Hyperion™

DOS (2.11)

Guide

620027-000-00

Hyperion

DOS (2.11)

Guide

This manual is a non-technical user's guide that describes the Disk Operating System (DOS), version 2.11, for the Hyperion personal business computer.

This user guide also describes four other commonly used programs: EDLIN, the single-line text editor; LINK, used to create application programs; DEBUG, used to step through and debug application programs; and LIB, used to create library files for use with application programs.

Published by: Comterm Inc.
1 July 1984
Version 00
Rev 00

Trademarks

Hyperion	is a trademark of Comterm Inc.
MS-DOS	is a trademark of Microsoft Corporation.
Microsoft	is a trademark of Microsoft Corporation.
MULTIPLAN	is a trademark of Microsoft Corporation.
IBM	is a trademark of International Business Machines Corporation.

Disclaimer

The information in this manual has been carefully prepared and checked for completeness and accuracy. There is however, always the possibility of omission or error. In such an event Comterm Inc. cannot assume liability for any damages resulting from the use of this manual.

Service Requirements

In the event of equipment malfunction, all repairs must be performed by Comterm Inc., an authorized agent (dealer) of Comterm Inc. or any other organization authorized by your warranty agreement.

Avoiding Radio-Television Interference

WARNING: This equipment has been certified to comply with the limits for a Class B computing device, pursuant to Subpart J of Part 15 of FCC Rules. Only peripherals (computer input/output devices, terminals, printers, etc.) certified to comply with the Class B limits may be attached to this computer. Operation with non-certified peripherals is likely to result in interference to radio and TV reception.

In order to comply with FCC emissions limits for Class B devices, this unit must be connected to peripherals only with shielded cables.

FFC NOTIFICATION
(applicable when a system is set up in the United States)

This equipment generates and uses radio frequency and if not installed and used properly, that is, in strict accordance with the manufacturer's instructions, may cause interference to radio and television reception. It has been tested and found to comply with the limits for a Class B computing device in accordance with the specifications in Subpart J of Part 15 of FCC Rules, which are designed to provide reasonable protection against such interference in a residential installation. However, there is no guarantee that interference will not occur in a particular installation. If this equipment does cause interference to radio or television reception, which can be determined by turning the equipment off and on, the user is encouraged to try to correct the interference by one or more of the following measures:

- Reorient the receiving antenna
- Relocate the computer with respect to the receiver
- Move the computer away from the receiver
- Plug the computer into a different outlet so that the computer and receiver are on different branch circuits.

If necessary, the user should consult the dealer or an experienced radio/television technician for additional suggestions. The user may find the following booklet prepared by the Federal Communications Commission (U.S.A.) helpful:

"How to Identify and Resolve Radio-TV Interference Problems"

This booklet is available from the U.S. Government Printing Office, Washington, D.C. 20402, Stock No. 004-000-00345-4.

WARNING: This equipment has been certified to comply with the limits for a Class B computing device, pursuant to Subpart J of Part 15 of FCC Rules. Only peripherals (computer input/output devices) certified to comply with the Class B limits may be attached to this computer. Operation with non-certified peripherals is likely to result in interference to radio and TV reception.

In order to comply with FCC emissions limits for Class B devices, this unit must be connected to peripherals only with shielded cables.

For users connecting registered equipment to the telephone network:

1. This terminal equipment may not be used with party lines or coin lines.

2. If problems arise with the system, the registered equipment shall be disconnected from the telephone line to determine if the registered equipment is malfunctioning, and if it is malfunctioning, the use of such equipment shall be discontinued until the problem has been corrected.

3. Before connecting the registered equipment to the telephone network, the telephone company must be provided with the following:

   - your telephone number
   - the FCC registration number
   - the ringer equivalence number
   - the USOC jack required

   The latter three items are indicated on the equipment label. The telephone company should be notified when the equipment is permanently disconnected from the line.

DOC NOTIFICATION
(applicable when a system is set up in Canada)

NOTICE: The Canadian Department of Communications label identifies certified equipment. This certification means that the equipment meets certain telecommunications network protective, operational and safety requirements. The Department does not guarantee the equipment will operate to the user's satisfaction.

Before installing this equipment, users should ensure that it is permissible to be connected to the facilities of the local telecommunications company. The equipment must also be installed using an approved method of connection. In some cases, the company's inside wiring associated with a single line individual service may be extended by means of a certified jack-plug-cord ensemble (telephone extension cord). The customer should be aware that compliance with the above conditions may not prevent degradation of service in some situations. Existing telecommunications company requirements do not permit their equipment to be connected to customer-provided jacks except where specified by individual telecommunications company tariffs.

Repairs to certified equipment should be made by an authorized Canadian maintenance facility designated by the supplier. Any repairs or alterations made by the user to this equipment, or equipment malfunctions, may give the telecommunications company cause to request the user to disconnect the equipment.

Users should ensure for their own protection that the electrical ground connections of the power utility, telephone lines and internal metallic water pipe system, if present, are connected together. This precaution may be particularly important in rural areas.

Caution: Users should not attempt to make such connections themselves, but should contact the appropriate electric inspection authority, or electrician, as appropriate.

TABLE OF CONTENTS

...continued

TABLE OF CONTENTS *(cont)*

...continued

TABLE OF CONTENTS *(cont)*

...continued

TABLE OF CONTENTS *(cont)*

TABLE OF CONTENTS *(cont)*

...continued

TABLE OF CONTENTS *(cont)*

TABLE OF CONTENTS *(cont)*

..continued

TABLE OF CONTENTS *(cont)*

TABLE OF CONTENTS *(cont)*

...continued

TABLE OF CONTENTS *(cont)*

...continued

TABLE OF CONTENTS (cont)

TABLE OF CONTENTS *(cont)*

...continued

TABLE OF CONTENTS *(cont)*

TABLE OF CONTENTS *(cont)*

SECTION PAGE

TABLES

Part I

Part II

Part III

Part IV

INTRODUCTION

Welcome to the Hyperion.

The Hyperion Disk Operating System (DOS) is a group of programs that provide an operating environment from which a user can access other programs, and from which a user can use files and diskettes to store information. This *Hyperion DOS (2.11) Guide* is divided into five parts:

- ***Part I – Basic Concepts,*** begins the reference part of the guide by explaining basic things about the Hyperion and its disk operating system, version 2.11.

- ***Part II – DOS Command Reference,*** describes each DOS command in detail. Examples are given for some of the more complex commands. Full descriptions are also given on the enhancements made to DOS to create this version, and on how to format a hard disk. The commands are listed alphabetically.

- ***Part III – EDLIN Command Reference***, describes each EDLIN command in detail. Examples are given for some of the more complex commands. The commands are listed alphabetically.

- ***Part IV – Other Programs Used with DOS***, is a detailed description of LINK, DEBUG, and LIB. These programs can be accessed from DOS, and enable a user to create and debug application programs written in Assembler, Pascal, C, and other programming languages.

- ***Part V – Useful Procedures***, describes the use of batch and configuration files.

The guide also contains appendices, an index, and a Quick Reference section. Appendix A outlines the enhancements made to this version of DOS (version 2.11). Appendix B describes technical specifications for the Hyperion. Appendix C is a copy of the Comterm Program License Agreement. The Quick Reference is a removable summary of DOS, EDLIN and other commands.

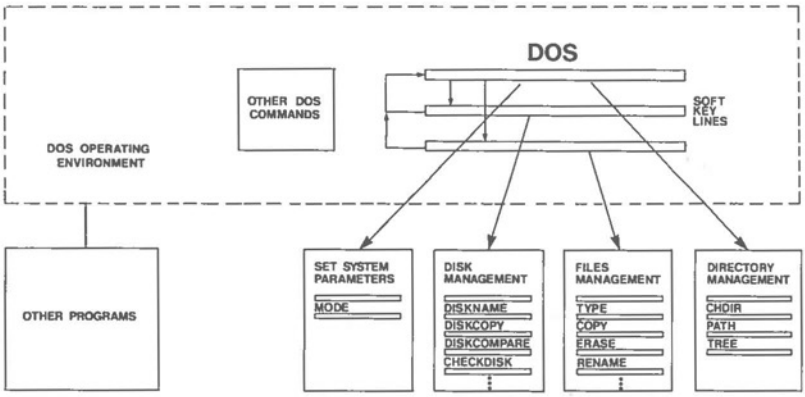


Fig. I – How DOS works. DOS has commands to perform 4 main types of functions. Some of the more commonly used DOS commands can be entered by using the soft keys. Others can only be typed in. DOS also provides an environment to let you run other programs.

The Disk Operating System (DOS) Version 2.11

The Hyperion uses MS-DOS as its diskette operating system. This operating system, from Microsoft Corporation, has defined a new standard for 16-bit microcomputers. The Hyperion is thus a member of a rapidly growing family of MS-DOS based systems, which includes the IBM Personal Computer.

From a user's (your) point of view, the Hyperion's Disk Operating System (DOS) is a collection of commands that enable you to:

a) maintain overall control of the operating system:

- setting certain system parameters (MODE),
- organize files into a tree-structured hierarchy of directories;

b) intelligently manage the disk(ette)s used to store information:

- prepare a new disk(ette) for use,
- give your disk(ette) a name,
- duplicate and compare disk(ette)s,
- find out how much free space is left on a disk(ette);

c) manipulate the files stored on a disk(ette):

- list the contents of a disk(ette),
- display, copy, erase, and rename files on a disk(ette).

DOS also provides an environment for the Hyperion that allows you to access other application programs (EDLIN, LINK, DEBUG, MULTIPLAN, BASIC, etc.)

Changes to DOS Version 2.11

Hyperion DOS version 2.11 has several major functional enhancements not found in earlier versions. Whether you are an experienced user of DOS or just starting out, we recommend that you read the section on Tree Structured Directories that follows this section, and Appendix A, "Enhancements to DOS Version 2.11".

EDLIN, the Single-line Text Editor

EDLIN is a single-line text editor which is provided to enable you to create and edit files. This text editor can be used to write memos, reports, letters, and longer documents. Programmers can use EDLIN to create data files and programs.

EDLIN is called a single-line text editor because it works with only one line of a file at any one time. Each line in the file can contain up to 256 characters of text.

Part III describes the EDLIN text editing commands, how to enter them, and what they do.

Other Programs

Other programs are also available. Though not essentially part of DOS, they are almost as necessary to the user.

These programs are described in detail in Part IV of this guide and are: LINK, the program that links smaller user-generated programs into complete application system packages; DEBUG, the program that enables users to debug their own programs; and LIB, which enables users to keep libraries of program subroutines.

Other Manuals for the Hyperion

The user guide is one of several available Hyperion manuals:

1) The ***Hyperion Setup Guide*** describes first-time setup procedures.

2) The ***Learning DOS 2.11*** tutorial booklet, which provides a hands-on introduction to DOS and EDLIN. Users unfamiliar with the Hyperion and DOS should read this booklet before using this manual.

3) This ***Hyperion DOS (2.11)*** Guide is third in the series.

4) A ***Hyperion BASIC Guide.*** This manual explains how to use the BASIC programming language.

5) The ***Hyperion EX User Guide*** describes the unpacking, installation and operation of the Hyperion expansion chassis and hard disk drive option.

6) Then there is a user guide written for each software system available for the Hyperion: **The Hyperion Text Editor**, word processor; **IN:TOUCH**, the communications management system; **MULTIPLAN™**, an electronic spreadsheet; and **1-2-3™**, an electronic spreadsheet with graphics capability and database management.

Things to Remember

When you are using diskettes, there are a few things to remember:

1) Your DOS is provided on a master DOS diskette. Always make a copy of your master DOS diskette as soon as possible and use this copy. Keep the original in a safe place and use it only when other copies are needed. The disk copy procedure is described in the "Learning DOS 2.11" tutorial booklet, and in Part II of this manual (the DISKCOPY command).

2) You must format every new, previously-unused diskette before using it with DOS. This procedure is described in the "Learning DOS 2.11" tutorial booklet, and in Part II of this manual (the FORMAT command).

Configurations and Conventions

DOS 2.11 is meant to run on all types of Hyperions, as well as on other computers compatible with the IBM Personal Computer.

Single- and double-drive Hyperions. In both cases the Hyperion software assumes that there are two diskette drives and calls them drive A and drive B, respectively. This means that, in a single-drive system, if prompted to insert a diskette into drive B, you would simply replace the diskette currently in drive A with the new diskette.

Using several keys simultaneously. Entering some commands may require pressing several keys simultaneously, or in succession. Many times you must hold down the **Ctrl** key while pressing another, such as **Break**. Throughout this guide, we have adopted the convention of linking the keys to be simultaneously pressed by a plus sign (+). Thus, **Ctrl + Break** means pressing the **Ctrl** and **Break** keys simultaneously.

Soft key lines. DOS, and many other software packages, display soft key lines across the bottom of the display screen. There are ten soft key labels, each containing a command. Pressing the corresponding soft key on the keyboard (F1 to F10) activates the command. In DOS, however, you have the option of displaying or not displaying this soft key line. If the soft key line is not displayed, remember that you can still enter DOS commands by typing in the appropriate command, plus parameters, and pressing the **Return** key.

Entering Commands. All commands are entered by typing in the command word and any other parameters, and then pressing the Return key. In this manual, we refer to "entering" as meaning typing in and pressing **Return**. When we ask you to type something in, do not press **Return**, until you are instructed.

Press any key. There are many instances where DOS prompts you to "Strike any key..." to recommence processing. This usually occurs when the system halts processing to allow you to switch diskettes or perform some other manual task. In this manual, we tell you to *Press any key* to continue. By this we mean press any *character* key, i.e. the grey keys on your Hyperion keyboard, or the **Return** key.

Part I

Section 1

INTRODUCTION TO PART I

This part of the guide describes certain basic concepts about DOS and the Hyperion personal business computer. Many of these concepts have already been covered in the accompanying tutorial booklet. If you are not familiar with microcomputers and the DOS operating system, we recommend you go through the tutorial booklet before proceeding.

This part of the user guide is organized in the following way:

- ***Section 2*** describes file management. Data is stored in files and this section covers file naming, and conventions concerning file names.
- ***Section 3*** shows you how to organize related groups of files into directories and subdirectories for easier access.
- ***Section 4*** shows you how to load and start DOS.
- ***Section 5*** describes the Hyperion keyboard and the function of each keyboard key.
- ***Section 6*** describes the Hyperion itself, including the rear panel connections. There is also a description of diskettes and how to use them.

...continued

Intro

Introduction to Part I *(cont)*

- ***Section 7*** describes the master DOS diskettes and their contents.
- ***Section 8*** introduces the MODE command and shows how it is used to configure the Hyperion serial and parallel interfaces, and the screen display.
- ***Section 9*** provides a step-by-step description on how to set up and format your hard disk.
- ***Section 10*** shows how to redirect the standard input and output so DOS can use input from a file or device other than the keyboard and send output to a file or device other than the screen.
- ***Section 11*** shows how to print files and screen displays.
- ***Section 12*** describes the DOS command line and how to edit it.
- ***Section 13*** is a list of some of the important things you should know when using your Hyperion.

Part I

Section 2

FILE MANAGEMENT

Section 2

FILE MANAGEMENT

2.1 FILES

All information is stored on the diskettes in the form of files. A file is any named collection of related information stored on a diskette.

The files on a diskette can be any size, and are automatically expanded (or shrunk) as you enter information into, or remove information from them. The only limitation, of course, is that the total size of all files on a diskette cannot exceed the storage capacity of the diskette. To use a filing cabinet analogy, individual folders can be as small or as large as desired, as long as the total space in a drawer is not exceeded.

Hidden Files

DOS diskettes often contain "hidden files", whose directory records cannot be displayed by the user. Hidden files cannot be deleted by the DEL or ERASE commands. They can only be erased by reformatting the disk on which they are located.

For more information on hidden files, see the descriptions of the CHKDSK and SYS commands in Part II of this manual.

Files

2.2 FILE NAMING

Every file on a DOS diskette must have a name. The filename itself is between 1 and 8 characters long. There is also (optionally) a filename extension of between 1 and 3 characters. The uniqueness of a file's identification (also called *specification*) requires only that one or more of the 11 characters used be different from any other 11-character file specification. Thus two files might have the same filename, but different filename extensions.

Filenames

The filename is any collection of up to 8 characters you select to name your file. Only the following characters **A-Z 0-9 $ & # @ ! % ' ( ) - { } _ ~ '** can be used for filenames. Any other characters or blank spaces are invalid. Lower case letters are accepted, but the system will convert them to upper case: entering a lower case letter is the same as entering an upper case letter.

Filename Extensions

The filename extension is an optional extension of 1 to 3 characters, preceded by a period, directly following the filename. It is used to segregate particular types of files into recognizable groups.

Through intelligent use of the filename extensions, you will minimize confusion about your files. You should define your own firm standards for filename extension use, and use them on every diskette. This procedure has several benefits, particularly in "wildcarded" filename searches, when searching for files with similar characteristics.

For example, if you wanted to organize files by month, you could use extensions such as .JAN, .FEB, etc. As a result, all files from January could be easily identified by the .JAN extension (e.g. MYFILE.JAN, ACCOUNTS.JAN, MEMOS.JAN).

