one line.

The tone-silence sequence can be made to repeat by enclosing the entire string in parentheses. If parentheses are used, the sequence will keep repeating until the <ENTER> key is pressed, at which time execution will continue. Pressing <BREAK> would abort the JCL. No display is made during the tone generation. Therefore, if your JCL has a repeating tone and you do not have an amplifier connected to the cassette port, the system may appear to hang.

The following example shows several uses of //ALERT:

```
. example of tone generation
//alert (1,0,7,0)
. another example
//alert 0,0,1,0,2,0,3,0,4,0,5,0,6,0,7,0
. still another
//alert (0,0,1,1,2,2,3,3,4,4,5,5,6,6,7,7)
//exit
```

JCL "KEYBOARD ENTRY" Macros

The //KEYIN and //INPUT macros provide a means to take keyboard inputs during JCL execution. Two other macros are used along with //KEYIN to establish execution-time conditional blocks.

//KEYIN optional comment string

This macro is used to prompt for a single character entry, with the entire //KEYIN line being displayed, including any comment message. The resultant entry must be a single numeric character in the range 0-9 and will be used to select one of up to ten different execution phase blocks of JCL. //KEYIN can not be used to enter data at execution time but can only provide for the selection of a predefined block of JCL lines. If it is necessary to provide run-time keyboard interfacing, then the //INPUT macro should be used instead of //KEYIN.

```
. MENU/JCL
. Program 1 is MAIL
. Program 2 is LEDGER
. Program 3 is PAYABLES
//KEYIN Select program, 1 - 3
```

This example shows how you could build a menu using execution comments to display different program choices. By pressing a single key, you could execute the desired program. Refer to the following expansion of the MENU/JCL example:

```
//KEYIN  Select program 1 - 3
//1
BASIC
RUN"MAIL/BAS"
//STOP
//2
BASIC
RUN"LEDGER/BAS"
//STOP
//3
BASIC
RUN"PAYABLES/BAS"
//STOP
///
```

There are two new macros used in this example. They are:

```
//NUMBER
///
```

The //NUMBER is used to start a block of lines that correspond to a value selected with the //KEYIN macro. This block will extend until the next //NUMBER or to the "///".

The triple slash "///" is used to mark the end of all //NUMBER blocks. Regardless if a //NUMBER has been found to match the //KEYIN entry, the JCL processor will stop looking for a match as soon as it encounters a ///. Execution will begin with the following line.

In the above example, pressing "1", "2", or "3" would select the corresponding block of lines, entering BASIC and running the appropriate program. If a key other than "1", "2", or "3" were pressed, all three //NUMBER blocks would be ignored, and execution would begin with the line after the "///". That line is totally dependent on what options you want to allow. If it is mandatory that one of the three programs be run, then an

//ABORT macro could be used to abort the JCL. If other options were available, they could be placed here in the JCL file.

One of these options may be to let the user type in his own command. If this is the case, the //INPUT macro could be used.

```
//INPUT optional message string
```

When using the //INPUT macro in a JCL file, it is recommended that the keyboard type ahead feature of the KI/DVR program either not be active or be disabled with a **SYSTEM (TYPE=OFF)** command. This can be done as a JCL line, if desired. If not done, using any macro such as //PAUSE that requires pressing the **<ENTER>** key will cause the JCL to abort if it later encounters a //INPUT.

The //INPUT macro is used to input a line from the keyboard. Our definition of JCL explained that JCL execution worked by taking over the keyboard, and substituting lines from a JCL file in place of keyboard entry. With this macro, control of the keyboard is temporarily returned to the operator. Now, any command may be typed on the keyboard and then passed to the system. The number of characters allowed will depend on where the JCL execution was when the //INPUT was encountered. For instance, if the JCL was executing at the DOS Ready level, then up to 63 [for DOS 5] or 79 [for DOS 6] characters could be entered, the same as for a normal DOS command. If the //INPUT was encountered after going into BASIC, then up to 255 characters could be entered.

Consider a slight re-write of the MENU/JCL example used with the //KEYIN macro:

```
. MENU/JCL
. Program 1 is MAIL
. Program 2 is LEDGER
. Program 3 is PAYABLES
BASIC
//KEYIN Select program 1-3
//1
//RUN"MAIL/BAS"
//STOP
//2
```

```
RUN"LEDGER/BAS"
//STOP
//3
RUN"PAYABLES/BAS"
//STOP
///
//INPUT your own choice, as RUN"PROGRAM"
//STOP
```

As you can see, this examples is slightly different than the //KEYIN example. First of all, BASIC is entered before the //KEYIN command is displayed. Therefore, the command to enter BASIC has been removed from the three conditional blocks. Now, if a key other than 1, 2, or 3 is pressed for the //KEYIN, the //INPUT line will be displayed. The user can then enter in a **RUN"PROGRAM"** command to start up his own program choice. In fact, the response to the //INPUT does not even have to be a **RUN"PROGRAM"** command. Any valid BASIC statement could be used. As soon as BASIC acted on the line, the //STOP would halt JCL execution and keyboard control would return to BASIC.

This type of direct input to the system is equally valid at the DOS Ready level. When describing the //WAIT macro, it was mentioned that the time for the system clock could be set by the operator in the middle of a JCL file. The next example shows how this is done:

```
. Example JCL for alarm program
//INPUT Enter the time command, use the format TIME
HH:MM:SS
//WAIT 02:15:00
BASIC
RUN"ALARMSET/BAS"
//EXIT
```

This example JCL will prompt the operator and allow the entry of a TIME command to set the system clock. The //INPUT message also describes the proper format of the TIME command. After the input, the //WAIT would pause the system until the clock matched 02:15:00, and then continue execution.

When using the //INPUT macro, some caution should be exercised to assure that the command typed in is valid at the level it will be executed.

For example, mistakenly pressing only <ENTER> with no other characters for an //INPUT at the DOS Ready level will abort JCL execution. The JCL would also abort if, for instance, a program name was entered incorrectly, resulting in an DOS "Program not found" error.

Simple JCL compiling

The previous section of JCL has shown how to create and use execute only JCL files. While this type of JCL file is useful, it does not allow for logical decisions, substitution capabilities, or combination of JCL files. To do that, you must use the features provided by the compile phase of JCL. As the title of this section implies, the basics of the compile phase will be discussed.

There are certain features of the JCL compile phase that will not be discussed in this section of the documentation. The purpose of this section is to describe the basic functions of the JCL compiler, and to show some practical examples of JCL files. The *ADVANCED JCL COMPILING* will contain further examples, including those features not discussed here.

Although JCL is a compiled language, you do **not** have to be a programmer to understand it! In fact, the main purpose of JCL is to let you create files to automatically control applications programs and to maintain the data generated by these programs.