Filespec (File Specification) Examples

The filename + extension = the *filespec*. The filename is separated from the extension by a period. All characters in the filespec must be consecutive, i.e., there can be no spaces in the filespec. Some examples of files are:

Text Files: LETTER.TXT
RESUME.TXT
AUG15.TXT

Data File: AUG15.DTA

When displaying filespecs on the Hyperion screen, the system will display the filename and extension separated by a space, rather than a period.

The choice of extension is up to you. You do not need to label text files as ".TXT" or data files as ".DTA". However, if a file already has an extension, you must use that extension as well as the filename to identify the file.

Reserved Filenames and Reserved Filename Extensions

Normally, DOS attaches no importance or special significance to filenames and their extensions. However, there are exceptions. DOS has reserved certain filenames for its own use. The reserved filenames are **CON, AUX, COM1:, COM2:, LPT1:, LPT2:, LPT3:, LPT,** and **NUL:.** If you attempt to assign one of these names to a new file, DOS will react in an unexpected way.

Similarly, DOS makes assumptions about certain filename extensions. Filename.COM is assumed to be a command. Filename.EXE is assumed by DOS to be an executable program. Filename.BAT is assumed to be a DOS batch file. Batch files are described in Part V of this manual. These filename extensions (.BAT, .COM, .CON, .EXE and .LPT) should not be used except as expected by DOS.

Part I

Section 3

TREE-STRUCTURED DIRECTORIES

Trees

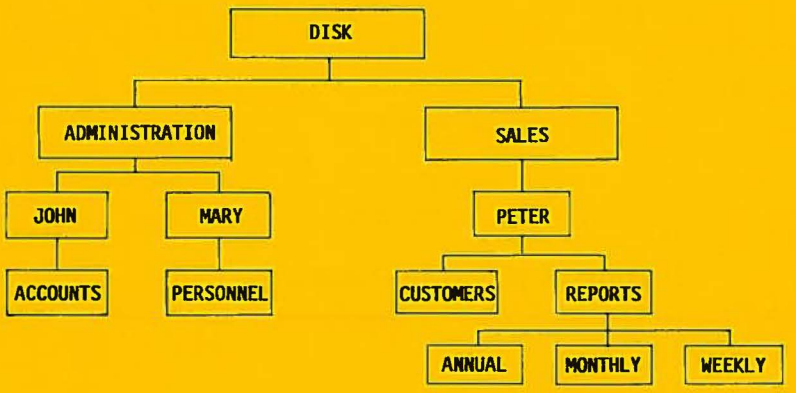


Fig. I-1 – Directories in a tree structure. Each name represents a directory of files.

Section 3

TREE-STRUCTURED DIRECTORIES

3.1 INTRODUCTION TO TREE-STRUCTURED DIRECTORIES

All previous versions of DOS provided only one directory on each disk. While this was sufficient for systems using diskettes, it provides problems for users storing files on hard disks. As more and more files are stored, it becomes difficult to keep track of them. This is made worse if there are several users with files on the same hard disk.

Having a large number of files in one directory is not only troublesome for users, but also inefficient for DOS, since it has to search through a large number of filenames when looking for a file.

This new release of DOS (version 2.11) provides the capacity to organize your files into groups that are convenient and make sense to you. These groups, called directories, are linked together in a hierarchy for logical and convenient access. Since this hierarchy resembles the branches on a tree, it is said to be "tree-structured".

For example, suppose your company has two departments, Administration and Sales, that share the use of a Hyperion and a hard disk.

To organize your files with DOS version 2.11, you can create a tree-structured hierarchy of directories on the Hyperion hard disk similar to that shown in Fig I-1.

When the directory structure is set up, Peter's reports can be grouped into three directories (ANNUAL, MONTHLY and WEEKLY), all contained in the REPORTS directory. As well his customer-oriented files can be put in the CUSTOMERS directory These reports are separated from all other files on the disk. In the same way, all the other files can be organized into directories, as you wish.

This means that if Peter is working on his customer files, DOS considers only the CUSTOMERS directory. All the files contained in other directories are normally ignored. If Peter wants to access a file not in the CUSTOMERS directory, he must specify a path to it (see Section 3.3, "Specifying the Path to a File").

When you format a disk, a single directory is created on it, just as with earlier versions of DOS. This directory is called the system directory or *root directory*. This directory is at the "top" of the directory structure (the DISK directory in Fig. I-1, for example).

A root directory can contain up to 112 files on a double-sided diskette, and a maximum of 64 files on single-sided diskettes. A 5 megabyte partition on a hard disk can hold up to 512 files in the root directory, and a 10 megabyte partition can hold up to 1,024 files.

In addition to file names, the root directory can contain the names of other directories, called *subdirectories*. Subdirectories, in turn, can contain the names of more files and subdirectories. Since subdirectories are treated like files by DOS (unlike the root directory), they have no restriction in the number of files and directories they may contain. They can have as many files and subdirectories as you want, limited only by the storage capacity of your disk.

The names you give your subdirectories must use the same format as names given to files: a directory name of up to eight characters, followed by an optional extension consisting of a period and up to three more letters (e.g. SALESDIR.REP). Any character valid when naming files is also permitted for directory names.

Any directory can contain the name of a file or directory already found in another directory. For example, in Fig. I-1, John or Mary could create a directory REPORTS even though Peter already has used this name for a directory. This is because DOS can identify each directory called REPORTS separately, depending on the name of the directory (JOHN, MARY, etc.) containing each REPORTS directory.

3.2 THE CURRENT DIRECTORY

In the same way that DOS keeps track of a "default" drive, it also remembers a "default" directory for each drive on your Hyperion. Defaults are the values DOS assumes when no specific value is given. The default directory is called the *current directory*. When you enter a filename without specifying a directory, DOS assumes you mean the current directory and searches there for the file.

You can change the current directory by using the **CHDIR** (change directory) command. To do so, press **F6** on the DOS soft key line or enter the command **CD** or **CHDIR**. If you use CHDIR without any parameters or only with the drive specification, DOS displays the current directory.

When you start your system, DOS automatically uses the root directory as the current directory on each drive until you enter a CHDIR (change directory command).

3.3 SPECIFYING THE PATH TO A FILE

When you want to create or access a file, you must provide DOS with three pieces or information: the drive specification, the directory name and the file name.

If the file you want is not in the current directory, you must provide DOS with a *path* made up of directory names to lead it to that file. The path can be either the path of names leading from the current directory or from the root directory.

The path format consists of a series of directory names separated by backslashes (\). If a filename is included, it also must be separated from the last directory name by a backslash.

If a path begins with a backslash, DOS begins its search from the root directory; otherwise, the path begins at the current directory.

Trees

3.4 DIRECTORY COMMANDS

To help you set up, use, and maintain your directories, DOS version 2.11 provides five directory commands. The commands, and their associated parameters, are described fully in Part II, Section 2, "DOS 2.11 Commands". They are:

CHDIR – Change the Current Directory

The CHDIR command is used to change the current directory on a drive, or to display the current directory path of a drive.

MKDIR – Create a Sub-Directory

The MKDIR command is used to create a new subdirectory on the specified or default drive.

PATH – Search Directories for Commands

The PATH command is used to tell DOS to search specified directories and drives for commands or files not found on the current directory.

RMDIR – Remove a Subdirectory

The RMDIR command is used to remove (delete) a subdirectory on the specified or default drive. You must remove all files from the directory except the "." and ".." entries before you can delete a directory.

TREE – Display all Directories

The TREE command is used to display all of the directory paths on the specified drive. With the /F parameter, the files in each subdirectory are also listed.

Part I

Section 4

LOADING AND STARTING DOS

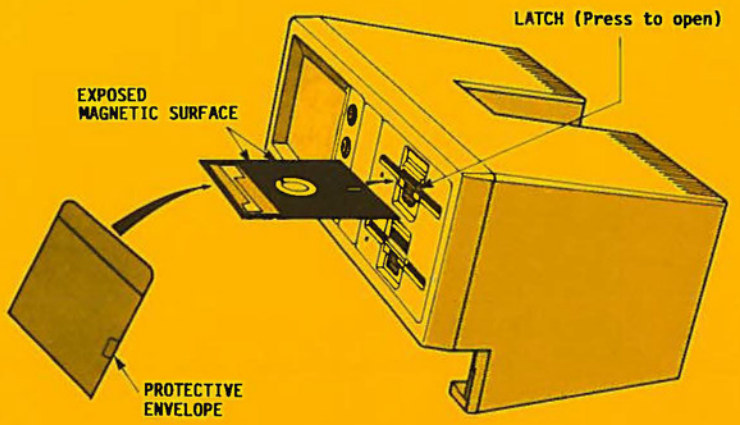


Fig. I-2 – Inserting a diskette into drive A. Note that each diskette drive is composed of an upper and a lower half. Pressing down on the upper half closes the drive. Pushing in on the lower half snaps the drive door open.

Section 4

LOADING AND STARTING DOS

4.1 INSERTING THE DOS DISKETTE

The master DOS diskette can be read, but cannot be modified in any way, as it is "write-protected". Master diskettes are not meant to be used regularly, but should be copied onto a second diskette, and the second diskette used. The master should be stored in a safe place and used to produce more copies as needed.

To insert the DOS diskette (copied from the master DOS diskette), or any diskette:

STEP

1) Remove the diskette from its protective envelope. DO NOT TOUCH THE EXPOSED MAGNETIC SURFACE.

2) Insert the diskette into the diskette drive slot (drive A) on the front of the Hyperion, in the following way:
   - open the diskette drive latch by pressing on the lower half of the latch,
   - hold the diskette with the labels on top and towards you, and insert the diskette all the way in, until it catches,
   - close the disk drive door by pressing down on the upper half of the drive latch until it clicks shut.

To remove the DOS diskette, or any diskette, from the drive:

STEP

1) Press on the lower half of the drive door latch until the top and bottom halves of the latch snap open and the diskette pops out.

2) Pull the diskette out of its drive and slide the diskette back into its protective envelope.

4.2 TURNING THE HYPERION ON

To turn the Hyperion on, press the amber power button on the front center of the unit until the button lights up.

After going through a 30-second self-test, during which the machine seems to be doing nothing, the system reads the diskette in drive A. If the DOS diskette is already in drive A when the Hyperion is turned on, the system automatically loads DOS into its internal memory.

System Startup Messages

If there is no diskette in drive A, the system looks for one in drive B. After both drives have been searched unsuccessfully several times, the message **DISK FAULT** is displayed. The system must be restarted by putting the DOS diskette into drive A and pressing the power button off and then on.

If the Diskette is Not the Right One

If there is a diskette, but without the needed information, the system displays the following messages:

Non-System disk or disk error
Replace and strike any key when ready

You have inserted the wrong diskette, or have put it in incorrectly. Reinsert the DOS diskette and press any key on the keyboard. The system will look for the needed information again, and load DOS into memory. If the system does not respond, reboot (restart) the system by pressing the **Ctrl, Alt** and **Del** keys simultaneously (**Ctrl + Alt + Del**).

4.3 RELOADING DOS

It is not necessary to press the power button off and then on to reload DOS and restart (reboot) the system. Making sure that the DOS diskette is in drive A, and then pressing **Ctrl + Alt + Del** will also reload DOS.

Anything stored in memory is lost, and any procedure being executed by the system is stopped.

Pressing **Ctrl + Alt + Del** bypasses the system self-test. There is no 30-second wait period. The system immediately begins to load DOS from drive A into memory.

4.4 ENTERING THE DATE AND TIME

After DOS is loaded, the date the system is set to will be displayed and you will be prompted to input a new date if you wish:

Loading

```
A>DATE
Current date is Tue  5-22-1984
Enter new date: 01-10-84

A>TIME
Current time is 14:52:08.95
Enter new time: 12:14

A>
A>

LASTLN Disks Files MODE DIR/P 12:00 CHDIR PATH TREE XPLAIN HELP
```

Fig. I-3 – The Hyperion screen, after starting up with DOS and responding to the DATE and TIME commands.

```
A>DATE
Current date is Tue 1-01-80
Enter new date:_
```

The cursor will be waiting, just after the colon, for you to input a date in the format indicated. You can either enter a new date by pressing the appropriate keys (e.g. 07-27-84) and pressing the **Return** key or, if you do not wish to change the date, simply press the **Return** key.

To verify or change the date at any time other than system startup, see the DATE command description in Part II.

Once you have correctly entered the date, the system will prompt you to reset the clock.

```
A>TIME
Current time is 0:00:33.99
Enter new time:_
```

Again, you can either set a new time by entering the appropriate numbers in the correct format (e.g. 8:15) and pressing the **Return** key, or you can accept the time indicated by the system and simply press the **Return** key.

Note that you can set the time to the degree of accuracy you want: to the hour only; hour and minutes; or hour, minutes and seconds. (Hundredths of a second will be calculated and appear when you query the time, but you cannot enter them yourself.)

The system clock, at the bottom center of the screen, should now show the correct time (as shown in Fig. I-3).

To change the time at any time other than system startup, see the TIME command description in Part II.

Loading

```
DOS Soft Key Line
LASTLN  Disks  Files  MODE   DIR/P          CHDIR  PATH  TREE   XPLAIN  HELP

FILES Soft Key Line
  Dos   Disks  TYPE   DATE   DIR/P          COPY   PRINT ERASE  RENAME  HELP

DISKS Soft Key Line
  Dos  D-NAME  Files  DATE   DIR/P         D-COPY D-COMP FORMAT CHKDSK  HELP

PARAMETERS Soft Key Line
  Dos   PRN   \PATH\  !MORE  !SORT           A:     B:     C:     D:     Rtn
```

Fig. I-4 – The four soft key lines in DOS.

4.5 SYSTEM PROMPT

A *prompt* in general is an indication from DOS to indicate that it is ready to receive a command or instruction.

DOS has a number of prompts **A>**, **B>**, **C>**, **D>**, etc. displayed on the left of the screen. The letter identifies the disk drive currently being accessed and the greater-than sign (>), indicates that the system currently in use is DOS version 2.11.

The Drivespec

The letter identifying the drive currently being accessed is also called the *drivespec*. The drivespec is often used in DOS command lines as a parameter, to direct the attention of a command to a particular drive.

4.6 THE SOFT KEY LINE

If the MODE settings (see the description of MODE in Part II) for Hyperion's soft key display are turned on, a row of 10 highlighted boxes across the bottom of the screen appears. This row is called the *soft key line*.

Each highlighted box on this line contains a label describing the command or feature that can be accessed by striking the key. The soft keys are the ten upper keys on the Hyperion keyboard, F1 through F10.

DOS has four soft key lines, as shown in Fig. I-4. The first three contain DOS commands; the fourth is displayed during the execution of some DOS commands. As shown in the figure, each soft key line has an identifying name – the DOS soft key line, the FILES soft key line, the DISKS soft key line and the PARAMETERS soft key line. Labels totally in upper case letters denote DOS commands. Labels in upper and lower case letters enable you to access other soft key lines.

Notice that, in each case, the name of the soft key line currently displayed is *not* displayed as one of the labels.

Changing Soft Key Labels

The Softkey Editing menu from the MODE command allows you to change soft key labels and designate a function to any key, except **F10**, which is reserved for HELP. The 40 soft keys available have already been given default labels and functions when your Hyperion is delivered. You may change them if desired.

System Time and Keyboard Indicators

As you can see in Fig. I-4, the ten soft keys are separated into two groups of five. Within this space is displayed the system time, and possibly two indicators: These indicators are displayed when the **Caps Lock** key has been used to lock the alphabetic keyboard into upper case (↑), and when the Num Lock key has been used to lock the number keypad into number mode (#).

4.7 SYSTEM AIDS

DOS has three system aids available for the user: the HELP function, the EXPLAIN command, and system messages.

The HELP Key

The HELP key is available on the tenth key of the 3 soft key command lines. When you press HELP, the screen clears to display information describing the action of all of the soft key functions available on that soft key line. If the soft key line changes, the contents of the HELP screens change to match.

You can also find out information on other Hyperion features and DOS commands using the EXPLAIN command. It is described in Part II.

Ctrl + HELP

Pressing the **Ctrl** and **HELP** keys simultaneously displays all the soft key lines for DOS. This display is called the *soft key map* for the system.

Error Messages

If the system cannot respond to a command, it will display an error message on the screen. This message is an aid to help you proceed, detailing where possible the problem area. Some error messages, such as "Bad command or file name", are often just the result of typing errors.

Error messages vary according to the command entered, and the system (DOS, EDLIN, etc.) being used. The error messages you can expect to get are described in the appropriate sections of this DOS guide.

4.8 ACCESSING EDLIN AND OTHER PROGRAMS FROM DOS

Once you are in DOS and have the DOS soft key lines displayed, you are free to either use DOS for disk and file manipulation, or to access other programs that use DOS as their operating system.

These programs are stored as **.EXE** or **.COM** files and are accessed by entering the filename and pressing **Return**. The system then searches the appropriate diskette(s) for a **.EXE** or **.COM** file with that filename and executes the instructions in the file.

The programs that are made available on the DOS diskettes and which are described in this manual are: **EDLIN,** the single-line text editor; **LINK**, which links together separately produced program files; **DEBUG**, to enable the debugging of application programs; and **LIB**, to store subroutines for access by more than one application program.

The filenames can also, by using the MODE command, be made into soft key labels. In such cases, you do not need to type in the filename and press **Return**. Simply pressing the appropriate soft key will call up the corresponding program.

Part I

Section 5

USING THE KEYBOARD AND KEYS

Section 5

USING THE KEYBOARD AND KEYS

5.1 CHARACTERS THAT LOOK ALIKE

There are several characters on any keyboard or screen that look alike and can be mistaken for each other. The computer, however does not accept one in place of the other.

The Letter "O" and the Number "0"

The letter "O" looks a little squarer than the number "0". Also, on screen, the number "0" has a diagonal slash through it so you can tell the difference.

The Letter "L" and the Number "1"

On certain typewriters the lower case "L" is often used for the number "1". The computer does not accept this substitution. When typing a number, always use the number "1".

5.2 SPECIAL KEYS

The Hyperion keyboard is the same as a typewriter keyboard, except for the addition of special keys. These keys are used to enter commands and instructions.

The special keys and their usual functions are shown in Table I-A. If their functions change during a particular operation, the change is documented separately.

Keys

Table I-A
SPECIAL KEYBOARD KEYS (WHITE)

SPECIAL KEY	FUNCTION
Soft keys F1 to F10	Enters soft key line commands.
Esc	Cancels a command line before processing.
Ctrl,Alt,Break	Used in conjunction with other keys to modify their normal use.
Caps Lock	Pushing this key once locks the alphabetic keys into an upper case mode: any letter you press will be entered as upper case. Using the *Caps Lock* key does not effect the numbers or symbols entered. An upwards arrow (↑) is displayed in the bottom middle of the screen whenever the *Caps Lock* key has been used to lock the alphabetic keyboard into upper case. Pressing this key a second time disables the *Caps Lock*.
Num Lock	Locks the number pad either into numeric mode, or into a function mode to move the cursor about the screen. An octothorpe (#) is displayed in the bottom middle of the screen whenever the *Num Lock* key has been used to lock the number pad into numeric mode.
Rub Out	Backspaces over and deletes previous characters entered.
Return	Used to signal the system that an instruction (or command line) is ready to be processed.
Shift	Enables you to enter upper case letters and symbols, as well as being used together with the **Print** key (see Table I-C). If the keyboard is in the *Caps Lock* mode, pressing the *Shift* key enables you to enter lower case letters.

5.3 SPECIAL KEY COMBINATIONS

Certain special keys can be used in combination with others to modify their normal use.

In this manual we use the convention **Key 1 + Key 2** as a shorthand for "hold down key 1 while pressing key 2". For example, **Ctrl + Break** means "hold down the **Ctrl** key, then press the **Break** key."

Table I-B
SPECIAL KEY COMBINATIONS

KEY COMBINATION	FUNCTION
STOPPING A PROCESS:	
Ctrl* + *Num Lock	Stops system processing so that you can review output (usually a screen display). Pressing any key resumes the operation that was interrupted.
QUITTING A PROGRAM OR SYSTEM:	
Ctrl* + *Break	Completely stops the processing of a command or function (such as the printing or display of a long document), and redisplays the system prompt for the entry of a new command. In some cases, a **Ctrl + Break** does not immediately stop a command. This is normal. When the command does stop, the system prompt reappears.
Ctrl* + *Alt* + *Del	Aborts any current process and restarts the system. Anything in the machine memory or on the RAM disk is deleted, and the system files are reloaded into memory.

Keys

Part I

Section 6

DESCRIPTION OF THE HYPERION

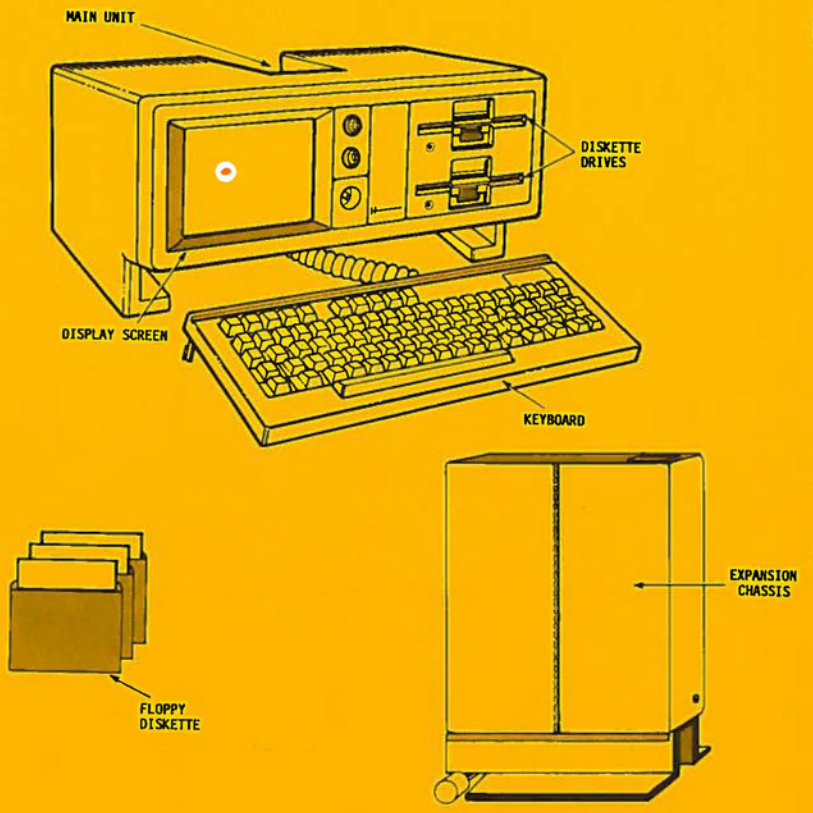


Fig. I-5 – The Hyperion hardware.

Section 6

DESCRIPTION OF THE HYPERION

6.1 THE COMPONENT PARTS

Your Hyperion system is composed of several pieces of equipment. These are shown in Fig. I-5.

a) ***Floppy Diskettes***. The Hyperion uses 5-1/4 inch floppy diskettes to store information.

b) ***The Main Unit***. The main unit accommodates the display screen, the controls, the internal devices, the diskette drives, and the rear panel connections.

c) ***The Keyboard***. The keyboard fits under the Hyperion when not in use and has a coiled extendable cord which connects it to the main unit.

d) ***The Hyperion Expansion Unit and Hard Disk***. The Hyperion expansion unit is an optional accessory, permitting the user access to a hard disk for information storage. The expansion unit also provides an additional port for serial printer interface. Expansion unit installation and operation are described in the *Hyperion EX User Guide*.

6.2 FLOPPY DISKETTES

The floppy diskette is a storage medium, with the capacity to hold up to 362,496 characters (bytes) of information, depending on how it's formatted.

If the diskette is formatted with a version of DOS earlier than version 2.11, it can have a maximum of 322,560 bytes of storage space. The same storage density can be formatted by using the /8 option with DOS version 2.11 (see the section in Part II on the FORMAT command). Earlier versions of DOS format 8 sectors per track, while DOS 2.11 formats 9 sectors per track..

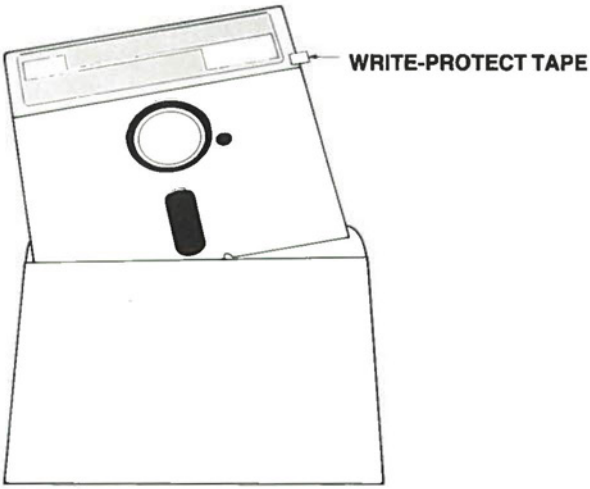


Fig. I-6 – The floppy diskette. Note the write-protect notch, covered with write-protect tape.

With DOS version 2.11, you can use diskettes formatted with earlier versions of DOS, but you cannot use diskettes formatted with nine sectors per track when running earlier versions of DOS. Therefore, to format diskettes for use with earlier DOS versions, you must use the /8 option when formatting with version 2.11.

If you use diskettes formatted with 9 sectors per track with earlier versions of DOS, you will corrupt the data contained on the diskettes.

Information is stored on diskettes as files, in permanent storage locations. The system requires you to name your files with a filename and optional filename extension for identification purposes. The filename acts as an address so the system can find the file when you wish to display data in a file or edit a file.

An analogy may be helpful in understanding the organization. The diskette is similar to a drawer in a filing cabinet. Each file on the diskette stores information as would a file in the drawer.

Master Diskettes

The diskettes supplied with this guide are labelled the Hyperion DOS diskette and Supplemental Programs diskette. These supplied diskettes are referred to as master diskettes. Standard diskettes have a notch on the upper right hand side which when uncovered enables you to write information to the diskette. To write-protect a diskette, you cover up this notch with write-protect tape. Our master diskettes are permanently write-protected because they do not have this notch in them. This protects the diskettes from accidental change or erasure.

The master DOS diskette contains the software (the programs) that you will require to create and manipulate data files. You should copy the system software and supplemental programs from the master diskettes onto other diskettes, and use the master diskettes as little as possible.

Protecting Diskettes From Erasure

It is important to understand the ability to write-protect diskettes. Only those Hyperion diskettes that have an uncovered write-protect notch on the upper right side (see Fig. I-6) can be erased or written on in any way. Diskettes may be write-protected by covering this notch with an adhesive tab.

Labelling a Diskette

Most diskettes have a permanent label in the upper left corner. We recommend that you apply temporary labels to identify your diskettes. These labels can be marked with a soft felt-tip pen with a name or other information to help the user with external diskette organization.

You should also enter "internal" labels for each diskette using the DISKNAME command. These labels are displayed whenever the DIR command is used, or the DISKNAME or VOL commands. For more information, see the descriptions of these commands in Part II.

Creating Backup Copies

Backup and diskette organization methodologies are a personal decision of every computer user. Typically, users make backup copies of important diskettes on a regular and frequent basis, to avoid potential data losses. Use the DISKCOPY command to make backup copies of entire diskettes, or the COPY command to make backup copies of specific files. Descriptions of these commands are provided in Part II of this guide.

Three of the most common methods of backing up diskettes are:

- **Method 1: Backup ↔ Current**

  The simplest backup methodology involves copying back and forth between two diskettes. The current diskette becomes the backup, and the backup becomes the new current diskette. While it is true that both are identical after the backup is made, it is wise to make the old backup become the new current. This ensures that each diskette is given a rest between backups, while the new current becomes the more heavily used version. The overall life of the pair is thus extended.

- **Method 2: Son ↔ Father ↔ Grandfather**

  Another commonly used scheme is referred to as the Grandfather-Father-Son methodology. This involves a rotation of three diskettes at each backup date. The current is copied onto the oldest (grandfather), and becomes the backup (father). The grandfather becomes the latest copy (son), and the old father becomes the new grandfather. This system is inherently safer than the two-diskette method, in that there is always one diskette sitting on the shelf, even while a backup is being made. It does, however, require more clerical diskette labelling effort.

- **Method 3: Snapshots**

  This technique can be used in conjunction with either of the above methodologies, or can in fact replace them. It consists of making a duplicate of a working diskette for filing. The duplicate becomes a "snapshot" of the working diskette's contents. The original working diskette continues to be used in normal operation, but should be manually marked as having been duplicated on this date.

Overall Recommendation on Backup Methods

Hyperion users should consider using the first proposal, for simplicity's sake, while taking occasional snapshots, for additional safety.

All of your diskettes should have backups. This includes diskettes that contain only programs, as well as those that contain data files.

The frequency of diskette backup is a factor of the level of change to the information in a diskette, your own perception of the reliability of your diskettes, and the perceived value of the information on a diskette.

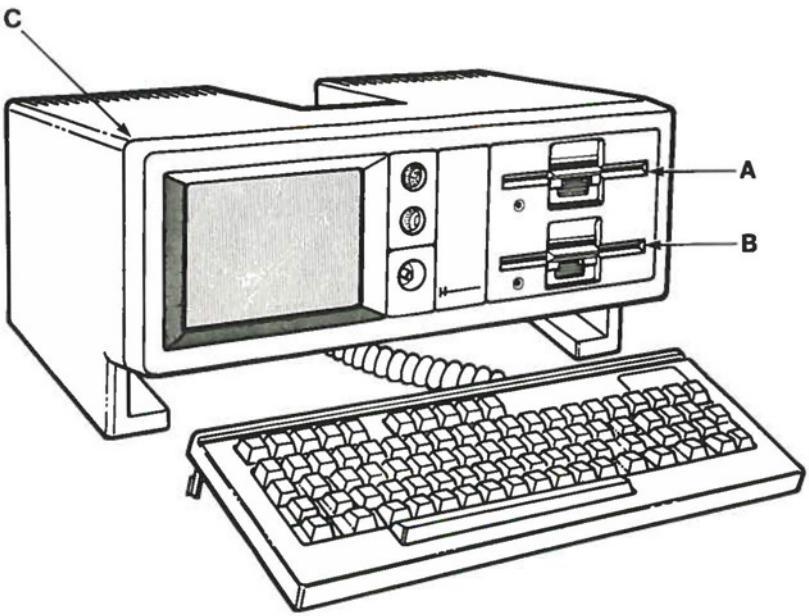


Fig. I-7 – The three drives.

6.3 DISK(ETTE) DRIVES

A diskette drive is the part of the Hyperion which holds the diskettes, reads the information stored in them, and writes new information onto them. The Hyperion main unit has three possible drives, two diskette drives labelled drive A and drive B, and an optional RAM disk which can be installed using the MODE command (see Figure I-7). The RAM disk is drive C if no hard disk is attached.

Drives A and B

In a two-drive Hyperion, drives A and B are located on the front of the Hyperion, at the right-hand side; in a single-drive unit, drive A only is located in this area. In this case, drive A acts as both drive A and drive B.

The RAM disk

A part of the internal random access memory (RAM) can be reserved (by the MODE command) as a third location in which to store files. To see how this is done, consult the description of the MODE command in Part II. This location is usually called "drive C", and is considered as a third diskette drive by DOS. DOS accesses drive C in the same way as it does drives A and B.

If a hard disk is installed, the RAM disk is called drive D, or E if there are two hard disks (i.e. the RAM disk is always the "last" drive in alphabetical order).

The advantage of having a RAM disk is that the storage and retrieval of information is much faster. However, anything stored on the RAM disk is erased if the Hyperion is powered off, or DOS is reloaded. Anything important stored on the RAM disk should be copied onto diskette or hard disk as soon as possible.

If a hard disk is attached to your Hyperion, you may find that storage and retrieval of files from the hard disk is almost as fast as from the RAM disk. You may thus decide to use the hard disk instead of the RAM disk, in order to benefit from the large storage capacity and decreased possibility of data loss. We do recommend, however, using the RAM disk for applications involving "piping" of output from one DOS command or activity to another.

Drives C and D

If an expansion chassis is attached to the Hyperion, up to two hard disks may be installed. DOS refers to these hard disk drives as drive C and drive D respectively.

Current Drive

The current drive is the designation for the drive that you are currently working on: either **A, B, C,** etc. This is the location from which DOS retrieves information, or to which DOS sends information.

When you start the Hyperion, the system automatically accesses drive A, which becomes your current drive. The prompt displayed on the screen will be **A>**, followed by the flashing cursor (_).

6.4 THE REAR PANEL CONNECTIONS

All of the jacks used to connect telephones, printers and other external devices to the Hyperion are located low on the back. The connectors are numbered and also identified by symbols.

Connector number 1 is used to attach another video display screen (monitor) to the Hyperion to provide a larger display, or viewing for onlookers. Connector 1 carries a composite video signal (a regular TV video signal). The cable which fits is a standard RCA Video Connector, available in most video and computer stores.

Connector number 2 is used to attach a telephone to the Hyperion when using the IN:TOUCH communications software package. IN:TOUCH, provides features such as automatic speed dialing, a search of your own input telephone directory (dialer file), and the ability to communicate with other computers, by sending the data across standard telephone lines.

Connector number 3 is used to hook the Hyperion into the public phone network. Connectors 2 and 3 are used together. By placing your Hyperion between the telephone and the network, you have placed its power and facilities where they may be directed over the public network.