Compilation description and terms

The purpose of the compilation phase is to read in the JCL file line by line, checking for directly executable lines, keyboard responses, and execution macros, and to evaluate any compilation statements, and to write the resultant lines to a file called SYSTEM/JCL. After the compilation is complete, control would normally then be passed to the second phase - the execution of the compiled SYSTEM/JCL file. The DO command allows for four different methods to DO a JCL file. Briefly recapped, they are:

DO = filespec	Execute only
DO *	Execute current SYSTEM/JCL file
DO $ filespec	Compile only to SYSTEM/JCL
DO filespec	Compile to SYSTEM/JCL, then execute SYSTEM/JCL

As stated earlier, the JCL works by substituting lines in a file for keyboard entries. However, when using the compile phase, a JCL file is not restricted to using a series of executable commands to create these substitution lines. All that is required is that the SYSTEM/JCL file contain only executable lines after the compile phase is completed. The user is allowed to create a file consisting of:

- Directly executable commands
- Pre-arranged keyboard responses
- JCL execution macros
- JCL conditional macros
- JCL labels

It is allowable to compile any JCL file, even if it contains only executable lines. Any of the examples in the previous execution JCL section could be compiled. The compile phase would examine each line, determine that it is an execution line, and write it to the SYSTEM/JCL file. After the compilation is completed, the SYSTEM/JCL file would be executed, producing exactly the same results as if the file were executed without compiling.

There are several new terms that will be used when discussing the JCL compilation phase. Briefly described, they are:

TOKEN

The term "token" is the most important term to understand when using the compile phase of JCL. It is, fortunately, very easy to understand. A token is merely a string of up to eight characters, and may contain upper and lower case alphabetic characters A-Z and a-z, and the numbers 0-9.

Tokens have two uses - as a "true" or "false" switch for logical decisions, and as a character string value for use in substitution fields.

LOGICAL OPERATOR

- Logical AND (represented by the ampersand symbol "&")
- Logical OR (represented by the plus symbol "+")
- Logical NOT (represented by the minus symbol "-")

Although mentioned here, these logical operators will be discussed only in the *ADVANCED JCL COMPILING* section.

LABEL

A JCL label is the AT sign "@" followed by up to eight alphanumeric characters. It is used to define the start of a JCL procedure, allowing many small JCL procedures to be combined into one large file.

Like the execution phase, there are several special JCL statements, or MACROS, available with the compile phase. As with the execution macros, they are in the form of two slashes "//" followed by the appropriate word. In order of explanation, they are:

//IF	//ASSIGN	//SET
//ELSE	//. COMMENT	//RESET
//END	//QUIT	//INCLUDE

JCL conditional decisions

Certain JCL macros can be used to establish "blocks" within a JCL file. During the compilation, these blocks will be evaluated for a logical true or false condition. The following shows the basic methods of evaluation:

- If the evaluation is true... Include all the lines until the block end.
- If the evaluation is false... Ignore all the lines until the block end.

An alternate to a false evaluation is also provided.

- If the evaluation is true...
  Include these lines...
  Or else...
  Include these lines.

There are three compilation macros provided to define a block of lines:

//IF	(defines the start of a block)
//END	(defines the end of a block)
//ELSE	(defines the alternative to a false //IF)

Translating the above three examples for use with these three JCL macros would produce the following:

- If the evaluation is true.

```
//IF
include these lines
//END
```

- If the evaluation is false.

```
//IF
ignore these lines
//END
```

- An alternative to a false statement.

```
//IF
include these lines
//ELSE
include these lines
//END
```

As might be apparent, the //IF macro by itself does not determine the truth or falseness of the block. There must be something that the //IF can test to determine a true or false condition. Since our definition of a token described one of its uses as a true or false switch, that something is a token.

A real example of a conditional block would be:

```
//IF drive
. Display this execution comment
//END
```

The token in this example is "drive". The //IF will test whether or not "drive" is true or false. Assume that this block of lines is contained in a file called TEST/JCL. Refer to the three following DO command examples:

1) **DO TEST (drive)**
2) **DO TEST (DRIVE)**
3) **DO TEST**

Examples 1 and 2 would both set the token "drive" to be true. These two examples again emphasize an important point - there is no difference between upper and lower case for any JCL macro, token, or label. Example 3 would set the token "drive" to be false. From these examples, you can see how easy it is do set a token true or false:

- To set a token true, simply specify it on the DO command line.
- To set a token false, do NOT specify it on the DO command line.

According to these rules, using either DO command line 1 or 2 would cause the execution comment in the TEST/JCL example to be written to the SYSTEM/JCL file. Using DO command line 3 would bypass any lines between the //IF and the //END.

This type of logical decision capability allows a single JCL file to be created, and lets the computer operator pick a course of action by merely typing in the same "DO filespec" command with different tokens. For example, consider the following JCL example, which shall be referred to as START/JCL (the first line is an execution comment, as previously explained in the *SIMPLE JCL EXECUTION* section).

```
. START/JCL for program start-up
SET *FF FORMS/FLT                (DOS 6)
FILTER *PR *FF                   (DOS 6)
```

```
FILTER *PR PR/FLT                    (DOS 5)
//IF PR1
FORMS (CHARS=80)
//END
//IF PR2
FORMS (CHARS=132)
//END
```

Let us assume that these are the first lines in a JCL file that will start some applications program running. The JCL installs the forms filter, while the two conditional blocks let the operator define the number of characters per line for printed output via the FORMS command. The DO commands to accomplish this would be:

```
DO START (PR1)
DO START (PR2)
```

When the first DO command is issued, the //IF PR1 will test true, and the 80 character FILTER command will be written to the SYSTEM/JCL file. Because PR2 was not specified, the //IF PR2 will be false, and the second FILTER command will not be written to the SYSTEM/JCL file. Using the second DO command example would reverse the results. In the case where either one or the other FILTER command is always desired, there is an easier way to accomplish the same results, and requires only the use of the PR1 token as shown in the following example:

```
. START/JCL for program start-up
SET *FF FORMS                        [DOS 6]
FILTER *PR *FF                       [DOS 6]
FILTER *PR PR/FLT                    [DOS 5]
//IF PR1
FORMS (CHARS=80)
//ELSE
FORMS (CHARS=132)
//END
```

By using the //ELSE macro, an alternative course of action is provided in case the //IF test is false. Thus the command "**DO START (PR1)**" would use the 80 character FILTER line, and ignore everything between the

//ELSE and the //END. The command "DO START" would cause the //IF PR1 to test false, and therefore use the 132 character FILTER line.

Although the previous examples have shown a single line in each conditional block, any amount of lines may be included. Refer to the following MENU/JCL example:

```
. MENU/JCL selection start-up
//if KI
SET *KI KI/DVR (TYPE,JKL)                    [DOS 5]
FILTER *KI MINIDOS/FLT                       [DOS 5]
FILTER *PR PR/FLT                            [DOS 5]
SET *FF FORMS                                [DOS 6]
FILTER *PR *FF                               [DOS 6]
FORMS (FFHARD,CHARS=80)
//end
BASIC
//if P1
RUN"PROGRAM1/BAS"
//end
//if P2
RUN"PROGRAM2/BAS"
//end
//stop
```

This example references three tokens - KI, P1, and P2. The first conditional block is easy to understand. If the KI token was entered on the DO command line, KI/DVR, MINIDOS/FLT, and forms filter would all be applied. If not, all lines up to the first **//end** would be ignored. Regardless, the command line "BASIC" would be written to the SYSTEM/JCL file for execution. As the compilation continued, the **//if P1**, and then the **//if P2** would be tested. Again, if either token was specified, the **RUN"PROGRAM"** line would be written out. In any case, the last line written out would be the **//stop** execution macro.