Hyperion

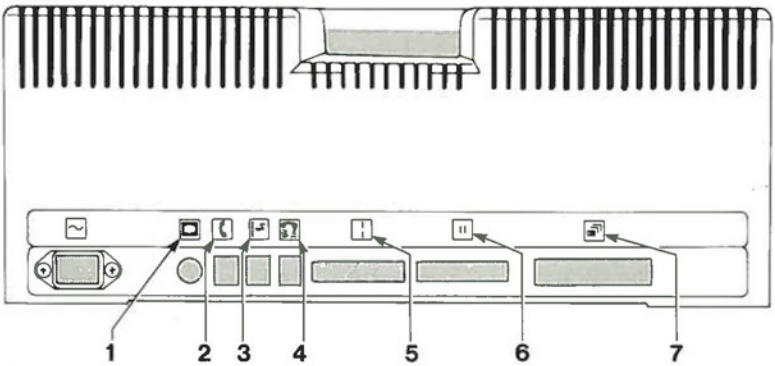


Fig. I-8 – The standard rear panel connectors on the Hyperion.

Connector number 4 connects the optional acoustic cups into the Hyperion. If you do not have standard telephone wall jacks on location, you will not be able to use the voice-call dialing capabilities of the Hyperion. To use your telephone line to communicate with other computer devices, you will have to connect the telephone to the acoustic cups that are available for the Hyperion.

Connector number 5 is used to connect printers, plotters, external communications devices, and other computer equipment that send and receive information in serial form.

Connector number 6 is a parallel interface jack used to connect a wide range of printers that receive information in parallel (character-by-character) form.

Connector number 7 is used to connect the Hyperion EX. The Hyperion EX is an expansion chassis designed to support seven special purpose cards, to provide a wide range of enhanced features: local area networks, special communications, memory expansion, and more. The EX also has an optional hard disk feature for extended storage capabilities, and an optional tape streamer for fast convenient backups.

Hyperion

Part I

Section 7

MASTER DOS DISKETTES

Section 7

MASTER DOS DISKETTES

All the software that you need with DOS, as well as other useful utilities and programs, are contained on two diskettes called the *Hyperion DOS* diskette and the *Supplemental Programs* diskette. When you receive the *Hyperion DOS (2.11) Guide*, these diskettes should be located in a protective plastic page at the back of the guide.

The two system diskettes you receive are referred to as **master** diskettes. These diskettes can be read, but cannot be modified in any way. They are "write-protected". Master diskettes should be copied onto a second diskette, and the second diskette used. The master should be stored in a safe place and used only to produce more copies. In this way you will be able to keep the software that you purchase in good running order.

7.1 PROGRAMS RESIDENT ON THE DOS DISKETTES

The master diskettes contain many files. They can be categorized into:

a) hidden files,
b) the AUTOEXEC.BAT file,
c) the COMMAND.COM file,
d) other .COM files,
e) files that contain programs accessible from DOS,
f) EXPLAIN (.EXP) files,
g) HELP (.HLP) files,
h) PRINT (.LPT) files.

Hidden Files

The master DOS diskette contains two hidden files. Their filenames are not displayed as part of the directory listing called up by the DIR command. Hidden files cannot be deleted by the DEL or ERASE commands. They can only be erased by reformatting the disk on which they are located.

The two hidden files are IO.SYS and MSDOS.SYS. They control the transfer of information between memory and diskette, and also have the ability to scan a diskette and list the filenames stored on it.

IO.SYS and MSDOS.SYS each have their own version numbers. Whenever DOS is loaded into the Hyperion, they are displayed briefly before the Hyperion star logo is displayed. These version numbers can be important when trying to access application programs since certain programs need a given version of IO.SYS or MSDOS.SYS to be able to run.

The AUTOEXEC.BAT File

Whenever DOS is loaded into the Hyperion, DOS automatically executes the instructions stored in the file called AUTOEXEC.BAT. Initially, this file contains DOS commands that configure the system using the MODE command and prompt you for the DATE and TIME.

You can modify the contents of the AUTOEXEC.BAT file at any time to include, or omit, instructions and control their execution. AUTOEXEC.BAT is a batch file. Batch files, and how to create and edit them, are described in Part V.

The COMMAND.COM File

The COMMAND.COM file contains the command interpreter for most DOS commands. Whenever you enter a DOS command, the system uses the routines stored in COMMAND.COM to execute the DOS command.

The COMMAND.COM file performs four main functions:

1) It handles device interrupts, including:

   - interpreting the Ctrl + Break command,
   - critical errors such as disk error or divide by zero,
   - performing end-of-program housekeeping.

2) It handles batch files, including searching for the AUTOEXEC.BAT file to be executed when DOS is first loaded into the Hyperion.

3) It controls the execution of most DOS commands.

4) It loads and executes other programs stored in files with filename extensions of .COM or .EXE.

Other .COM Files

Not all DOS command instructions are stored in COMMAND.COM. Some, like DISKCOPY, FORMAT, etc. are stored in their own .COM files.

Files That Contain Programs Accessible from DOS

The following files contain programs that you can access from DOS:

- EDLIN.COM,
- LINK.EXE,
- DEBUG.COM,
- LIB.EXE,
- BASICA.COM,
- BASIC.COM,
- BASICA.EXE.

EDLIN is the single-line text editor described in Part III. LINK, DEBUG, and LIB are described in Part IV. The BASIC files contain the programming language BASIC. BASIC is described in its own manual, the *Hyperion BASIC Guide*.

EXPLAIN and HELP Files

The Supplemental Programs diskette contains a number of files with the filename extension .EXP. DOS uses all files with the extension .EXP as "explain" files. Each contains a screen display that explains the purpose and use of a DOS command or feature. The appropriate explain file is called up by entering the EXPLAIN command and parameter.

You can create your own explain files to provide on-line information for other users, for example to describe batch files or programs you have created. Enter the information in a file using a text editor, and give the file an .EXP extension when naming it.

The HELP key calls up the appropriate explain file for each of the soft key lines in DOS. Special help files (MODE.HLP and FDISK.HLP) are accessed by the HELP key for the menus displayed by the FDISK and MODE commands.

PRINT Filters

Files with the filename extension of .LPT are print filters. Print filters enable you to use a printer with the Hyperion. The master DOS diskette comes equipped with one print filter: EPSON.LPT, for the Epson printer. Other print filters are available.

How the system selects which print filter to use is determined by the MODE command, described in Part II.

Part I

Section 8

CONFIGURE YOUR SYSTEM FOR EXTERNAL DEVICES AND SPECIAL DISPLAYS

Section 8

CONFIGURE YOUR SYSTEM FOR EXTERNAL DEVICES AND SPECIAL DISPLAYS

The term "configuring your system" refers to a number of system attributes which you can determine, including the way the screen displays information; the size and existence of the RAM disk; and the ability of the Hyperion to communicate with external devices such as printers or local area networks.

Configuring the Hyperion is done by using the MODE command, which calls up the MODE program. The MODE command is described in detail in Part II.

8.1 INTRODUCTION TO THE MODE PROGRAM

MODE is a utility program, used to modify certain operating system settings. MODE performs the following functions:

1) Sets screen options such as column width and soft key displays.

2) Directs printer output to a specified port and defines character interpretation.

3) Sets printer parameters.

4) Installs RAM disk and controls its size.

5) Sets communications port parameters.

6) Sets soft key labels and translations.

7) Enter values for a number of miscellaneous settings.

8) Displays all current MODE settings (MODE SHOW).

9) Updates settings to the ones specified in the last session.

10) Saves all current operating system (DOS) settings to the diskette.

8.2 THE AUTOEXEC.BAT FILE

As the system boots, the programs resident in DOS are loaded. As well, the AUTOEXEC.BAT file executes and runs the MODE program, setting the Hyperion system configuration and loading any optional commands and programs. This AUTOEXEC.BAT file must exist to run the MODE program automatically. However, a diskette that requires no special programs can omit running this MODE program, and will be identical to an IBM DOS diskette.

8.3 SAVING MODE CHANGES

All changes made to the MODE configuration should be saved to a DOS diskette. The new MODE settings take effect whenever the DOS on that diskette is loaded into the system.

If the MODE settings are not saved to a DOS diskette, they remain effective only for the current session and are erased when the Hyperion is powered off or restarted.

Part I

Section 9

SETTING UP YOUR HARD DISK

Section 9

SETTING UP YOUR HARD DISK

9.1 INTRODUCTION

If you have a hard disk attached to your Hyperion, you must set it up and format it before you can use it. If you try to use it before it is set up, DOS will access the RAM disk (if set up) instead, since it would consider the RAM disk as drive C.

If a hard disk is installed, DOS refers to it as drive C. A second hard disk would be referred to as drive D.

Each hard disk drive can be divided into separate areas called partitions, one for each operating system you are using. Each partition is set up using the special hard disk setup program provided by the respective operating system. There can be up to four different partitions of varying sizes, depending on your needs.

To set up or modify the hard disks for use with DOS, enter the **FDISK** command. This command calls up a menu on the screen (see Fig. I-9).

Cursor movement is controlled in the same way for this menu as for the menus called up by the MODE command. For more information on block movements on menu screens, see Table II-A in the description of the MODE command in Part II.

There are four basic options open to you on the Main FDISK menu shown in Fig. I-9. You can:

- select which disk you wish to set up, if you have two hard disks,
- create the DOS partition (and specify size and location),
- delete the DOS partition,
- set the bootable DOS partition.

```
Hyperion Hard  Disk Support

Hard  Disk No.  1            Partition Status
Partition  Bootable?  Type     Start    End     Size

    1         Yes     DOS      0        304     305


Main:

Set Bootable Partition to  :  1

DOS Partition              : (Nochange) Create  Delete

Select Hard Disk           : (1)

Cancel  Save                    15:36      ↑↑  ↓↓  ←←  →→  HELP
```

Fig. I-9 – The Main FDISK menu screen.

9.2 The Main FDISK Menu Screen

When you enter the the FDISK command, the menu shown in Fig. I-9 appears on your screen.

Above the menu selections, DOS displays information on the current status of the hard disk partitions.

DOS measures the size of your hard disk in cylinders. Most 10-megabyte hard disks have 305 cylinders, with each cylinder containing 32,768 characters (bytes) of data.

The partition number is shown in the **Partition** column.

If a partition is *bootable* (i.e., you can start, or *boot* your system from this partition), **Yes** appears in the **Bootable?** column. If a partition is not bootable, **No** appears in this column.

The **Type** column shows which partition, if any, is a DOS partition.

The **Start** and **End** columns give the cylinder numbers of the start and end points. The size of the partition, again in cylinders, is given in the **Size** column.

Underneath this table, there are 3 menu selections: **Set Bootable Partition to:**, **DOS Partition:**, and **Select Hard Disk**.

To save any changes you make to these menu selections, press the **F2** soft key (Save). This returns you to DOS. To cancel any selections and return to DOS, press **F1** (Cancel).

Setting the Bootable Partition

The bootable partition is the partition whose operating system is run by your Hyperion when you restart the system. If the DOS partition is bootable, then the Hyperion will run DOS when started. If the partition assigned to some other operating system is bootable, it will be run instead.

The entry under the **Bootable?** heading in the table above the menu selections gives the status of each partition. Only one partition can be bootable at a time. To set the bootable partition, move the cursor to the value to the right of the **Set Bootable Partition to:** menu selection. The number shown there by DOS will be that of the currently bootable partition. After you move the cursor, enter the number of the partition you want to make bootable. The partition numbers are found under the **Partition** heading in the table above the menu options.

Creating/Deleting the DOS Partition

The **DOS Partition:** menu selection lets you Create or Delete the DOS Partition, or leave it unchanged (**Nochange**). Partitions are deleted on this menu screen (the Main FDISK menu), while creating a DOS Partition calls up the FDISK Create menu (Fig. I-10).

To Delete the DOS Partition

This procedure destroys all information stored in the partition you are deleting. Before you delete the partition, make sure you have backed up all files containing valuable information that reside in the partition.

To delete the DOS partition, move the cursor over the **Delete** menu selection and press **Return**. The following prompt appears:

```
You are about to DESTROY all files
contained in the DOS partition!!
Are you Sure? (Y/N)...
```

Type in **Y** to delete the partition. If you type in **N** or press any other key, the DOS partition (and any files it might contain) is retained.

To Create the DOS Partition

If you want to run operating systems other than DOS on your Hyperion and its hard disk, you must divide the hard disk into partitions. You need one partition for each operating system.

These partitions can be any size and in any order on the hard disk. You cannot transfer data directly from one operating system's partition to another. If you try to access the hard disk with an operating system that hasn't been assigned a partition, you should get an error message.

The FDISK menu screens are used to create or delete the DOS partition. To set up partitions for other operating systems, use the appropriate formatting programs.

To create a hard disks partition for DOS, simply move the cursor to the **Create** menu selection and press **Return**. This will bring up another menu to allow you to specify the size and location of the new DOS partition (see Fig. I-10). How to use this menu to create a DOS partition is explained on Page I-49.

Selecting the Hard Disk

If your system has more than one hard disk, you can set up a second hard disk by moving the cursor to the desired disk number and pressing **Return**. Any changes you make on this screen or on the FDISK Create menu screen then apply to that disk.

```
Hyperion Hard  Disk Support

Hard  Disk No.  1            Partition Status
Partition  Bootable?  Type     Start    End     Size



Create:
Total Hard  disk space            :  305 Cylinders
Maximum available space           :  305 Cylinders
Starting at cylinder              :  0

DOS Partition Size                :  305

DOS Partition Starting Cylinder   : (  0)



Cancel  Save                      15:41      ↑↑  ↓↓  ←←  →→  HELP
```

Fig. I-10 – The FDISK Create menu screen.

9.3 The FDISK Create Menu

When you select the **Create** option from the **DOS Partition:** menu selection, the menu shown in Fig. I-10 appears on your screen.

Above the menu selections, DOS displays information on the current status of the hard disk partitions. This table is identical to that found on the FDISK Main menu, and is described above.

Underneath this table, there are 3 lines of disk space information, providing the total hard disk space, maximum available space, and the cylinder at which this largest available space begins (**Starting at cylinder**).

The values shown here cannot be changed on this menu screen. The **Maximum available space:** and **Starting at cylinder:** values can only be changed by changing the number or size of the partitions.

There are also two menu selections you can modify to meet your hard disk needs. These are the **DOS Partition Size:**, and **DOS Partition Starting Cylinder:**.

The values for the **DOS Partition Starting Cylinder** and **DOS Partition Size** menu selections are automatically assumed by DOS to give you the largest DOS partition possible on the hard disk. To change either of the values, move the cursor to the appropriate field and enter a new value.

Once you have specified the size of the partition and the starting cylinder, press the **F2** soft key (Save) to save your selections and return to the main FDISK menu. Press **F1** (Cancel) to cancel any selections you have made and return to the main FDISK menu.

9.4 Formatting the Hard Disk

Once you have created the DOS partition, you must format it to make it usable, just as you format floppy diskettes. To format the hard disk, simply type in the command **FORMAT/S** followed by the drivespec of the hard disk (**C:** for the first hard disk, **D:** for the second, if present).

The **/S** parameter can be omitted if you do not want to boot the system from the DOS partition. You may also use the **/V** parameter with the FORMAT command if you want to give the hard disk a volume name (diskname).

9.5 Example

For this example, consider the case where there are to be two operating systems sharing the hard disk. To set up the DOS partition you would:

STEP

1) Type in the command **FDISK** and press **Return**.

2) Move the cursor to the **Create** option of the DOS Partition: menu selection, and press **Return.**

...continued

STEP *(cont)*

The FDISK Create menu appears. If no other operating system is using the hard disk, the table in the upper half of the screen should be blank. The Total Hard disk space and Maximum available space entries should read 305 Cylinders, and the Starting at cylinder should read 0.

The DOS Partition Size should read 305 cylinders and the DOS Partition Starting Cylinder should read 0. For this example, consider a DOS partition of 200 cylinders starting at 0.

3) Move the cursor to the DOS Partition Size menu selection and type in **200** over the value shown (305).

Since the value shown for the DOS Partition Starting Cylinder menu is **0**, you do not have to move the cursor there to enter a new value.

4) Press the **F2** (Save) soft key to return to the main FDISK menu.

The table in the upper half of the main FDISK menu screen should read as follows:

Hard Disk Partition	No. 1 Bootable?	Type	Partition Status Start	End	Size
1	No	DOS	0	199	200

...continued

Hard Disk

STEP *(cont)*

5) Move the cursor to the Set Bootable Partition menu selection, type in **1** over the value shown (0), and press **Return**.

   The value for the Bootable? entry on the table in the upper half of the main FDISK menu screen should now be Yes for Partition number 1.

6) Press the **F2** (Save) soft key.

   Since a new DOS partition has been created, the operating system must be reconfigured to recognize the new drive. To do so, DOS automatically restarts (reboots) the system, and prompts:

   Insert DOS diskette in Drive A:
   Press any key when ready ...

7) Make sure the DOS diskette is in drive A and press any character key. The system then goes through its startup procedure.

8) Type in the command **FORMAT/V/S C:** and press **Return**. DOS formats the hard disk partition and prompts you for a volume label. Enter a volume label if desired, or else press **Return**.

   Your hard disk partition is now formatted and ready to be used.

Part I

Section 10

REDIRECTION OF STANDARD INPUT AND OUTPUT

Section 10

REDIRECTION OF STANDARD INPUT AND OUTPUT

10.1 INTRODUCTION TO STANDARD INPUT AND OUTPUT

There may be occasions when you wish to have a program running on your Hyperion receive its input from some other source than the keyboard. As well, there may be times you want to have a program's output sent to some destination other than the display screen.

DOS lets you do this with a feature called "*Redirection of Standard Input and Output.*" Normally the standard input device is the keyboard, and the standard output device is the display screen. However, the standard input and output can be redirected to or from files or other devices. The commands described below show how this can be done.

If a program does not use DOS function calls to perform standard input and output, then the redirection commands will not work. Examples of this occur when a program puts text or graphics directly into the video buffer.

10.2 REDIRECTION OF STANDARD OUTPUT

To redirect the output from a program to a file or device other than the display screen, use one of the following formats. Remember, the symbols > and >> are treated like commands by DOS.

To a New File: >[d:][path]file

This creates a file with the filename and extension specified by file, or empties the file if it already exists, to prepare it to receive data output by the program. Any output that would have gone to the display screen goes to this file instead.

To an Existing File: >>[d:][path]file

This opens the file with the filename and extension specified by file, or creates it if it doesn't exist, and adds any data output by the program to the end of the file.

10.3 REDIRECTION OF STANDARD INPUT

To have the input for a program taken from a file or device other than the display screen, use the following command format. Remember, the symbol < is treated like a command by DOS.

From an existing file: <[d:][path]file

This causes the standard input to be assigned to the file with the filename and extension specified by *file*. All input for the program will be taken from this file instead of the keyboard. When you use this command be sure all the input needed by the program is in the file. If a program tries to read information after reaching the end of file (EOF) mark, DOS cannot supply it with any input, and processing will stop. If this happens, you can return to DOS by pressing the **Ctrl + Break** combination.

10.4 PIPING OF STANDARD INPUT AND OUTPUT

If the output being redirected from one program is used as input for another, DOS can store it in a temporary file using a feature called *piping*. This feature is often used with filters, which are programs or commands that read data from a standard input device, modify the data, and write the results to a standard output device. The DOS commands that can "filter" data contained in files are FIND, MORE, and SORT.

The temporary files created by the piping feature are of the form:

%PIPEx.$$$

where *x* is a different number for each file, e.g. %PIPE1.$$$, %PIPE2.$$$, etc.

Since the output of one program becomes the input for another, the programs are said to be chained together. The names of the programs being chained are entered on the command line and are separated by the vertical bar sign (|). For example:

DIR | SORT>filename

would take the output from the DIR command, sort it, and send the results to the specified file.

Part I

Section 11

PRINTING

Section 11

PRINTING

11.1 CONNECTION TO A PRINTER

The Hyperion may be connected to a printer through one of two types of output ports. A parallel port is provided by connector number 6 on the rear of the main unit. A serial port is provided by connector number 5, also on the rear of the main unit.

11.2 PRINTING A SCREEN DISPLAY

If your Hyperion is connected to a parallel printer, then using the **Shift + Print** or **Ctrl + Print** key combinations will send, or stop sending, information from the Hyperion screen to the printer (refer to Table I-C).

If a printer isn't attached to your Hyperion when you press these keys, the system spends about 30 seconds "trying to print". During this time, the computer appears to be "dead" and won't accept any input from the keyboard.

11.3 PRINTING A FILE

The most convenient way to print files is to use the PRINT command. This command lets you *queue* or line up a number of files for printing while you perform other tasks on your Hyperion. The printing is said to be in the "background" while you work in the "foreground". For more information on how to use the PRINT command, see Part II.

You may also print files using a special form of the COPY command. The COPY command works in the foreground and ties up the Hyperion while "copying" files to the printer. However, it has the advantage that nothing else is done that could interrupt the system. For example, if you were printing a long file with the PRINT command, you would not want some activity being performed in the foreground to stop the system.

To print files with the COPY command, use the following format:

COPY [d:]file PRN

where *file* is the filename and extension, if any (eg. MYFILE01.RPT), and *d:* is the optional disk drive location, needed only if the file isn't located on the default drive.

Wildcards (* or ?) can also be inserted into the filename or extension to print out several files by entering only a single command line (eg. A:MY*.R??).

Table I-C
SPECIAL KEY COMBINATIONS USED FOR PRINTING

KEY COMBINATION	FUNCTION
Shift + Print	Causes the current contents of the screen to be sent to an external printer. If the printer is on, the contents of the screen are printed.
Ctrl + Print	Causes the printer to act as a typewriter. The printer prints out whatever is being generated or entered onto the screen. Pressing **Ctrl + Print** a second time suppresses this typewriter mode.

Printing

You can also create batch files (see Part V) to stack COPY commands (i.e. have them executed sequentially without user interaction).

The Print Filter and MODE Command

The parameter PRN calls up a print filter. This print filter translates certain Hyperion escape sequences and characters into instructions that a particular printer can understand. To use the Hyperion with your printer, you should:

1) have the appropriate print filter on your DOS diskette,

2) have used the MODE command to select that print filter. The changes made in MODE must be saved to the disk you use to start your system.

Part I

Section 12

THE DOS COMMAND LINE

Section 12

THE DOS COMMAND LINE

12.1 THE COMMAND LINE IN DOS

In DOS you enter instructions into the system by using the keyboard to prepare command lines. Each command line is entered after a system prompt, which indicates that the Hyperion is ready for instructions.

The command line consists of a *command word*, which directs the nature of the computer's efforts; and *parameters*, which define the scope of the command.

COMMAND-WORD [parameter] [parameter] [...]

A command line cannot exceed 128 characters in length.

The Command Word

The command word is the first word in the command line. It identifies the command. To type in a command word, either press the appropriate soft key (from **F1** to **F10**) or type in the command word.

Drivespec

The drivespec is short for "drive specification". It is the letter identifying the drive that the system is currently interested in. The Hyperion main unit has two drives, A and B. Consequently, there are two drivespecs, A: and B:.

A drivespec is used as the system prompt to remind you which drive is currently active.

When a RAM disk is set up on your system, DOS refers to it as drive C.

If you add a hard disk to your system, the hard disk becomes drive C, and DOS renames the RAM disk as drive D. The RAM disk becomes drive E if you add a second hard disk, and the extra hard disk becomes drive D.

The command line you enter usually applies only to the default drive, identified by the system prompt. For certain commands, however, you can enter a drivespec before a parameter. This tells the system to search a given drive for that parameter. If you want DOS to search other drives for a command, you must use the PATH command (described in Part II).

Parameters

You may follow certain command words with other words which may qualify how the system is to interpret the command. For example, following the command word EXPLAIN with another command word instructs the system to display information about one particular command, and not about all the commands. These qualifying words are known as "parameters".

Optional parameters are enclosed with square brackets ([,]) when listed in the command line.

When a series of parameters are each enclosed with braces ({,}), you may use only one of them in the command line. For example, {A}{B}{C} would mean that you could choose only one of A,B, or C.

Filespecs

Most parameters are used to identify diskette files. This is done by entering the filespec (file specification) as a parameter.

Filespecs are composed of a filename (1 to 8 characters long) and an optional filename extension (0 to 3 characters long). The filename and extension must be separated by a period, and may not contain spaces or the characters reserved by DOS for special purposes (see Page II-7).

Certain filename extensions have special meanings for DOS and should not be used in unauthorized cases. These extensions (.CON, .AUX, .COM, .LPT, .NUL, .EXE and .BAT) are described in Section 2, "File Management".

Wildcarding

When entering filespecs (filename + extension) as parameters to certain commands, you may not remember the exact filespec, or you may want to refer to several similar filespecs at one time. In that case, substituting a *wildcard* for a character, or sequence of characters, instructs the system to search for all the files which have the remaining characters the same as those entered but which may have different characters in place of the wildcard character. DOS allows two wildcard characters, the question mark and the asterisk.

A ? in a filename or extension is a single character wildcard: DOS will find any file that matches all other characters in the filespec, and that has *any character* in the character position where the ? is typed.

An * in a filename or extension is a multiple character wildcard: DOS will find any file that matches all other characters in the filespec, and that has *between zero and eight other characters* in the position where the * is typed. Characters after *, but before the end of the filename or filename extension are ignored.

EXAMPLES:

***.TXT** identifies all files that have the extension ".TXT".

AF*.T?? identifies all files that begin with the letters "AF", and that have extensions beginning with "T".

WARNING:

You should always be sure that you know which files will be effected when using wildcards. Because so many files can be affected by wildcarding, it is often impossible to reverse any unexpected consequences.

The best way to check is to do a directory listing (with the DIR command) of all files matching the wildcarded filename and extension.

Be extremely careful when using wildcards with the COPY, ERASE, and DEL commands.

A batch file may contain wildcards, but a wildcard cannot then be used to load and execute such a batch file.

Table I-D
KEYS USED TO EDIT THE COMMAND LINE IN DOS

KEY(S)	EDITING FUNCTIONS
Rub Out	Rubs out and backspaces over the character directly to the left of the cursor.
←	Moves the cursor to the left one position.
→	Moves the cursor to the right one position.
Home	Moves the cursor to the beginning of the command line.
End	Moves the cursor to the end of the command line.
Ins	Inserts a space at the cursor, shifting following characters right one position.
Del	Deletes character at cursor, shifting following characters left one position.
Ctrl + ←	Moves the cursor to previous start of word.
Ctrl + →	Moves the cursor to next start of word.
Esc	Cancels the command line. Places backslash at end of line (to indicate cancelled command) and redisplays a new system prompt in preparation for the entry of a new command line.
Return	Submits the entire command line in its presently displayed form. No command is processed in DOS until the Return key is pressed.

12.2 EDITING THE COMMAND LINE IN DOS

A DOS command is not accepted for processing until you have pressed the **Return** key. You can therefore change the characters on the command line until you are satisfied with them.

The Cursor

The flashing underline or rectangle you see on the screen is called a cursor. It identifies where the next character you enter (by pressing a key) is going to appear on the screen.

Editing Keys

Moving the cursor back and forth along the command line makes it possible for you to insert or change the characters in the command line. Certain keys on the Hyperion keyboard, listed in Table I-D, enable you to edit the command line.

Section 13

SUMMARY OF BASIC CONCEPTS

COMMANDS

Entering Commands

To enter a command for DOS to execute, you must type in the *command word* followed by as many **parameters** as you need. When this **command line** is correct, press **Return**.

Recalling Commands

Pressing the **LASTLN** soft key (**F1** on the DOS soft key line) brings back the previously entered command line. You can use these characters as part of the current entry, and edit them if necessary.

DISKETTES

Backing up Diskettes

To back up a diskette, use the **DISKCOPY** command, as described in Part II.

Labelling Diskettes

Use the /V (for VOL) parameter to label diskettes when formatting the diskettes. Use the DISKNAME command to label disks after they have been formatted. Either the VOL or DISKNAME command can be used to display the label. These commands are described in Part II.

Master Diskettes

Master diskettes contain the original copy of your software. You should make working copies of these diskettes (if they can be copied) and store the master diskettes in a safe place.

...continued

Summary of Basic Concepts (cont)

FILES

Backing up Files To back up a file, use the COPY command, as described in Part II.

Displaying the Contents of a File To display what is in a file, use the TYPE or MORE commands, as described in Part II.

Erasing a File To erase a file, use the ERASE or DEL commands, as described in Part II.

REDIRECTION OF STANDARD INPUT AND OUTPUT

You can send the output from a DOS command to a destination (including a file) other than the screen by using the < or << symbols. Input for DOS commands can be taken from a source other than the keyboard by using the > symbol. Output from one DOS command can be used as input for another (or "piped") by using the symbol. For details, see Section 10 of this part.

WILDCARDS

Using the asterisk (*) and question mark (?) in place of characters in a filename and filename extension enables you to apply the DOS command to more than one file at a time. You should be careful when using wildcards with the **ERASE**, **DEL** and **COPY** commands as you can easily erase or write over many more files than you intend.

Section 1

INTRODUCTION TO PART II

1.1 TO ENTER A COMMAND

You use the soft keys and/or the keyboard to enter instructions into the Hyperion. To enter an instruction, you:

1) Press a soft function key **F1** to **F10** on the keyboard;

   OR

   type in a command word.

2) Use the soft key line or keyboard to enter the desired parameters.

3) Press the **Return** key.

System Prompt

When entering a command into the computer, you are responding to a system *prompt*, which tells you that the system is ready and waiting for you to enter instructions.

System Aids

The Hyperion has two ways of helping you, with on-line HELP and EXPLAIN descriptions of features and commands, as well as by displaying error messages.

Recalling the Last Command Line

Pressing the **LASTLN** soft key **(F1)** on the DOS soft key line brings back the previously entered command line. It automatically brings back any sequence of characters that you have entered. You can then accept these characters as part of the current entry, and edit them as necessary. LASTLN is a very useful aid to command entry.

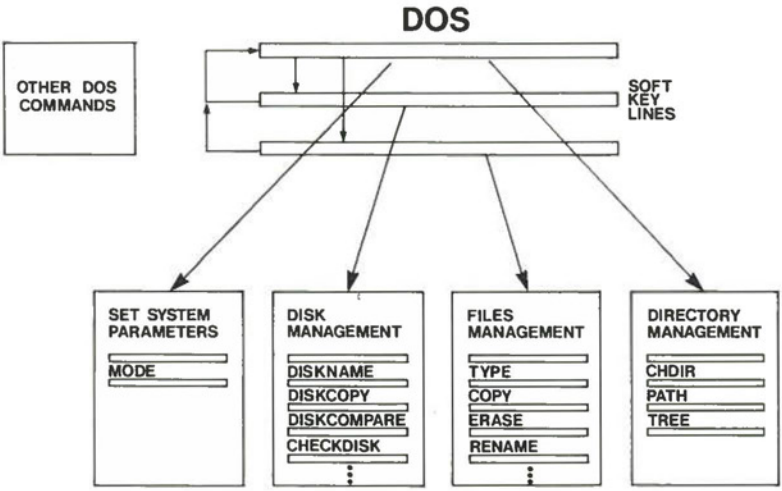


Fig. II-1 – Some of the most common DOS commands can be entered using the soft keys. Others must be typed in using the keyboard.

1.2 CATEGORIES OF DOS COMMANDS

The DOS commands that follow fall into five categories. Initially, some of the commands in each category are displayed on soft key lines. We say "initially" because you can, if you wish, reassign the DOS commands, and other commands, to any soft key. You use the MODE command to do this reassignment.

The 5 categories of DOS commands are:

a) The DOS commands, which are used for over-all system control, including maintenance of subdirectories.

Some of these are found on the DOS soft key line. They let you:

- configure the system (**MODE**)
- display files on diskette (**DIR**)
- change current directory (**CHDIR**)
- define directories to be searched (**PATH**)
- display all directory paths (**TREE**)
- display explanations (**EXPLAIN**)

There are also some system control commands that are not found on soft keys and must be typed in. They let you:

- clear display screen (**CLS**)
- set standard input/output console (**CTTY**)
- print graphics displays (**GRAPHICS**)
- create subdirectories (**MKDIR**)
- set system prompt (**PROMPT**)
- remove subdirectories (**RMDIR**)
- insert environment strings (**SET**)
- display or change system time (**TIME**)

One DOS command, **FDISK**, used to set up and maintain a hard disk, is discussed in Part I, Section 9.

b) The FILES commands which are used to control files on disks.

Some of these are found on the FILES soft key line. They let you:

- display files (**TYPE**)
- display/modify current date (**DATE**)
- list files on a disk (**DIR**)
- copy files (**COPY**)
- print files while doing other tasks (**PRINT**)
- erase files (**ERASE**)
- rename files (**RENAME**)

There are also some FILES commands that are not found on soft keys and must be typed in. They let you:

- back up files from hard disks (**BACKUP**)
- convert .EXE to .COM files (**EXE2BIN**)
- search files for text (**FIND**)
- recover files from bad diskettes (**RECOVER**)
- restore files to hard disk (**RESTORE**)
- verify data as it's copied (**VERIFY**)

Two other FILES commands are usually typed in, but can be entered as well from the PARAMETERS soft key line when following another command. They let you:

- display text a screenful at a time (**MORE**)
- sort text data (**SORT**)

c) The DISKS commands which involve the diskette as a whole.

Some of these are found on the DISKS soft key line. They let you:

- display name or rename a diskette (**DISKNAME**)
- display/modify current date (**DATE**)
- list files on a diskette (**DIR**)
- duplicate a diskette (**DISKCOPY**)
- compare two diskettes (**DISKCOMP**)
- prepare a new diskette for use (**FORMAT**)
- check diskette status (**CHKDSK**)

There is also one DISKS command that is not found on soft keys and must be typed in. It lets you:

- assign a different drive (**ASSIGN**)

(Notice that DIR appears as DIR/P on all 3 soft key lines and EXPLAIN is labelled as XPLAIN on the DOS soft key line. Also, DISKNAME, DISKCOPY,and DISKCOMP are abbreviated to D-NAME, D-COPY and D-COMP on the DISKS soft key line.)