Instead of the two separate **//if** macros to determine which, if any, BASIC program would be run, we could have used an **//else** macro in the following manner:

```
//if P1
RUN"PROGRAM1/BAS"
//else
RUN"PROGRAM2/BAS"
//end
```

Although this lets the operator select either program with a single token, it also means that one program or the other will always be selected.

To test a JCL procedure, the compile only "DO $ filespec" command can be used. Once the compiling is complete, the results can be examined by using the LIST command to list the SYSTEM/JCL file to the video or printer. Refer to the following examples, using the previous MENU/JCL example.

```
DO $ MENU (KI,P2)
```

The resultant SYSTEM/JCL file would be:

```
. MENU/JCL selection start-up
SET *KI KI/DVR (TYPE,JKL)                [DOS 5]
FILTER *KI MINIDOS/FLT                   [DOS 5]
FILTER *PR PR/FLT                        [DOS 5]
SET *FF FORMS                            [DOS 6]
FILTER *PR *FF                           [DOS 6]
FORMS (FFHARD,CHARS=80)
BASIC
RUN"PROGRAM2/BAS"
//STOP
```

```
DO $ MENU
```

The resultant SYSTEM/JCL file would be:

```
. MENU/JCL selection start-up
BASIC
//STOP
```

From the above examples, you should now have a basic understanding of how the //IF macro can be used to create a single JCL file that can be used for different purposes. All that is required is that the proper tokens to be

entered on the DO command line. To reduce the number of tokens needed, and to provide for higher conditional logic statements to be handled, JCL provides the //SET, //RESET, and //ASSIGN tokens.

Using //SET, //RESET, //ASSIGN

The //SET macro is used to give a token a logical true value, the same as specifying it on the DO command line. The //RESET token does the opposite, giving a token a false value. One basic use for //SET is to let one token set the value of another. For example:

```
//IF KI
//SET P1
//END
```

If these lines were added to the beginning of the MENU/JCL example file, you can see that specifying only the KI token will also set P1 to a true condition. Again referring to the MENU/JCL file, it is possible that the operator could enter both the P1 and P2 tokens, generally producing undesired results. To keep this from happening, the following lines could be added to the beginning of the file:

```
//IF KI
//SET P1
//RESET P2
//END
```

This conditional block would assure that P2 was reset if P1 was entered on the command line, **even if P2 were also entered!** Now let's rewrite MENU/JCL slightly, assuming that PROGRAM1 requires the keyboard driver and 80 character print lines, and that PROGRAM2 requires no keyboard driver and 132 column print lines. We will also assume that if P1 is not entered, P2 should be the default.

```
. MENU/JCL, revision 1
SET *FF FORMS                          [DOS 6]
FILTER *PR *FF                         [DOS 6]
FILTER *PR PR/FLT                      [DOS 5]
//IF P1
//RESET P2
```

```
SET *KI KI/DVR (TYPE,JKL)                    [DOS 5]
FORMS (CHARS=80)
//ELSE
//SET P2
FORMS (CHARS=132)
//END
BASIC
//IF P1
RUN"PROGRAM1/BAS"
//END
//IF P2
RUN"PROGRAM2/BAS"
//END
//STOP
```

Evaluating the results of different DO command lines would show the following:

DO $ MENU (P1) or DO $ MENU (P1,P2)

```
. MENU/JCL, revision 1
SET *FF FORMS                          [DOS 6]
FILTER *PR *FF                         [DOS 6]
FILTER *PR PR/FLT                      [DOS 5]
SET *KI KI/DVR (TYPE,JKL)              [DOS 5]
FORMS (CHARS=80)
BASIC
RUN"PROGRAM1/BAS
//STOP
```

DO $ MENU or DO $ MENU (P2)

```
. MENU/JCL, revision 1
SET *FF FORMS                          [DOS 6]
FILTER *PR *FF                         [DOS 6]
FILTER *PR PR/FLT                      [DOS 5]
FORMS (CHARS=132)
BASIC
RUN"PROGRAM2/BAS"
//STOP
```

The first //IF macro tests if P1 is true. If so, P2 is reset false, and the KI/DVR and 80 character print mode are applied. If P1 is was not entered on the DO command line, the //ELSE sets P2 to true, and applies the 132 character print mode, even if P2 was not entered. The compiling continues, writing the BASIC line, the selected PROGRAM line, and the //STOP to the SYSTEM/JCL file.

As previously mentioned, the //SET macro can be used to reduce the number of tokens that have to be entered on the DO command line. Consider the following SYSOPT/JCL example:

```
. Establish DOS system options
//IF ALL
//SET KITY
//SET PR
//SET MINI
//SET SRES
//END
//IF KIALL
//SET KITY
//SET MINI
//END
//IF KITY
set *ki ki/dvr (type)                    [DOS 5]
//END
//IF PR
filter *pr pr/flt (chars=80)             [DOS 5]
//END
//IF MINI
filter *ki minidos                       [DOS 5]
//END
//IF SRES
system (sysres=2)
system (sysres=3)
system (sysres=8)                        [DOS 5]
system (sysres=10)
//END
```

This example shows how many different DOS options can be established with a JCL file. The way it is structured, the operator can choose any or all

of the options. Without the use of //SET, it would be necessary to enter four separate tokens to establish all of the options. By using a conditional block, the single token ALL can be made to set all of the necessary tokens true. Also, the token KIALL can be used to set only the two keyboard related options. Notice the use of upper and lower case. As stated previously, the case of a line has no effect on any JCL macro, token or label. This is also true when the line is an DOS command, as are the lower case lines in this example. You may find that for editing purposes, the readability of a JCL file can be improved by using upper case for macros and lower case for executable lines, or vice versa. In the case of large JCL files, this generally makes itself readily apparent.

//ASSIGN

The //ASSIGN macro has two purposes. Like the //SET macro, it will set a token's logical value true. It can also assign a character string value to a token. The syntax for the //ASSIGN macro is:

```
//ASSIGN TOKEN=CHARACTER STRING
```

The character string value assigned to the token will be useful as described in the next section of the JCL documentation, *SUBSTITUTION FIELDS*. The character string can consist of up to 32 upper or lower case alphabetic characters, the numbers 0-9, and the special characters slash "/", period ".", and colon ":". Parameter strings of any ASCII character are acceptable by enclosing them within double quotes. The important point to remember is that the //ASSIGN does set the token's logical value to true, the same as the //SET macro. Note that any time //ASSIGN is used, there must be at least one character assigned as a value. The statement **"//ASSIGN ALL"** would be an invalid statement, and would abort the compiling. You must have an equal sign "=" and some character string value following it when using the //ASSIGN macro, such as **"//ASSIGN ALL=EVERYONE"**.