The next 2 categories of DOS commands are unlike the others, in that they cannot be entered directly into the system. Batch commands must be in a batch file for DOS to use them, and configuration commands must be in the configuration file (CONFIG.SYS) in order to be used.

d) The BATCH commands (described in Part V) which are used to increase the flexibility of batch files, let you:

- display commands while executing (**ECHO**)
- use FOR loop in batches (**FOR**)
- transfer control to another line (**IF**)
- use additional parameters (**SHIFT**)
- halt processing to display message (**PAUSE**)
- display remark from file (**REM**)

e) The CONFIGURATION commands (described in Part V) which are used in the configuration file accessed by DOS when starting your Hyperion. They tell DOS to:

- check for control break (**BREAK**)
- set memory buffer size (**BUFFERS**)
- specify device driver file (**DEVICE**)
- specify number of open files (**FILES**)
- specify top-level command processor (**SHELL**)

1.3 VERSION 2.11 ENHANCEMENTS TO DOS

Hyperion DOS version 2.11 has several major enhancements as well as some minor differences from previous versions of DOS. For more information, see Appendix A.

There are a number of new commands in this version of DOS. They are:

– ASSIGN	**– BACKUP**	**– BREAK**
– CHDIR	**– CLS**	**– CTTY**
– GRAPHICS	**– LOCK**	**– MKDIR**
– PATH	**– PRINT**	**– PROMPT**
– RECOVER	**– RESTORE**	**– RMDIR**
– SET	**– TREE**	**– UNLOCK**
– VER	**– VERIFY**	**– VOL**

as well as batch commands (**ECHO, IF, FOR, SHIFT,** and **GOTO**) and configuration commands (**BREAK, BUFFERS, DEVICE, FILES, SHELL**). The batch and configuration commands are described in Part V. Another new command on this DOS version is **FDISK** used to set up a hard disk for use with DOS. It is described in Part 1, Section 9.

Some of the commands found in earlier versions of DOS have been enhanced as well. They are:

– CHKDSK	**– DIR**	**– DISKCOPY**
– DISKCOMP	**– ERASE**	**– FORMAT**

1.4 CHARACTERS RESERVED BY DOS

Some characters have a special meaning for DOS, and are used as part of the format of DOS commands. These characters must not be used in path names or filenames. Since DOS version 2.11 offers several enhancements over earlier versions of DOS, some characters that were permitted in filenames are now reserved by DOS. These are:

| < > \

If you have any files with names containing one or more of these characters, you will have to change the names to remove these newly reserved characters. To do this, you must use your old version of DOS. After you have changed the filenames, you may use the files with DOS version 2.11.

In addition to the reserved characters mentioned above, characters reserved in earlier versions of DOS are still reserved in DOS version 2.11. These are:

* ? / . , ; :

1.5 ORGANIZATION OF COMMAND DESCRIPTIONS IN THIS MANUAL

Section 2, which follows, is a detailed description of all DOS commands (except for FDISK and the batch and configuration commands), presented in alphabetical order.

Each command is described in the same way, under the headings: "Entering the Command", "Command Description", "Command Format", "Parameters", "Warnings", "User Interaction", "Error Messages", "See Also", and "Examples".

1.6 COMMAND FORMAT CONVENTIONS

Parameters shown in square brackets "[" and "]" are optional.

When parameters are enclosed in braces "{" and "}", you may choose only one of the members of the set. For example, {**A**} {**B**} {**C**} would mean that only one of A, B or C is to be chosen. If the three parameters were enclosed in square brackets, (e.g. [{**A**}{**B**}{**C**}]), then the parameters would be optional. You could enter any *one* of them, or none, as desired.

1.7 EXECUTING DOS COMMANDS

In order for DOS to execute any command, it must access the file containing the necessary instructions. If DOS cannot find the file with these instructions, it displays the following message:

Bad command or filename

The instructions for certain commands are contained in the COMMAND.COM file (see Part I, Section 7) but the instructions for most commands are in separate files. These files are:

ASSIGN.COM	BACKUP.COM	CHKDSK.COM
COMMAND.COM	DISKCOMP.COM	DISKCOPY.COM
DISKNAME.COM	EXE2BIN.EXE	FIND.EXE
FORMAT.COM	GRAPHICS.COM	LOCK.COM
MODE.EXE	MORE.COM	PRINT.COM
PRINT.COM	RECOVER.COM	RESTORE.COM
SORT.EXE	SYS.COM	TREE.COM

As you can see, the command word (ASSIGN, BACKUP, etc.) is used as the filename for each of the files. The file corresponding to the command must be available (i.e. located on one of the disk drives or RAM disk) for the command to be executed.

Part II

Section 2

DOS (2.11) COMMANDS

ASSIGN

Designate a Different Drive for Disk Operations

ENTERING THE COMMAND

STEP

1) Type the command word **ASSIGN.**

2) Enter the parameters, edit the command line if necessary, then press the **Return** key.

COMMAND DESCRIPTION

ASSIGN is used to tell DOS to route all requests from one drive to a second drive. All commands referring to the first drive will use the second drive instead.

If you enter ASSIGN without any parameters, all drive reassignments will be reset to the normal drive assignments.

COMMAND FORMAT

ASSIGN [d1 = d2[...]]

PARAMETERS

— If no parameters are entered, ASSIGN resets all drive reassignments to the normal drive assignments.

d1: The *drive* being reassigned. The *d2* drive will be considered as the *d1* drive by DOS.

ASSIGN
(cont)

d2: The *drive* to which all requests for *d1* are routed.

WARNINGS

ASSIGN is intended to help you with applications set up to perform their disk operations specifically on drives A and B. For example, a command such as:

```
ASSIGN A=C B=C
```

would mean that those applications would run on drive C instead of A and B, where drive C is a hard disk. Reassignments should only be used when necessary in such cases. Do not use it with the PRINT command when running normal DOS operations, since it can hide the true device type from commands and programs that require the actual drive information.

If you are writing any application programs, we strongly suggest you do not make specific drive assignments in your program, but leave it up to the user to specify the drives to be used.

If you use DISKCOPY, DISKCOMP or FORMAT commands, DOS ignores any drive reassignments you have made. We recommend that you cancel all drive assignments (enter ASSIGN without any parameters) when using these commands to avoid confusion.

ERROR MESSAGES

Possible error messages are:

- **Invalid drive in search path**
- **Bad command or file name**
- **Invalid parameters**
- **Incorrect DOS version**

BACKUP

Back up Hard Disk Files to Diskettes

ENTERING THE COMMAND

STEP

1) Enter the word **BACKUP**.

2) Enter the parameters, edit the command line, if necessary, and then press **Return**.

COMMAND DESCRIPTION

This command is used to back up files from a hard disk to diskettes so that files can be restored (with the RESTORE command) if they are inadvertently lost or erased from the hard disk.

COMMAND FORMAT

BACKUP [d1:][path][file] [d2:][/S][/M][/A][/D:mm-dd-yy]

PARAMETERS

d1: d2: The *drivespecs*. The first (*d1*) is the location where the file to be backed up resides. The second (*d2*) is the location of the diskette to which the file will be sent.

path The *path* defining the current directory. The *path* consists of a list of directory names, separated by backslashes (\), needed to tell DOS the location of the current or specified directory.

BACKUP
(cont)

file The *filename and extension*, if any. Wildcards may be used, and result in all files matching the filename being backed up.

/S This parameter causes all files in all associated *subdirectories* to be backed up in addition to the files in the specified directory.

/M This parameter specifies that only files created or modified since the last back-up should be backed up. Use */M* to avoid backing up files that never change. DOS can tell which files have been changed since it sets an indicator in each file's directory whenever DOS modifies the file.

/A This parameter indicates that backed up files should be added to the files on the backup diskette already in the specified drive. If */A* is not entered, you will be prompted to insert a diskette when the BACKUP program starts.

/D:mm-dd-yy This parameter is used to back up files changed on or after the specified *date*. The *date* must conform to one of the valid date formats as explained in the DATE command description.

BACKUP
(cont)

WARNING

The files on the backup diskettes cannot be used in normal processing. They should only be used with the RESTORE command to restore copies of the files to the hard disk.

MESSAGES

Possible messages include:

Warning! No files were found to back up

SEE ALSO

Files created on backup diskettes by the BACKUP command can only be used by the RESTORE command.

EXAMPLES

Example 1) – *Backing up all files created after a certain date:*

The command **BACKUP C:*.GL A:/D:01-01-84** backs up all files with the extension **.GL** on disk **C:** modified or created from January 1, 1984 to the current date. You are then prompted to insert a diskette in drive **A:**. If **.GL** files for February have been added, before you back up, these will be backed up as well as the January files.

BACKUP
(cont)

Example 2) – *Backing up all files and associated subdirectories modified since the last backup:*

The command **BACKUP C:/A/M/S** backups up all files and subdirectories on disk **C:** to the diskette you are prompted to insert in drive **A:**.

CHDIR
Change the Current Directory

ENTERING THE COMMAND

STEP

1) Press the soft key **F6** (CHDIR) on the DOS soft key line, or type in the command **CHDIR** (or **CD**).

2) Enter the parameters, edit the command line if necessary, then press the **Return** key.

COMMAND DESCRIPTION

The CHDIR command is used to change the current directory of the specified or default drive, or to display the current directory path of a drive.

The current directory is where DOS looks to find files and commands which were entered without specifying the directory they are in.

COMMAND FORMAT

CHDIR [[d:]path]

or

CD [[d:]path]

CHDIR
(cont)

PARAMETERS

–	If no parameters are entered, (or if only the drivespec is entered) CHDIR displays the current directory path on the specified or default drive.
d:	The *drivespec* of the drive where the current directory is located. If no *drivespec* is given, DOS assumes the default drive.
path	The *path* defining the current directory. The *path* consists of a list of directory names, separated by backslashes (\), needed to tell DOS the location of the current or specified directory.

ERROR MESSAGES

Possible error messages include:

– **Invalid directory**

The specified path or directory is incorrect.

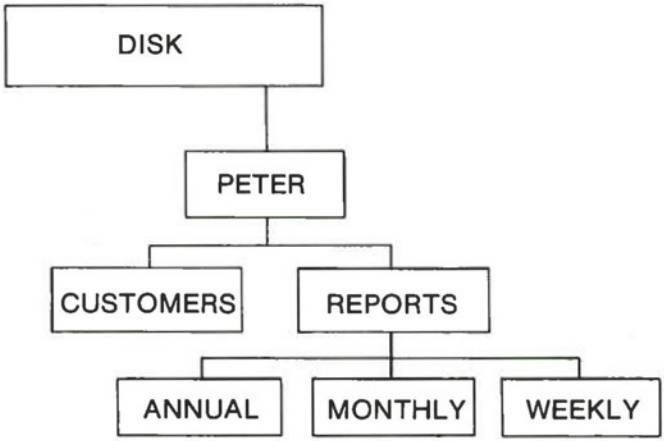
– **Invalid drive specification**

The drive specification is incorrect.

CHDIR
(cont)

EXAMPLES

Assume the following is a part of your directory hierarchy:



Example 1) – *To change the current directory from the root directory to the **WEEKLY** subdirectory:*

Entering **CD C:\PETER\REPORTS\WEEKLY** sets the WEEKLY subdirectory on drive **C:** as the current directory. The path defining the directory is \PETER\REPORTS\WEEKLY.

CHDIR
(cont)

Note, you can define a path that starts at the root directory from any directory. Otherwise, you must provide a continuous path from the current directory to the directory you want to make current.

Example 2) – *To change the directory from the **WEEKLY** subdirectory to the **CUSTOMERS** subdirectory:*

The command **CD ..\ ..\ CUSTOMERS** sets the CUSTOMERS subdirectory as the current directory. The .. tells DOS to back up one level in the directory hierarchy.

CHKDSK

Check and Display Diskette Status

ENTERING THE COMMAND

STEP

1) Press the soft key **F9** (CHKDSK) on the DISKS soft key line or type in the command **CHKDSK**.

2) Type in any desired parameters, edit the command line if necessary, and then press **Return**.

After the diskette status report is displayed on the screen, the system redisplays the prompt in preparation for the entry of a new command.

COMMAND DESCRIPTION

CHKDSK lets you examine both the directory and the file allocation table on a diskette, and produces a status report. It also reports the amount of internal (RAM) memory that is available within the Hyperion main unit, exclusive of the part used by the RAM disk.

If you want to have DOS fix any discrepancies found in the file allocation table or directory, use the /F parameter. If there are any errors, DOS prompts you to convert these lost clusters of data to files (see below under "User Interaction").

CHKDSK
(cont)

If you do not specify the /F parameter, DOS tells you that corrections won't be written to the disk (i.e. no corrections are saved) and then shows how many files would have been saved if you had used the /F parameter.

COMMAND FORMAT

CHKDSK[/F][/V] [d:][file]

PARAMETERS

–	If no parameters are entered, DOS will produce a diskette and memory status report for the current drive.
/F	This parameter must be used if you want to correct any errors in the file allocation table. If */F* is not entered, DOS still tells you how many files would have been saved if */F* had been used.
/V	When this parameter is specified, DOS displays messages showing its progress while checking the disk.
d:	The location of the *disk drive* on which a status report is produced. Note that a CHKDSK of the RAM disk also works, even though no real diskette exists in that virtual drive.

CHKDSK
(cont)

file If a *file* or *group of files* are specified, DOS displays the number of non-contiguous areas occupied by the file(s). You cannot specify a directory, so the *files* must be in the current directory.

WARNINGS

If a parameter is entered, CHKDSK temporarily makes that drive become the current drive. If an error causes CHKDSK to terminate prematurely, the current drive may not be correctly reset. You should check the system prompt upon termination of this command to ensure that the current drive is as expected.

Unlike FORMAT, DISKCOPY, and DISKCOMP, the CHKDSK command does not wait for the user to insert a diskette. The diskette should be inserted in the specified drive before the **Return** key is pressed to initiate the CHKDSK command.

USER INTERACTION

If DOS finds lost data sectors while performing CHKDSK, it prompts you with the following message:

```
NN lost clusters found in N chains
Convert lost chains to files (Y/N)?
```

If you respond with a **Y** to the prompt asking to recover lost data, the status report that follows includes a statement showing how much data and how many files have been recovered.

CHKDSK
(cont)

If you respond with a N, the status report that follows shows how much data and how many files would have been recovered if you had entered Y instead of N.

If the /F parameter was not specified, DOS still displays the message prompting you to convert lost clusters to files, just to show you what it would have done if /F were used.

The typical status report produced by DOS is of the following form. Note that depending on what DOS finds, some of the messages are not always shown:

```
Volume DISKNAME      Created January 10, 1984 10:30

    NNNNNN bytes disk space freed

    NNNNNN bytes total disk space
    NNNNNN bytes in N hidden files
    NNNNNN bytes in N directories
    NNNNNN bytes in N user files
    NNNNNN bytes in bad sectors
    NNNNNN bytes available on disk

    NNNNNNN bytes total memory
    NNNNNNN bytes free
```

The first line appears in certain cases when the CHKDSK command can free up disk space by an internal reorganization. This only happens if the /**F** parameter is specified.

The next 4 lines describe the state of the diskette in the specified drive. One byte is equivalent to one character of storage. If a large number of the bytes are in bad sectors, we suggest replacing and discarding the diskette.

CHKDSK
(cont)

The last 2 lines refer to the internal (RAM) memory within the Hyperion main unit. The number of bytes free will always be less than the amount of total memory, as a part of DOS itself is stored in this internal memory.

If the line displaying the number of bytes in hidden files reads "0 bytes in 1 hidden files" the "hidden file" in question is simply the location of the volume name (diskname) of the disk. This location is read by the VOL and DISKNAME commands, but CHKDSK considers it as an empty hidden file.

Certain applications may require a large amount of free RAM memory. If your application is displaying error messages indicating insufficient memory, use CHKDSK to check the amount of free memory, and reduce the size of the RAM disk if necessary using the MODE command.

The total amount of internal memory reported by CHKDSK does not include the part of memory used for the RAM disk. To obtain information on the RAM disk (drive C, or if a hard disk is installed, drive D), use CHKDSK C: (or D:) or MODE SHOW.

ERROR MESSAGES

Possible error messages include:

- **Invalid drive specification**
- **Invalid parameter**
- **Allocation error for file:** *filename.ext*
- **Directory error – file:** *filename.ext*
- **Disk not initialized**
- **File size error for file:** *filename.ext*
- **Files cross-linked:** *filename.ext* **and** *filename.ext*
- **NNNNNN bytes of disk space freed**

CLS
Clear the Display Screen

ENTERING THE COMMAND

STEP

1) Enter the characters **CLS**.

2) Press **Return**.

COMMAND DESCRIPTION

This command clears the display screen. If foreground and background colors have been selected, the selections remain unchanged, and the screen when cleared, displays the background color. These colors are represented by shades of grey on monochrome display screens.

This command is useful if you want to erase confidential information displayed on the screen when leaving your work area, or when someone without access privileges to the information is within viewing range.

WARNING

The contents of the screen are lost when this command is used. If the information has not been stored to one of the drives, it cannot be retrieved.