In any of the previous examples that used the //SET macro, the //ASSIGN macro could have been substituted. The character string value assigned to the token would have no effect on the JCL logic. The important fact would be that //ASSIGN also sets the token true, as shown in the next example.

```
. TEST/JCL                    . TEST/JCL
//IF A                        //IF A
//SET P1                      //ASSIGN P1=PROGRAM/BAS
//SET KI                      //ASSIGN KI=ALL
//SET PR                      //ASSIGN PR=80
//END                         //END
```

Logically speaking, these two examples are identical. If the token "A" is true, the tokens P1, KI, and PR will all be set to true. Additionally, the //ASSIGN macro example will assign character string values to the tokens. However, these character string values will have no effect on the logical value.

```
//. COMMENT
//QUIT
```

So far, the only method described for testing and debugging a JCL file has been to use the compile only DO command and then list the resultant SYSTEM/JCL file. The //. COMMENT and the //QUIT are provided to give you run time debugging. Unlike execution comments, the compilation comments are not written to the SYSTEM/JCL file. Rather, they are displayed on the screen immediately as encountered when the compiling is done. Thus, they can act as a visual status log of the compile. The //QUIT macro is used to abort the compiling if an invalid condition is detected. This gives you the ability to make sure all needed tokens are entered before any execution takes place.

```
. START/JCL
set *ff forms                          [DOS 6]
filter *pr *ff                         [DOS 6]
filter *pr pr/flt                      [DOS 5]
forms (lines=60)
//IF KI
set *ki ki (type)
//ELSE
//. KI was not entered!
//QUIT
//END
basic
```

```
run"program/bas"
//STOP
```

If this JCL file was compiled without the token "KI" being entered on the DO command line, the screen display would show:

```
//. KI was not entered!
//QUIT
```

No actual lines would be executed from the SYSTEM/JCL file, as the compile phase was aborted before completion. The compilation comment tells the operator why the abort took place. The //QUIT macro may seem very similar to the //ABORT execution macro. The main difference is when the actual abort takes place. Substituting //ABORT for //QUIT in the previous example, and then doing the JCL without the token "KI" would produce the following screen display:

```
//. KI was not entered!
. START/JCL
set *ff forms                          [DOS 6]
filter *pr *ff                         [DOS 6]
filter *pr pr/flt                      [DOS 5]
forms (lines=60)
//ABORT
```

As you can see, the comment line will be displayed as the compiling is taking place. However, since **//ABORT** is an execution macro, the SYSTEM/JCL file will execute until it reaches the //ABORT line! This means that any executable lines up to that point will be executed. In this case, the forms filter program would have been applied. Now, if you would do the same JCL file again, specifying the "KI" token on the command line, problems will occur. Since the forms filter program cannot be applied if it is already active, the JCL will abort again when it tries to execute the FILTER line for the second time. As you can see, the //QUIT macro definitely should be used rather than the //ABORT.

JCL substitution fields

Perhaps the most powerful feature of the JCL language is the ability to substitute and concatenate character strings to create executable lines. The character strings can be entered as token values on the DO command line,

or can be set with the //ASSIGN macro. A substitution field is created by placing pound signs "#" around a token. For example:

```
. TEST/JCL
forms (chars=#C#)
basic
run"#P1#"
//STOP
```

This example uses two substitution fields; one in the FORMS command line representing the number of characters, and one in the "run"program"" line representing the name of the program. If the DO command "DO TEST (C=132,P1=PROGRAM1)" were used, the lines written to the SYSTEM/JCL file would be:

```
. TEST/JCL
forms (chars=132)
basic
run"PROGRAM1"
//STOP
```

As you can see, the compile phase substituted the character string value of the tokens into the actual command line! In effect, you could set any valid number of characters for the FORMS command and run any program simply by specifying different values for the C and P1 tokens. This example brings out another important point - the number of characters in the replacement string can be less than, equal to, or greater than the length of the token name in the replacement field between the "#" signs.

To reduce the number of tokens needed on the DO command line, and to increase the program options at the same time, the //ASSIGN macro can be used as follows:

```
. TEST/JCL
//ASSIGN c=80
//ASSIGN p1=program1
//IF num2
//ASSIGN c=132
//ASSIGN p1=program2
//END
```

```
forms (chars=#C#)
basic
run"#P1#"
//STOP
```

In this example, the DO command would not have to specify any tokens if the default of the 80 character printer filter and PROGRAM1 were desired. Otherwise, specifying NUM2 would override the defaults. The values of C and P1 would automatically be set with the //ASSIGN tokens inside the //IF conditional block.

Another very practical use of the substitution field feature is for replacing drive specifications. The following example shows how a FORMAT and BACKUP JCL file could be structured:

```
. FB/JCL, FORMAT with BACKUP
//PAUSE insert disk to format in drive #D#
format :#D# (name="data1",q=n,ABS)
backup :#S# :#D# (mod)
//EXIT
```

In this example, the token "D" could be used to represent the destination drivespec, and the token S the source drivespec. Entering the command "DO FB (S=1,D=2)" would first pause the JCL and prompt you to insert a disk in drive :2. As you can see, the substitution fields can be used in message lines and comments as well as in executable command lines. After pressing <ENTER>, the JCL would continue, formatting the disk in drive :2, and then executing the backup command with drive :1 as the source drive and drive :2 as the destination drive.

Because the "#" sign is used to mark the start of a substitution field, some caution is necessary when trying to display a single "#" in a comment or message. Consider the following example.

```
//PAUSE Insert a disk in drive #1
```

If the JCL file was execute only, this line would be properly displayed. However, if the JCL were compiled, an error would occur. For this line to be properly displayed in a compiled JCL, it would have to be written as:

```
//PAUSE Insert a disk into drive ##1
```

The double pound sign is a special case, and lets the JCL compiler know that you wish a single "#" sign to be displayed, and do not wish to start a substitution field.

Another practical use for substitution fields is copying password protected files from one drive to another. In this example, a group of files will be copied from drive :0 to a drive specified in the DO command. Also, the user will have to supply the proper password for the copies to work.

```
. MOVE/JCL file transfer
copy program1.#P#:0 :#D#
copy program2.#P#:0 :#D#
copy program3.#P#:0 :#D#
copy program4.#P#:0 :#D#
//EXIT
```

This JCL would be done with a command such as "**DO MOVE (D=2,P=SECRET)**". Now, as long as the password for the files were SECRET, the JCL would move the files from drive :0 to drive :2. If the wrong password was used, the appropriate DOS error would be displayed and the JCL would abort.

Substitution fields can also be added together, or concatenated, to create new fields. The next example shows how this is done.

```
. ADD/JCL
copy #F##E#:0 :1
copy #F1##E#:0 :1
//EXIT
```

This example uses two substitution fields, one for the filename and one for the extension. The results of a DO command such as "**DO ADD (F=SORT,E=/CMD,F1=SORT1)**" would produce the following SYSTEM/JCL file after compiling:

```
. ADD/JCL
copy SORT/CMD:0 :1
```

```
copy SORT1/CMD:0 :1
//EXIT
```

As in previous examples, the //IF and //ASSIGN macros could be used to allow a single token to select the "F", "F1", and "E" tokens.

Combination of files

Most of the JCL examples in the previous sections have been very short. In a practical operating environment, this is often the case. However, each of these small files is taking up the minimum disk allocation of one gran and using one directory entry. Also, you may sometimes wish to duplicate a JCL file inside of another, without having to retype the lines. To allow this, the //INCLUDE macro and the LABEL feature of JCL can be used.

//INCLUDE

The //INCLUDE macro is used to merge together two or more JCL files during the compile phase. The correct syntax is:

```
//INCLUDE filespec
```

Before describing the //INCLUDE macro any further, one point must be emphasized - an //INCLUDE macro **cannot** be the last line in a JCL file. If it is, a "Record number out of range" error will occur, and the JCL will abort.

The filespec would be the name of the JCL file to be included. This command is similar to specifying the filespec in a DO command line. However, it is not allowable to enter tokens or other information after the file name, and any information after the filespec will be ignored. If you need to pass tokens to the included program, they will have to be established in the program that is doing the //INCLUDE. This next example will show two JCL files and the results of the compile phase.

```
. TEST1/JCL
. comment line 1
//INCLUDE TEST2
. comment line 2
//EXIT
```

```
. TEST2/JCL
. This comment is included
```

The command "DO TEST1" would produce the following SYSTEM/JCL file:

```
. TEST1/JCL
. comment 1
. TEST2/JCL
. This comment is included
. comment line 2
//EXIT
```

As you can see, the compiling starts with the file named in the DO command line. As soon as the //INCLUDE is reached, all lines in the second JCL file are processed, and then the compiling returns to the rest of the original file. There is no limit to the number of //INCLUDE macros you can use other than having enough disk space for the resulting SYSTEM/JCL file.

For example, let us assume that the TEST2/JCL file contains a procedure that you wish to repeat three times, with pauses in between. You could re-write the TEST1/JCL file as follows:

```
. TEST1/JCL
//PAUSE Initial pass now ready
//INCLUDE TEST2
//PAUSE Get ready for pass 2
//INCLUDE TEST2
//PAUSE Get ready for pass 3
//INCLUDE TEST2
//EXIT
```

As should be evident, this JCL will compile to a series of pauses with the TEST2 procedure done after each pause.

JCL labels

The LABEL feature of JCL will allow you to permanently merge together many small JCL procedures into one large file, and then access those procedures individually. This will save disk space and directory entry space for you. The format for a LABEL is:

@LABEL

The label name can be up to eight characters long, either upper or lower case letter "A-Z" or "a-z", or the numbers "0-9". Following is a brief example of a JCL file containing labels:

```
. TEST/JCL label example
@FIRST
. this is the first procedure
//exit
@SECOND
. this is the next procedure
@THIRD
. this is the last procedure
```

This file contains three labels. To select any procedure, specify the label on the DO command line. "**DO TEST (@FIRST)**" would start compiling with the first line after the @FIRST label. The following rules determine how much of a labeled JCL file will be included in the compile phase:

- If no label is specified on the DO command line, all lines from the beginning up to the first label will be compiled.
- If a label is specified, compiling will include all lines until the next label or the end of the file is encountered.

Doing the TEST/JCL file using the **@FIRST** label would write the first comment and the **//EXIT** macro to the SYSTEM/JCL file for execution. Specifying either of the other labels would include only the appropriate single comment line. If the file were done with no label specified, only the initial execution comment "**. TEST/JCL label example**" would be written out.

There is no limit to the size of labeled procedures. They may range from one to as many lines as you can fit on your disk. The only requirement is that a JCL file containing labels must be compiled.

When using labels in a JCL file, one word of caution is necessary. It is recommended that the file start with a comment line or some executable line other than a label. Consider the following short example:

```
@FIRST
. Print this comment
```

Now, if a DO command were to do this file without specifying the @FIRST label, the following would result. First the compiling phase would get the first line, see that it is a label, and quit. This is normal, as the compiler will start with the first line and continue to the first label or the end of the file. Since the compile is complete, the SYSTEM/JCL file would be executed! In other words, whatever lines had been compiled to the SYSTEM/JCL file from a previous DO command would now be executed. Needless to say, this is **not** what normally is desired.

Advanced JCL compiling

The previous section on JCL compiling showed the basic uses of tokens and compilation macros. If you do not understand the *SIMPLE JCL COMPILING* section, please re-read it. If you actually type in and try the examples, you should have an understanding of how to structure a compiled JCL file. This section will describe additional features, and show different ways to accomplish logical decision branching.

Using the logical operators

There are three logical operators available for use with the //IF macro. These operators will specify the type of logical testing of the tokens. They are "AND", "OR", and "NOT", represented as follows:

AND	is represented by the ampersand "&".
OR	is represented by the plus "+".
NOT	is represented by the minus "-".

All previous examples of //IF have tested the logical truth or falseness of a token, such as "**//IF token**". By using the logical operators, more complex and more efficient testing can be done. Consider the following series of examples using the tokens A and B:

```
//IF A                    "if A" - true only if A is true
. include these lines
//END
```

```
//IF -A                    "if not A" - true only if A is false
. include these lines
//END
```

By using NOT, it is possible to see not only if a token is true, but to see if it is false. This provides an alternative method to select a block of lines for compiling.

```
//IF A+B                   "if A or B" - true if either is true.
. include these lines
//END

//IF A&B                   "if A and B" - true only if both are true
. include these lines
//END
```

These examples show how multiple tokens may be tested in a single //IF statement. In the OR example, the //IF will test true if either "A" or "B" were true. The AND example requires that both "A" and "B" be true to include the lines up to the //END. It is allowable to use any combination of logical operators in an //IF statement. When doing so, it is important to know how the statement will be evaluated.

- Evaluation of the statement will be from left to right.
- In the evaluation of any compound expression, the expression will be FALSE if the left hand symbol of an AND is FALSE.
- Parentheses are not allowed, and will abort the JCL compiling.
- All logical operators have the same precedence.

Following are some examples of //IF statements using multiple logical operators:

```
//IF A+B+C          if either A or B or C is true

//IF -A&-B          if A is false and B is false

//IF -A&B+C         If A is false and either B or C is true
```

As you can see, the logical operators can be combined to test almost any arrangements of tokens you may need. This is especially handy for setting

up default conditions and in checking for missing tokens, as the following examples will demonstrate.

```
. CHECK/JCL
//IF -S
//ASSIGN S=0
//END
//IF -D
//ASSIGN D=2
//END
```

```
. CHECK1/JCL
//IF -S+-D
//. You MUST enter S and D!
//QUIT
//END
```

Let us assume that the "S" and "D" in these two examples will be used as source and destination drivespecs later in the file. The CHECK example tests "S" and "D" individually, and assigns them default values if they were not true. The CHECK1 example, on the other hand, is structured so that both "S" and "D" must be true, or the JCL will abort. The //IF line in the CHECK1 example reads "if not S or not D". Although the use of logical operators may seem harder to understand than a simple "//IF token" statement, it does provide easier ways to determine if needed tokens have been specified.

Nested //IF macros

By definition, a conditional block begins with an //IF and concludes with an //END. When the //IF evaluates true, the lines between the //IF and the //END are compiled. It is also possible to include other //IF-//END blocks within the main conditional block, in effect nesting them. As previously explained, the //ELSE macro provides an alternative course of action in case an //IF evaluates false. It is also allowable to have more //IF-//END statements following the //ELSE. Refer to the following examples:

```
. TEST/JCL
//IF A
. comment 1
//ELSE
//IF B
. comment 2
//END                    (ends the //IF B statement)
//END                    (ends the //IF A statement)
```

The TEST example is fairly straight forward. If "A" evaluates true, comment 1 would be written out, and the //ELSE would be ignored. If "A" was false, "B" would be tested. Comment 2 would be written out only if "B" was true. Notice the two //END macros. As stated earlier, there must be one //END for every //IF. What might not be readily apparent is which //END matches which //IF.