COPY
Copy Files

ENTERING THE COMMAND

STEP

1) Press the soft key **F6** (COPY) on the FILES soft key line or type in the word COPY.

2) Type in the parameters, edit the command line if necessary, then press the **Return** key.

The system copies the files, as specified by the parameters chosen, and redisplays the system prompt in preparation for the entry of a new command.

COMMAND DESCRIPTION

COPY is used to copy diskette files into other files, onto other diskettes, or onto external system devices. It is also used to make complete backups of all files on a given diskette. Because the same command performs so many different functions, and is used so frequently, it is important to understand all of its capabilities.

COMMAND FORMAT

COPY[{/**A**}{/**B**}][/**V**][**d:**][**path**][**file1**][**d:**][**path**][**file2**][{/**A**}{/**B**}]

(Note: the braces "{","}" indicate that only one of the parameters in that particular sequence can be used (e.g. either /A or /B, but not both).

COPY
(cont)

PARAMETERS

WHEN USED WITH THE SOURCE FILE:

/A Causes the file to be treated as an ASCII text file with data copied up to the first end of file (EOF) flag. The remainder of the file is not copied.

/B Causes the entire file to be copied.

/V Causes a *verification* of the copy process.

file The *filename and extension*, if any, of the *file* being copied.

WHEN USED WITH THE TARGET FILE:

/A Places an end of file flag on the file.

/B Causes no end of file flag to be added.

file The *filename and extension*, if any, of the *target file* into which the contents of the source file are written.

WHEN USED WITH EITHER FILE:

d: *Drivespec*. This is the location where the file to be copied either to or from resides. The file can be on any of drive *A: B: C:* or *D:*. If this is left blank on the source file or the target file, the system assumes that the file is on the default drive.

path The *path* (series of directory names) leading to the files.

COPY
(cont)

Copying more than one source file to one target file:

***file1*[+*file*+*file*+...]**

When *source files* are preceded by plus signs, they are concatenated (chained together in sequence) and copied to the target as one file. As with other uses of the COPY command, the *source files* are unchanged. If any wildcard characters such as asterisks or question marks are included, the system copies all files matching the wildcard filename and extension. See example 4.

WARNINGS

Because of the many options available within the COPY command, it is important to use extreme caution when entering the command line. If wildcards are going to be used, the DIR command (page II-43) should be used first to find out exactly what files match the wildcard pattern.

The contents of the target files in any copy operation are replaced by the contents of the source files. Even if a drive is specified as the target, any files on that drive that have the same names as any of the source files are replaced.

When using wildcard patterns to identify source files for concatenation, be sure the target file is not one of the files that matches the source pattern. This causes an error message, but it is displayed too late to save the contents of the target file.

COPY
(cont)

USER INTERACTION

Most user interaction with the COPY command takes place at command entry time, and has therefore already been described. After COPY has finished performing the requested file copies, the following message is displayed:

N files copied

and the system prompt reappears.

ERROR MESSAGES

Possible error messages are:

- **Invalid drive specification**
- **Invalid parameter**
- **File cannot be copied onto itself**

SEE ALSO

Use **DIR** to verify wildcard pattern selections before applying them to the COPY command.

When copying all of the files on a diskette to another diskette, it may be useful to use **FORMAT** to clear the diskette first. **FORMAT/S** will also move DOS onto a target diskette. **DISKCOPY** can be used to create an exact diskette copy, in place of using the COPY command followed by the parameter *.*.

COPY
(cont)

EXAMPLES

Example 1) – *Copying a group of files onto another diskette:*

The command **COPY A:*.TXT B:** copies all files on drive A that have the extension **TXT** onto drive B.

Example 2) – *Copying all files on a diskette:*

The command **COPY A:*.* B:** copies all files on the diskette in drive A onto the diskette in drive B.

Example 3) – *Copying a file into a file with a different name:*

Entering **COPY A:ANYFILE.EXT B:*.BAK** copies the file ANYFILE.EXT from the diskette in drive A into a file called ANYFILE.BAK on drive B. Note that when COPY copies one file into another in this way, the two files can be on the same drive if desired.

Example 4) – *Copying several files into one:*

Entering **COPY A:MYNAME.TXT + *.BAK A:BACKUP.BIG** copies MYNAME.TXT into BACKUP.BIG first. Then, all files that have the extension .BAK are also copied into BACKUP.BIG.

COPY
(cont)

Example 5) – *Performing several file copies at once:*

The command **COPY A:*.TXT A:*.BAK** copies each file with the extension .TXT into a file with the same filename, but the extension .BAK.

Example 6) – *Joining several files together and copying onto other files in one step:*

Entering **COPY A:*.INX + A:*.TXT A:*.BAK** joins each file with the extension .INX, to a file with the same name but the extension .TXT, and copies both into a file with the same name but with the extension .BAK.

CTTY
Change Standard Input/Output Console

ENTERING THE COMMAND

STEP

1) Enter the command **CTTY.**

2) Enter the device name, and then press **Return**.

COMMAND DESCRIPTION

This command is used to change the standard input/output console to an auxiliary console, or to restore the keyboard and screen as a standard input/output console.

The CTTY command accepts the name of *any* character-oriented device to allow you to install your own device drivers and specify their device names. However, the device must be able to send as well as receive data. Do not specify a printer, for example, since DOS will attempt to read from as well as write to it.

COMMAND FORMAT

CTTY devicename

CTTY
(cont)

PARAMETERS

devicename The name of the device being set up as the standard input/output device. AUX, COM1, or COM2 causes DOS to use that device as the main console. DOS reverts to using the primary console as the standard input/output device when you enter CON.

DATE

Display or Modify System Date

ENTERING THE COMMAND

STEP

1) Press the soft key **F4** (DATE) from the DISKS or FILES soft key line,

OR

Type in the word **DATE**, followed by any desired parameters, and press **Return**.

(STEPS are continued under USER INTERACTION)

COMMAND DESCRIPTION

The **DATE** and **TIME** commands allow the user to display the current values used by the Hyperion, and to reset them.

COMMAND FORMAT

DATE [mm-dd-yy]

DATE
(cont)

OPTIONAL PARAMETERS

–	Not entering a parameter makes the system prompt for a date, displaying: **Enter new date:**
mm-dd-yy	The new date must be entered with hyphens or dashes as mm-dd-yy or mm/dd/yy. *Mm* is the *month* of the year (*1 to 12*); dd is the *day* of the month (*1 to 31*); *yy* is the *year* of this century (*80 to 99*) or next century (*2000 to 2099*).

WARNINGS

It is very important to keep accurate time and date values. DOS uses them to maintain the last change date and time information in each diskette's directory. As well, certain programs may use these values in other ways.

USER INTERACTION

If a new date is entered on the command line, the system resets the date and redisplays the system prompt.

Pressing the **Return** key without entering a parameter, makes the system prompt reappear. The date is not changed.

DATE
(cont)

USER INTERACTION

STEP *(cont)*

2) Press **Return**.

OR

Enter a new date, and press **Return**.

The system prompt reappears, in preparation for the entry of a new line.

Day-of-week is displayed as an extra memory aid, for date verification by the user. When a new date is entered, the day of the week must not be typed in.

ERROR MESSAGES

Possible error messages include:

- **Invalid date**
  Enter new date:

SEE ALSO

The **TIME** command is used to reset the internal clock time of day. This command is not available on any soft key line since it would probably be used only very infrequently.

DEL
Delete a File from Disk

See the ERASE command in this chapter on Page II-57.

```
COMMAND  COM   15957  3-20-84  12:00p
ANSI     SYS    9314  5-10-84  10:17a
RAMDISK  SYS     720  5-08-84   3:52p
LPFILTER SYS    4226  3-20-84  12:00p
CLOCK    SYS     973  3-20-84  12:00p
CONFIG   SYS      42  5-22-84   2:52p
EXTVIDEO SYS    3250  3-20-84  12:00p
STAR     COM    1993  3-20-84  12:00p
LOCK     COM     364  3-20-84  12:00p
UNLOCK   COM     368  3-20-84  12:00p
FORMAT   COM    5437  5-11-84   8:19p
SYS      COM    1103  3-20-84  12:00p
DISKCOPY COM    2364  3-20-84  12:00p
DISKCOMP COM    1948  5-10-84  11:03a
EDLIN    COM    8080  3-20-84  12:00p
BACKUP   COM    3253  3-20-84  12:00p
RESTORE  COM    3378  3-20-84  12:00p
PRINT    COM    4506  3-20-84  12:00p
RECOVER  COM    2295  3-20-84  12:00p
ASSIGN   COM     813  3-20-84  12:00p
TREE     COM    2226  3-20-84  12:00p
GRAPHICS COM     962  3-20-84  12:00p
MORE     COM     295  5-11-84   5:25p
Strike a key when ready . . .
LASTLN Disks Files MODE DIR/P 14:27   CHDIR PATH TREE XPLAIN HELP
```

Fig. II-2 – A typical directory listing. Note the filenames, filename extensions, file sizes, dates and times.

DIR

List Files and Directories on a Disk

ENTERING THE COMMAND

STEP

1) Press the soft key **F5** (DIR/P) or type in the command **DIR**.

2) Type in any desired parameters; edit the command line if necessary; and then press **Return**.

The system lists the files, as specified by the parameter entered, then redisplays the system prompt in preparation for the entry of a new command.

Note that using the soft key **F5** enters **DIR/P** onto the screen, automatically adding the parameter **/P.**

COMMAND DESCRIPTION

This command is used to list the files on a disk. See Fig. II-2 at left for a sample listing. Note that the date and time listed in the directory entry is the date and time the file was created or last changed. The size of the file (in bytes) is given, and all subdirectories are identified with the legend <DIR>.

DIR
(cont)

COMMAND FORMAT

DIR [/P][/W][d:][path][file]

OPTIONAL PARAMETERS

–	Not entering a parameter (pressing **Return** without entering a parameter) instructs the system to list all files on the default disk in the current drive.
/P	*Pause*. Instructs the system to *pause* after the screen has been completely filled. The system stops scrolling the directory display until you press any character key.
d:	*Drivespec*. The letter *d* stands for the drive where the files are to be found. Note that there is no space between this parameter and the file, if a file is specified. If a *drivespec* is not entered, DOS looks at the current drive identified by the system prompt.
path	The *path* (series of directory names) leading to the files.

DIR
(cont)

file The *filename and extension*. Substituting an asterisk (*) for any sequence of letters in a *filename* or *filename extension* results in a display of all *files* with *filenames* that match the characters preceding the asterisk.

Substituting a question mark (?) for a *single* character in the file displays all files whose filenames and extensions match the other characters.

/W *Wide*. This parameter limits extent of display. Only filenames and extensions are displayed. This allows the system to list five files per line, so that all files on a disk can be displayed in any one screenful.

EXAMPLES

Example 1) – *Listing files with the same extensions:*

The command **DIR A:*.TXT** lists all files on the current drive that have the extension **TXT:**

```
Volume in drive B has no label
Directory of B:\

LETTER     TXT  4567  3-05-84  11:12a
CHAPTER1   TXT  5066  3-12-84   3:47p
           2 File(s)    45345 bytes free.
```

DIR
(cont)

Example 2) – *Listing files with similar filenames:*

The command **DIR B:MY*.DTA** lists all files on drive B that begin with **MY** and have the extension **DTA**.

```
Volume in drive B has no label
Directory of B:\

MYGL     DTA    5531    3-07-84    10:47a
MYPLAN   DTA    5034    3-12-84     2:57p
         2 File(s)    42345 bytes free.
```

Example 3) – *Listing files with similar filenames:*

The command **DIR A:MYFILE?.*** lists all files on drive A that begin with **MYFILE**, that have exactly one subsequent character in their filenames, with or without an extension.

```
Volume in drive B has no label
Directory of B:\

MYFILE1  TXT    5531    3-07-84    10:47a
MYFILE1  DTA    5337    3-12-84     2:57p
MYFILEZ  DOC    4034    3-12-84     2:57p
         3 File(s)    39346 bytes free.
```

Example 4) – *Using the [/W] option:*

The command **DIR/W A:** lists all the files on the diskette in drive A, five to a line, without their size or last change date information.

DISKCOMP

Compare Two Diskettes

ENTERING THE COMMAND

STEP

1) Press the soft key **F7** (D-COMP) on the DISKS soft key line.

OR

Type in the command **DISKCOMP**, add any desired parameters and edit the command line if necessary. Then press **Return**.

(STEPS continued under USER INTERACTION)

Note that using the soft key **F7** to enter **DISKCOMP** automatically adds a **Return** after the command. The system then assumes that the two diskettes to be compared are to be found in drives A and B.

COMMAND DESCRIPTION

This command does a direct physical comparison of two diskettes, on a use-of-location basis, *NOT* on a file by file basis.

Note that a diskette copy that is made using the file-oriented COPY command won't compare the new copy with its parent diskette unless the /V parameter is specified. With the COPY command, DOS tries to write all the data in the file in adjoining areas. However, on the parent diskette, any given file may be contained in sectors scattered over the diskette.

DISKCOMP
(cont)

As a result, the only way to make sure both diskettes are identical is to use the DISKCOPY command. DISKCOPY also copies hidden files (the DOS system files, for example).

COMMAND FORMAT

DISKCOMP [d1: d2:]

PARAMETERS

–	If no parameters are entered, DOS assumes the diskettes in drives A and B are being compared.
d1: d2:	The *drivespecs* (locations of the diskettes being compared). The same drive specification may be given for both diskettes: DOS then operates by asking the user to swap between diskettes at certain intervals.

WARNING

Two diskettes are normally reported as being identical only if they are DISKCOPYs of each other, or if they were both generated in exactly the same way. Two diskettes containing identical files may not necessarily be reported as comparable, because of different internal allocations of storage space to files.

DISKCOMP
(cont)

USER INTERACTION

DISKCOMP always displays the following message:

Insert first diskette in drive x :

Insert second diskette in drive y :

Strike a key when ready . . .

where **x** and **y** are the two *drivespecs* entered on the command line, or **A** and **B** respectively if no drivespecs are entered.

STEPS *(cont)*

2) Insert the first diskette into drive **x**.

3) Insert the second diskette into drive **y**.

4) Press any key on the keyboard to start comparing.

OR

Press **Ctrl + Break** to cancel the compare request.

...continued

If the two diskettes are not identical, this is followed by messages of the following form, where track numbers vary between 0 and 39, and the side number is 0 or 1.

Compare error(s) on
Track NN Side N, Sector(s):NN

DISKCOMP
(cont)

If the two diskettes are identical, however, DISKCOMP reports successful comparison:

Diskettes compare OK

After each diskette comparison is completed, DISKCOMP asks:

Compare more diskettes (Y/N)?

STEP *(cont)*

5) Enter **Y** for yes, and return to Step 2, or enter **N** for no to stop comparing diskettes.

ERROR MESSAGES

Possible error messages are:

- **Invalid drive specification**
- **Invalid parameter**
- **Compare error(s) on track NN side N, sector(s):NN**
- **Incompatible diskette or drive types**
- **Unrecoverable read error on drive X**
  Track NN, side N
- **Not ready error reading drive X**
  Correct, then strike any key
- **First and second diskettes are not formatted identically**
  Continue comparing? (Y/N):

DISKCOPY
Duplicate a Diskette

ENTERING THE COMMAND

STEP

1) Press the soft key **F6** (D-COPY) on the DISKS soft key line.

OR

Type in the command **DISKCOPY**, add any desired parameters and edit the command line if necessary. Then press **Return**.

(STEPS continue under USER INTERACTION)

Note that using the soft key **F6** to enter **DISKCOPY** automatically enters a **Return** after the command. The system then assumes that the diskette in drive A is to be copied onto the diskette in drive B.

COMMAND DESCRIPTION

This command is used to make backup copies of diskettes. It works with a "source drive" (**d1**) and a "target drive" (**d2**). All information on the diskette mounted in the source drive is exactly duplicated onto the diskette mounted in the target drive.

DISKCOPY
(cont)

COMMAND FORMAT

DISKCOPY [d1: d2:]

PARAMETERS

– If no parameters are entered, DISKCOPY assumes the source drivespec is A: and the target drivespec is B:.

d1: The *source drivespec*, which can be *A:* or *B:*, identifies the diskette to be copied from. If a source drivespec is specified, then the target drivespec must also be provided.

d2: *Target drivespec*. Identifies the diskette to be copied onto. The source and target drives can be the same if desired. This capability has been provided so that backups can be made even if a hardware problem has temporarily put one drive out of commission.

WARNING

DISKCOPY destroys any files that were present on the target diskette prior to diskcopying.

DISKCOPY
(cont)

USER INTERACTION

DISKCOPY always displays the following message on the screen; where **x** and **y** are the two *drivespecs* entered on the command line, or A and B respectively if no drivespecs were entered.

Insert source diskette in drive x:

Insert target diskette in drive y:

Strike a key when ready . . .

STEP *(cont)*

2) Make sure that the target diskette *does not contain any needed files*, and that it is write-enabled (no write-protect sticker on the write-protect notch).