In this example, there are comments in parentheses to show the way the //ENDs correspond to the //IFs. It is allowable to use this type of comment identifier in real JCL files. You will find that labeling //END macros greatly increases the readability of the file, especially when editing a file that you have created some weeks (or months) previously.

This next example and the following description again show how nested //IFs are evaluated.

```
//IF A                    First IF
. Comment A
//IF B                    Second IF
. Comment B
//IF C                    Third IF
. Comment C
//END                     ends Third IF
//END                     ends Second IF
. Comment D
//END                     ends First IF
```

Evaluating this example produces the following results. When the first //IF is false, all lines up to the corresponding //END will be ignored. Since the last //END corresponds to the first //IF, none of the lines in this example would be written out to the SYSTEM/JCL file.

Assuming from this point on that the first //IF evaluates true, two lines will always be written out. These are the Comment A and Comment D lines.

The first nest is //IF B. If "B" is true, the Comment B line will be written out. If "B" is false, all lines, including the //IF C block, will be skipped up to the //END corresponding to the //IF B.

The next nest is //IF C. The only time this will be considered is if both "A" and "B" have tested true. As normal, if "C" is true the Comment C will be written out.

Although not shown in the example, it is perfectly allowable to use the logical operators when nesting //IFs. Again, note the use of the comments after the //END macros. Using comments such as these will help you follow the logic flow, especially until you become familiar with using nested //IF macros.

Nested //INCLUDE macros

When using the //INCLUDE macro, it is allowable for the included file to also contain another //INCLUDE macro. This is referred to as nesting. Briefly stated, the following rules will apply:

- The maximum nest level will be ten active //INCLUDE macros.
- An //INCLUDE macro cannot be the last line in a JCL file.

The following example uses three files to show how the lines in nested //INCLUDE files are processed:

```
.File #1
//. NEST0/JCL
. nested procedure example
//INCLUDE nest1
. this is the end of the primary JCL
//EXIT

.File #2
//. NEST1/JCL
. this is the first nest
//INCLUDE nest2
. this is the end of the first nest

.File #3
//. NEST2/JCL
. this is the second nest
```

The above will result in a nest level of two (two pending //INCLUDEs). If these three JCL files are saved as NEST0/JCL, NEST1/JCL, and NEST2/JCL, and the NEST0/JCL is compiled and executed, it will result in the following dialogue:

```
//. NEST0/JCL
//. NEST1/JCL
//. NEST2/JCL
. nested procedure example
. this is the first nest
. this is the second nest
. this is the end of the first nest
. this is the end of the primary JCL
```

The three compilation comments will be shown immediately as the JCL file is compiled. When the compilation phase is complete, the compiled SYSTEM/JCL file will be executed. In this example, the execution phase will merely display a series of execution comments. As you can see from the order of the displayed comments, the files are executed similarly to nested FOR-NEXT loops in BASIC. After all //INCLUDEs are detected, the innermost (last encountered) //INCLUDE file completes execution first, with execution proceeding back towards the original //INCLUDE.

The //INCLUDE macro can very easily be used to compile a large JCL procedure from a series of smaller JCL routines. If the finished SYSTEM/JCL file is a procedure that will be executed many times, it may easily be saved by copying SYSTEM/JCL to a file with another name.

Using the special "%" symbol

The "%" symbol is used to pass character values to the system as though they came from the keyboard. The proper syntax is the "%" symbol directly followed the hexadecimal value of the character, such as %1F. The following values are all valid after the "%" symbol:

HEX VALUE	RESULT
09	TAB eight Spaces
0A	Line Feed
1F	Clear Screen

Also, the value of any printable character may be used, although this is not normally done.

When using the clear screen character, it should be placed at the start of a line. For example:

```
%1F. This is a comment line
%1F//PAUSE Insert disk in drive 1, press <ENTER>
```

In both examples, the screen will clear and the JCL line will be displayed in the top left corner of the screen. The TAB (09H) and line feed (0AH) characters can be used to position comments or other lines in different positions on the screen. These characters should always be placed after the period in the comment line, or after the macro in an executable line. For example:

```
.%09%09 This comment will be tabbed 16 spaces
//PAUSE %0A%0A%0A This line will appear 3 lines down
```

When this file is compiled and executed, the comment line will be tabbed over sixteen places. Notice that the first "%" is after the period in the comment. If the symbol were before the period, DOS would not recognize it as a comment line and the JCL would abort. In the //PAUSE line, the //PAUSE would be displayed, and the remaining message line would be displayed three lines lower on the screen. Using the tab and linefeed characters in this manner can sometimes help improve the readability of the messages displayed during JCL execution.

Although any other ASCII character may also be sent to the keyboard, the system generally will not respond to any other characters less than a space (X'20'). Characters above this value may be used with the "%" symbol, but it is easier merely to type them in as a command line in the JCL file.

Interfacing with applications programs

This section will describe how to use JCL to start up and even control applications programs. After reading this section, it should be very easy for a user to interface between an application and the JCL processor. Two languages will be discussed - BASIC and Z-80 assembler.

Interfacing with BASIC

A JCL file is the perfect method to interface between the operating system and the BASIC language. JCL can be used to create procedures that require only the insertion of a diskette to start up a program. Additionally,

DOS version 5 users may fully utilize the features of JCL from within a BASIC program; DOS 6 users can invoke a DO command from BASIC, but control will not return to BASIC once the command stream completes.

To use a JCL file to initiate an automatic start up of an BASIC program, it will be necessary to use the AUTO library command to execute a JCL file. Assuming the JCL file is named BAS/JCL, issuing the command **"AUTO DO BAS/JCL"** will automatically invoke the desired BASIC program every time the computer is booted with the AUTOed system disk.

The actual JCL file should be sequenced as this next example shows:

```
Establish necessary drivers, filters, or other DOS options.
BASIC (any needed parameters)
RUN"PROGRAM/BAS"
//STOP
```

This example shows the normal way to execute a BASIC program from a JCL file. Any necessary system options are established, BASIC is entered with any necessary parameters (such as memory size and number of files), and the BASIC program is loaded and executed. Notice the termination macro **//STOP** used in the JCL file. As explained in the *JCL EXECUTION* section, if this macro was not used or if the **//EXIT** macro was used instead, the JCL file would return to the DOS Ready prompt as soon as the first keyboard entry was requested. The **//STOP** macro will terminate the JCL execution and leave keyboard control with BASIC.

It is not necessary to use the AUTO library command when using a JCL file to execute a BASIC program. The DO command may be entered directly from the DOS Ready prompt, such as **DO BAS/JCL.**

To execute a JCL file once you have entered BASIC, the command format is:

```
CMD"DO filename"                    [DOS 5]
SYSTEM"DO filename"                 [DOS 6]
```

This command can be typed in directly or may be entered as a BASIC program line. As with any DOS version 5 **CMD"dos command"** function invoked from BASIC, it will be necessary to have approximately

4K of free memory available or an "Out of memory" error will occur. DOS 6 DOS BASIC can only invoke a DO file; it cannot return to BASIC.

Any JCL file that will be called from DOS 5 BASIC should have the //EXIT termination macro so control will return to BASIC when the JCL is completed. For example, suppose you wished to use the JCL //ALERT macro to inform you when a lengthy BASIC procedure had completed. After the lines containing the BASIC procedure, you could have an BASIC program line such as:

```
1000 CMD"DO =ALERT/JCL:0"
```

which might execute the ALERT/JCL file:

```
. Your procedure is complete , press <ENTER> to resume
//ALERT (1,0,7,0)
//EXIT
```

When BASIC reached line 1000, the JCL file ALERT/JCL would be executed. This would send a series of repeating tones out the cassette port or Model 4 sound port. Assuming you had a suitable amplifier hooked to the cassette cable, you would be notified you that your BASIC procedure had completed. Pressing **<ENTER>** would end the JCL alert and return you to BASIC. There are two important points to be made about this example. First, the comment line in the ALERT/JCL file is absolutely necessary, as a JCL file cannot start with an execution macro. Second, the //EXIT termination macro is mandatory to assure that keyboard control will be returned to BASIC.

Although the example demonstrated an execute only JCL file, it is perfectly allowable to call compiled JCL procedures from BASIC. You may even construct a **CMD"DO filename (parameters)"** command using BASIC string substitution.

Anytime you wish to use a DOS version 5 **CMD"DO filename"** command to execute a JCL file and not return to BASIC, you will have to change the format of the command. This is especially important if the new JCL file is one that will also enter BASIC and run a program. To do these types of JCL files from BASIC, use the format:

```
CMD"I","DO filename"
```

Using this format for the command will assure that a proper exit is made before the new JCL file is started.

Controlling a BASIC program

In some cases, the prompts in a BASIC program can be answered with a line from a JCL file. This will be true if the program uses the INPUT or LINEINPUT BASIC statement to take the input. If the INKEY$ statement is used, response will have to come from the keyboard rather than from a JCL file. If the program is using the proper input method, creating a JCL for totally hands-off operation can be done as follows:

- Run through the program normally, making note of every prompt to be answered.
- Create a JCL file to enter BASIC and run the program as explained above in the BAS/JCL example, leaving off the //STOP macro.
- Now, add the responses to the prompts as lines in the JCL file.

Using this method will provide automatic program execution. All that is required is for you to have the proper responses for any program prompts as lines in the JCL file. Terminating the JCL file will depend on what needs to be done when the application program has completed. If you desire to run more programs, you could add the proper RUN"PROGRAM" line to the JCL file, followed by any needed responses to program prompts. If you desired to return to the DOS Ready mode, you could end the file with the //EXIT macro. Ending the file with a //STOP would leave you at the BASIC Ready prompt when the program completes.

Interfacing with assembler

All programs that utilize the line input handler (@KEYIN service call) will be able to accept "keyboard" input from the JCL file, just as though you typed it in when the program was run. This gives the capability of pre-arranging the responses to a program's requests for input, inserting the responses into the JCL file, initiating the procedure, then walking away from the machine while it goes about its business of running the entire job.

Keyboard input normally handled by the single-entry keyboard routines (@KBD, @KEY, and BASIC's INKEY$) will continue to be requested

from the keyboard at program run time and will not utilize the JCL file data for input requests. Thus by understanding fully the dynamics of JCL processing, you can write applications that take full advantage of the power inherent in the Job Control Language.

Practical JCL examples

It is virtually impossible to explore all the possibilities that exist concerning the creation of JCL files. The examples that follow will give you some ideas as to how JCL may be used to make your day to day operating of the DOS system even more efficient.

Example #1

This example will show you how to SYSRES system modules using a JCL file. The modules that will be resided are 2, 3, 8 and 10. These modules are required to be resident in order to perform a backup by class between two non-system diskettes in a two drive system. The JCL file used to perform such a function may look something like this:

```
. BURES/JCL - JCL used to SYSRES system modules 2,3,8,10
SYSTEM (SYSRES=2)
SYSTEM (SYSRES=3)
SYSTEM (SYSRES=8)                                    [DOS 5]
SYSTEM (SYSRES=10)
. end of BURES/JCL
```

When executed, this JCL file will cause the system modules 2, 3, 8 [if DOS 5], and 10 to be resident in high memory. Because this JCL uses no labels or compilation macros, the compilation phase may be skipped.

Example #2

This example will show you a JCL file that may be used to perform diskette duplication. A minimum of three drives will be required. Drive :0 will contain a system diskette with the JCL file. Drive :1 will be the source diskette of the backup. Assume that the diskette name is MYDISK, and it is a single sided, 40 track, double density diskette, with a master password of PASSWORD. Drive :2 will be used as the destination of the backup. It may contain either a new, unformatted diskette or a diskette which has

been previously formatted and contains information. The following JCL may be used to perform such a duplication:

```
. DUPDISK/JCL - Disk duplication JCL
//pause Source in 1, Destination in 2, <ENTER> when ready
format :2 (name="mydisk",dden,cyl=40,q=n,abs)
//pause format ok? <ENTER> if yes, <BREAK> if no
backup :1 :2
do *
```

The second line of the JCL will cause the computer to pause until the <ENTER> key is pressed. This will allow you to insert the proper diskettes into drives :1 and :2. Once this has been done, you may press <ENTER>, and the third line of the JCL will be executed.

The format line passes the parameters NAME, DDEN, and CYL to the format utility. Note that the number of cylinders, diskette name and diskette password of the destination diskette must be an exact match of the source disk. If they do not exactly match, the backup command that follows will issue some type of unwanted prompt, which would cause the JCL to abort. Also, note that the parameters Q=N and ABS were specified. Both are necessary. The Q=N parameter causes the computer to use the default of PASSWORD for the master password, and hence the "Master Password" prompt will be bypassed. The ABS parameter ensures that no prompt will appear if the destination diskette contains data.

The pause after the format statement allows you to check whether or not the format was successful. If the destination diskette was formatted properly, you may press <ENTER> to continue the JCL. If tracks were locked out during the format, you may press <BREAK>. Realize that doing so will cause the JCL to abort, and it will be necessary to restart the JCL activity.

After <ENTER> has been pressed in response to the second pause, the backup will take place. If any error occurs during the backup, the JCL will be aborted.

Once the backup has been completed successfully, the comment line will appear, and the DO * command will be executed. This command will cause the SYSTEM/JCL file to be executed. Realize that if this is to be a duplicating JCL, the compilation phase cannot be skipped.

ALPHANUMERIC - consisting of only the letters A-Z, a-z, and the numerals 0-9.

ASCII - The alphanumeric representation of controls and characters as a single byte, falling within a range from 1 to 127 (sometimes including 0).

ASCII files - Files generally containing only ASCII characters.

BACKGROUND TASK - A job that the computer is doing that is not apparent to the user or does not require interaction with the user. Some examples are the REAL TIME CLOCK, the SPOOLer and the TRACE function.

BAUD - A term that refers to the rate of serial data transfer.

BIT - One eighth of a byte, one binary digit.

BOOT - The process of reseting a computer and loading in the resident operating system from the system drive.

BUFFER - An area in RAM that will temporarily hold information that is being passed between devices or programs.

BYTE - The unit that represents one character to your computer. A byte consists of eight binary "bits" that are either ON (1) or OFF (0). One byte can represent a number from 0 to 255.

CONCATENATE - To add one variable or string onto the end of another.

CONFIGURATION - The status of the system and physical devices that are available to it. This configuration may be dynamically changed with several library commands and the SYSTEM command, and may be saved with a SYSGEN. If the system is SYSGENed, that configuration will be re-established each time the machine is re-booted or re-started.

CURSOR - The location on the video display where the next character will be printed. It will be marked by the presence of a cursor character.

CYLINDER - All tracks of the same number on a disk drive. On single sided drives, cylinders will synonomous with tracks. On a double sided

drive, there will be two tracks per cylinder, one on the front and another on the back of the diskette.

:d - This indicates that a drive spec (number) may be inserted where this is used. A drive spec must always be preceded immediately by a ":" as shown. If a drive spec is not to be given, then the ":" must not be used.

DAM (Data Address Mark) - An identifying data byte put on a diskette to mark the difference between data and directory cylinders.

DCB - Device Control Block, a small piece of memory used to control the status and the input and output of data between the system and the devices.

DCT - Drive Code Table, a piece of memory containing information about the disk drives and/or diskettes in them.

DENSITY - Refers to the density of the data written to a diskette. Double density provides approximately 80% more capacity than single density.

DEVICE - A physical device located outside of the CPU (Central Processing Unit), whose purpose is to transmit/receive data to/from the operating system. The operating system is in total control of any activity directed to/from a device.

devspec - The name associated with a device by which it is referenced. A "devspec" will always consist of three characters; the first of which is an asterisk, followed by two upper case alphabetic characters.

DIRECTORY - A cylinder on a diskette used to store information about a diskette's free and used space and file names.

DRIVER - A machine language program used to control interactions between the operating system and a device.

***DO** - A DOS system device, the Video Display.

EOF - End Of File, a marker use to denote the end of a program or data file.

/ext - The extension of a filespec. The use of /ext is sometimes optional. An extension (if used), must contain as its first character a "/" (slash), and may be followed by one to three alphanumeric characters.

FCB - File Control Block, a small piece of memory used to control the status and the inputting and outputting of data between the operating system and disk files.

filespec - The name by which a disk file is referenced. A "filespec" consists of four fields and two switches, of which the first field is always mandatory. A filespec is designated by the following format:

!filename/ext.password:d!

"!" - (preceding filename) is an optional switch. If this switch is set, filespec is taken to be absolute. This allows the accessing of a filespec that would otherwise be inaccessible (i.e. a filespec that is the same as an DOS library command).

filename - The mandatory name of the file

/ext - The optional file extension

.password - The optional file password

:d - The optional drive specification

"!" - (following :d) is an optional switch. If this switch is set, the end of file marker for file (filespec) will be updated after every write to the file.

filename - The mandatory name used to reference a disk file. A filename consists of one to eight alphanumeric characters, the first of which must be alphabetic.

FILTER - A program which monitors and/or alters I/O that passes through it. "FILTER" is also the library command which establishes a FILTER routine.

/FIX - The desired file extension for a PATCH file.

FOREGROUND TASK - Jobs the computer does that are apparent to the user, such as running an applications program or a utility and interacting directly with the user.

GRAN - An abbreviation used for the term granule. A "gran" is the minimum amount of storage used for a disk file. As files are extended, file allocation is increased in increments of granules. The size of a "gran" varies with the size and density of a diskette.

**HIGH$** - "High Dollars", a pointer used to mark the highest unused memory address available for use. Any machine language programs loaded above HIGH$ will never be overwritten by DOS.

I/O - The abbreviation for Input/Output.

INTERRUPT - an interruption of the system generated by a hardware clock. The interrupt periods are used by DOS with such functions as type ahead and the printer spooler.

/JCL - The desired file extension for a DO file. "JCL" is an abbreviation for Job Control Language.

***JL** - A DOS system device, the Joblog.

***KI** - A DOS system device, the Keyboard.

/KSM - The desired file extension for a KSM file. "KSM" is an abbreviation for Key-Stroke Multiply.

(L)COMM - A communications program capable of interacting with disk, printer, video, keyboard and the serial interface. COMM will dynamically buffer all the system devices.

LIBRARY - A set of commands used to perform most operating system functions.

load module format - A file format that loads directly to a specified RAM address.

LSB - The Least Significant Byte in a hexadecimal word, sometimes referred to as the "low order byte".

MACRO - Statements or verbs used in JCL.

MSB - The Most Significant Byte in a hexadecimal word, sometimes referred to as the "high order byte".

MOD DATE - The date a file was last written to.

MOD FLAG - A "+" sign place after a filename to indicate that the file was written to since its last backup.

NIL - A setting of a device, indicating an inactive state. All I/O to/from this device will be ignored.

NRN - An acronym for "Next Record Number" which indicates the number of the next 256-byte record to be accessed from a file.

PACK I.D. - A diskette's name and master password assigned during formatting.

PARSE - The breaking up of a parameter line into its individual parameter values.

partspec - An abbreviation representing "PARTial fileSPEC". A partspec may be used with certain commands in lieu of a filespec. A partspec may be composed of any combination of the four fields defining a filespec. No leading or trailing "!" may be contained in a partspec.

In addition, the filename and /ext fields of a partspec may be abbreviated with leading information and/or "wildcarded". Examples of partspecs are given in this manual where applicable.

-partspec - Identical to a partspec, except that it is used as exclusion criteria during certain functions.

PARAMETER - The information that follows a library command or a utility, on the command line. This information is passed to the job that will be executed to tell the job how you wish execution to take place. Parameters usually follow the command and are enclosed in parentheses.

parm - The abbreviation for parameter described above.

.password - The password associated with a filespec, the use of which is optional. A password (if used), must contain as its first character a "." (period) , and may be followed by one to eight alphanumeric characters, the first of which must be alphabetic.

PATCH - A utility to make minor alterations to disk files.

***PR** - A DOS system device, the Line Printer.

RAM - Random Access Memory. In the TRS-80, the free user memory in the Computer unit.

ROM - Read Only Memory. In the TRS-80 Models I and III, the BASIC language and drivers stored in the Computer unit.

SECTOR - A contiguous block of disk storage, defined to be 256 bytes, where each byte within the sector has an absolute location and byte identification number. All sectors have a predefined, absolute starting and ending location.

***SI** - A DOS system device, the Standard Input. It is not presently used by the DOS system.

***SO** - A DOS system device, the Standard Output. It is not presently used by the LDOS system.

SWITCH - A parameter with a definite setting, such as ON/OFF.

TOKEN - A variable used in JCL.

UTILITY - A program that provides a service to the user. Utility programs usually run "outside" of the operating system itself.

wcc - The abbreviation for WildCard Character. In DOS, the replacement of filespec characters with "$" during certain DOS commands.

WORD - Two bytes in HEXadecimal format X'nnnn'. Usually entered in reverse notation: low byte, then high byte (LSB,MSB).

G

H

I

J

K

T