... continued

At this time, the user must make absolutely sure that the diskette in the target drive does not contain any needed files. They are destroyed as soon as any key is struck. *To cancel the DISKCOPY command* at this point, enter **Ctrl + Break.**

DISKCOPY
(cont)

STEP *(cont)*

3) Insert the source diskette into drive *x*.

4) Insert the target diskette into drive *y*.

5) Press any key on the keyboard, to start the copying.

OR

Press **Ctrl + Break** to cancel the copying request.

... continued

DISKCOPY automatically detects whether one or two sides are being used on the source diskette, and proceeds accordingly. The command informs the user by displaying the following message:

Copying N side(s), N sectors per track

In the case of a single drive DISKCOPY, where source and target diskettes are to occupy the same drive, the user is prompted to swap between source and target diskettes during the DISKCOPY. These swap instructions must be followed very carefully, to avoid copying *from* target *to* source diskette by mistake.

When the first diskette copy is completed, DISKCOPY informs the user of this, and asks whether more diskette copies are to be made:

Copy Complete
Copy Another (Y/N)?

DISKCOPY
(cont)

STEP *(cont)*

6) Enter **Y** for yes, and return to Step 3, or enter **N** for no to stop copying diskettes.

If **Y** is typed, the complete sequence is repeated using the same source and target drives. If **N** (or anything else) is typed, DISKCOPY terminates, and DOS prompts for another command.

If the format of the target diskette is different from the source diskette the following message is displayed:

Source and destination diskettes are not formatted identically Continue Copying (Y/N)?

If your response to this message is "Y", the system displays the following message:

Formatting while copying

ERROR MESSAGES

Possible error messages are:

- **Invalid drive specification**
- **Invalid parameter**
- **Attempted write-protect violation:**

... continued

DISKCOPY
(cont)

ERROR MESSAGES *(cont)*

- **Not ready error reading drive X**
  Correct, then strike any key
- **Not ready error writing drive X**
  Correct, then strike any key
- **Target diskette may be unsuitable**
- **Target diskette write protected**
  Correct then strike any key
- **Unrecoverable format error on target diskette**
- **Target diskette unusable**
- **Unrecoverable read error on source**
  Track NN, side N
- **Unrecoverable verify error on target**
  Track NN, side N
- **Unrecoverable write error on target**
  Track NN, side N

SEE ALSO

The COPY command can also be used to make backup diskettes on a file-by-file basis. In fact, using COPY offers an advantage: files are copied into contiguous areas on the diskette, whereas DISKCOPY duplicates diskettes exactly.

During day to day use, DOS shrinks and expands files as needed. Often, a file is expanded by allocating storage space that *is not* directly beside the space previously allocated to the file. This means that a diskette file can be spread over many different areas of the diskette. Using COPY restructures each file into one physical area. Using DISKCOPY does not.

DISKNAME

Display a Diskname or Rename a Disk

ENTERING THE COMMAND

STEP

1) Press the soft key **F2** (D-NAME) from the DISKS soft key line, or type in the command **DISKNAME**.

2) Enter the parameters, edit the command line, if necessary, and then press **Return**.

After displaying or re-setting a diskname (volume name), the system redisplays the system prompt in preparation for the entry of a new command.

COMMAND DESCRIPTION

This command allows you to display a diskname for reference, or to assign a name to your disk. The diskname is identical to the volume name. Entering either DISKNAME or VOL displays the diskname (volume name), but only DISKNAME allows you to change the diskname (volume name).

While the Hyperion does not enforce the use of the DISKNAME command, regular use of the command is a highly recommended practice.

COMMAND FORMAT

DISKNAME [d:] [newname]

DISKNAME
(cont)

PARAMETERS

–	When the DISKNAME command is entered without any parameters, it displays the disk-name of the current drive.
d:	*Drivespec*. Can be any valid disk drive. This identifies the location of the disk.
newname	*Newname* is a name of up to 11 characters that is to be assigned to the disk in the specified drive.

ERROR MESSAGES

Possible error messages are:

– **Unable to set diskette name**

Disk error in writing name, check to see if disk is write-protected.

– **This diskette does not allow a diskname.**
– **Diskette name unreadable**

Diskette not formatted by a Hyperion, format diskette before using DISKNAME command.

– **No diskette mounted**
– **No diskette name assigned**

Diskette is either not in the drive, or has not been assigned a name.

ERASE
DEL

Remove a File From Disk

ENTERING THE COMMAND

STEP

1) Press the soft key **F8** (ERASE) on the FILES soft key line or type either the command word **ERASE** or command word **DEL**.

2) Enter the parameters, edit the command line if necessary, then press the **Return** key.

(STEPS continued under USER INTERACTION)

COMMAND DESCRIPTION

ERASE and DEL perform the same command function. This command completely removes a file or files from a disk. In fact, it removes all references to the file in both the directory and the storage allocation table on the diskette. It is *impossible* to recover a file after it has been erased.

COMMAND FORMAT

ERASE [d:][path]file

or

DEL [d:][path]file

ERASE
DEL
(cont)

PARAMETERS

d: The *disk drive* location of the files to be deleted. If no *drivespec* is entered, the current drive is assumed.

path The *path* (series of directory names) leading to the file(s) you want to erase.

file *Filename* plus *filename extension*. Identifies the file(s) to be erased. Wildcards (* and ?) can be used, but it is strongly recommended that you perform a DIR to check just how many files match the wildcarded filename and extension. All *files* that match are erased from the specified diskette.

WARNINGS

An erased file cannot be recovered. Use extreme caution if using wildcards in the filename and extension supplied on the command line.

Certain DOS system files cannot be erased, even if the filename and extension pattern matches their own.

ERASE
DEL
(cont)

USER INTERACTION

If *.* is the specified file parameter, DOS checks that you are sure you want to execute the command, since all files on the disk match this filename and extension. DOS asks:

Are you sure (Y/N)?

STEP *(cont)*

3) Type in **N** for no, or **Y** for yes. Then press **Return**.

If **Y** is typed, all files are erased. Typing **N** (or anything else) cancels the ERASE request.

ERROR MESSAGES

Possible error messages are:

- **Invalid drive specification**
- **Invalid number of parameters**
- **File not found**
- **Missing filename**

EXE2BIN

Convert .EXE Files to .COM Files

ENTERING THE COMMAND

STEP

1) Enter the command **EXE2BIN**.

2) Enter the parameters, edit the command line, if necessary, and then press **Return**.

COMMAND DESCRIPTION

This command is used to convert .EXE files that have no segment fixup to a form that is compatible with .COM program files. This results in saving diskette space and faster loading of programs. Two kinds of conversions are possible, depending on the specified initial CS:IP. They are pure binary conversion, or segment relocatable conversion.

If CS:IP is not specified in the input program, a pure binary conversion is assumed. If segment fixups are necessary, the following prompt appears:

Fix-up needed - base segment (hex):

By typing a legal hexadecimal number and then pressing **Return**, execution will continue.

EXE2BIN
(cont)

If CS:IP is specified as 100H in the input program, then it is assumed the file is to be run as a .COM file ORGed at 100H, and the first 100H of the file is to be deleted. No segment fixups are allowed, as .COM files must be segment relocatable.

Note that to produce standard .COM files with the Macro Assembler, one must both ORG the file at 100H and specify the first location as the start address (this is done in the END statement). A sample program listing that does this would be:

```
ORG     100H
START:
        .
        .
        .
END         START
```


COMMAND FORMAT

EXE2BIN [d1:][path]file [d2:][file]

PARAMETERS

d1: The *drivespecs*. The first *(d1)* is the location
d2: where the file to be converted resides. The second *(d2)*, is the location of the diskette to which the converted file is sent. If *d2* is not specified, it is assumed to be the same as *d1*.

EXE2BIN
(cont)

path The *path* defining the current directory. The *path* consists of a list of directory names, separated by backslashes (\), needed to tell DOS the location of the current or specified directory.

file The *filename and extension*, if any. Wildcards may be used, and will result in all files matching the *file* being converted. If no output *file* is entered, the input *file* is used. If no destination *filename* is entered, the program assumes the *filename* of the input file. If no input *file extension* is specified, the program assumes the extension is .EXE. If no destination *file extension* is specified, the new file is given the extension .BIN.

WARNING

The input files must be valid .EXE format as produced by the linker. The *resident*, or actual code and data part of the file must be less than 64K. There must be no STACK segment.

EXE2BIN
(cont)

ERROR MESSAGE

Possible error messages include:

- **File cannot be converted**

  This message is displayed if the file is not a valid .EXE file, or if the CS:IP does not meet the necessary criteria (see command description above).

EXPLAIN

Explain System Commands or Features

ENTERING THE COMMAND

STEP

1) Press **F9** (XPLAIN) on the DOS soft key line, or type in the word **EXPLAIN**.

2) Enter the feature or command to be explained, and press **Return**.

COMMAND DESCRIPTION

EXPLAIN displays information on all DOS commands and some features of the Hyperion. This information provides, in effect, a quick reference guide inside the computer. If a drive is specified in the command line all the files with a filename extension .EXP on the specified drive are presented for selection.

You can write an explain file to be included on the explain selection list by editing a text file and putting it on diskette with a filename extension .EXP.

COMMAND FORMAT

EXPLAIN [command or feature name]

EXPLAIN
(cont)

PARAMETER

–	Entering no parameter displays a description of the EXPLAIN command, followed by a list of the commands or features for which there are EXPLAIN descriptions.
command or feature name	The *command or feature name* you enter prompts the system to display a description of that command or Hyperion feature.

USER INTERACTION

Many of the available explanations are longer than the 24 lines of the Hyperion screen. To continue the explanation, strike any key on the keyboard. When there is no more information to be displayed, the system prompt reappears.

FIND

Search Files for Specified Text

ENTERING THE COMMAND

STEP

1) Enter the word **FIND**.

2) Enter the parameters, edit the command line, if necessary, and then press **Return**.

COMMAND DESCRIPTION

This filter is used to search for specified strings of text in the files you specify. The output depends on the parameters you specify: the lines containing the specified string, the lines not containing the specified string, or a list of the files indicating which ones contain the string and which files do not.

COMMAND FORMAT

FIND[/V][/C][/N] "string"[d:][path]file

PARAMETERS

/V This parameter causes all lines *not containing* the string to be displayed.

FIND
(cont)

/C This parameter indicates that the FIND command will only display the number of matching occurrences of the matching lines in each file, without displaying the matching lines themselves.

/N This parameter causes the *relative line number* of each matching line to be displayed along with the selected lines.

d: The *drivespec*. The location of the files to be searched. If no *drivespec* is entered, the current drive is assumed.

path The *path* leading to the file being searched. The *path* consists of a list of directory names, separated by backslashes (\).

file The *filename* and extension, if any. Wildcards may not be used.

WARNING

If there are any discrepancies between the text you are searching for, and the string you type in, DOS searches only for the string that's typed in. Extra spaces between words, or capital letters can affect the FIND command.

SEE ALSO

The FIND command can be used with a number of other commands using the piping feature provided by DOS (see the examples below).

FIND
(cont)

EXAMPLES

Example 1) – *Finding the number of occurrences of a string of text in a file:*

Entering **FIND/C "to be or not to be" A:HAMLET.TXT** causes DOS to display the message:

```
_________ 2A:HAMLET.TXT: 1
```

Example 2) – *Listing the occurrences of a string of text in a file along with the line number:*

Entering **FIND/N "to be or not to be" A:HAMLET.TXT** causes DOS to display the message:

```
_________ 2A:HAMLET.TXT
[897] to be or not to be, that is the question
```

Example 3) – *Listing the lines in a file not containing a string of text:*

Entering **DIR A:FIND/V "1-10-84"** causes DOS to display the following:

```
Volume in drive A has no label
Directory of A:\

myfile1   txt     2346   1-07-84    9:47a
myfile2   txt     2346   1-08-84    5:16p
myfile3   txt     2346   1-09-84    2:33p
myfile7   txt     2346   1-11-84   11:17a
myfile8   txt     2346   1-12-84    9:32a
  5 File(s)     193,4546 bytes free
```

FORMAT

Prepare a Diskette for Use

ENTERING THE COMMAND

STEP

1) Press the soft key **F8** (FORMAT) on the DISKS soft key line, or type in the word **FORMAT**.

2) Enter the parameters, edit the command line, if necessary, and then press **Return**.

(STEPS continued under USER INTERACTION)

COMMAND DESCRIPTION

When a diskette leaves the manufacturer's plant, it typically contains a useless random magnetic pattern. Every computer system provides a diskette formatting capability that replaces this random pattern with an organized and recognizable pattern, and prepares special diskette areas as later required.

With DOS on the Hyperion, the FORMAT command performs three functions. First, it writes a recognizable magnetic pattern onto the diskette, including addressing information. Then it creates an empty "directory" and "space allocation table" on the diskette.

The third step is performed only if the /S option is selected on the command line: a copy of DOS itself is placed onto the diskette.

FORMAT
(cont)

COMMAND FORMAT

FORMAT[/B][/V][/S][/1][/8][d:]

PARAMETERS

/B Formats the diskette with 8 sectors per track (see the /8 parameter) and allocates a space on the diskette for the system files. You can then add the system files for any version of DOS using the SYS command for the particular version of DOS you wish to use. (This is useful when formatting diskettes for use with application programs that use DOS but do not supply DOS with the program.)

/V Prompts you for a *volume name (diskname)* after formatting the diskette. If you do not use /V, you can use DISKNAME to enter a volume name on the diskette.

/S If the /S option is selected on the command line, DOS copies certain files onto the specified diskette from the diskette in the current drive: they are IO.SYS, MSDOS.SYS, and COMMAND.COM, in that order.

/1 If the */1* option is selected, the target diskette will be formatted for *single-sided use* even if it is a double-sided diskette. This feature is used to create diskettes for non-Hyperion systems that have *single-sided* drives.

FORMAT
(cont)

/8 The */8 parameter* formats the diskette with *8 sectors per track* to maintain compatibility with earlier versions of DOS. DOS 2.11 normally formats diskettes with 9 sectors per track to give you more storage space on the diskette. If the /8 is not specified, DOS always formats diskettes with 9 sectors per track.

d: The *drivespec*, which identifies the drive in which the diskette is to be formatted. The *drivespec* cannot be that of the RAM disk.

WARNING

FORMAT destroys any files that were present on the target diskette before formatting.

USER INTERACTION

Entering the FORMAT command line always displays the following message on the screen (where d: is the drive specified on the command line):

Insert new diskette for drive d:
and strike any key when ready

STEP *(cont)*

3) Make sure that the new diskette does not contain any needed files, and that it is write-enabled (the write-protect tab is off).

...continued

FORMAT
(cont)

At this time, the user must make sure that the diskette in the specified Hyperion drive does not contain any needed files: they are destroyed, as soon as any key is struck. *To cancel the FORMAT command* at this point, enter **Ctrl + Break.**

STEP *(cont)*

4) If you are formatting a diskette, insert it in the appropriate disk drive. (Ignore this step if you are formatting a hard disk.)

5) Press any key on the keyboard.

...continued

If the **parameter** is specified, and the DOS system files are not on the disk in the default drive (nor on drive A) DOS prompts:

```
Insert DOS disk in drive A:
and strike any key when ready
```

The following message appears on the screen:

```
Formatting cylinder: 000
```

and the three-digit number begins to increase as the system formats the disk. When the formatting is completed, DOS displays:

```
Format complete
```

FORMAT
(cont)

If the /S parameter was specified, the following message then appears on the screen:

System transferred

If the /V parameter was specified, DOS then prompts you:

Volume Label (11 characters, Enter for none)?

STEP *(cont)*

6) If you are prompted for the volume label (diskname), type in a label (11 characters maximum) and press **Return**. If you do not want to type in a volume label, press the **Return** key (which the prompt refers to as the "Enter" key).

continued

When a diskette format is completed, a status report is displayed in the following form:

NNNNNN bytes total disk space
NNNNNN bytes used by system
NNNNNN bytes in bad sectors
NNNNNN bytes available on disk

FORMAT
(cont)

More diskettes can then be formatted. The system prompts you with the message:

Format Another (Y/N)?

STEP *(cont)*

7) Enter **Y** for yes, and return to Step 3, or enter **N** to stop formatting diskettes.

If "Y" is typed, the complete sequence is repeated. If "N" (or anything else) is typed, FORMAT terminates, and DOS prompts for another command.

ERROR MESSAGES

Possible error messages are:

- **Invalid drive specification**
- **Invalid parameter**
- **Attempted write-protect violation:**
- **Disk unsuitable for system disk**
- **Format failure**
- **Track 0 bad disk unusable**
- **Unable to write BOOT**
- **Parameters not compatible**

GRAPHICS

Print Graphics Displays

ENTERING THE COMMAND

STEP

1) Enter the word **GRAPHICS**.
2) Press **Return**.
3) Press **Shift** + **Print**.

COMMAND DESCRIPTION

This command is used to print the contents of a graphics display to a dot-matrix printer. This command increases the resident size of DOS in memory by 688 bytes.

If the screen is in text mode, the text can be printed in less than 30 seconds.

If the screen is in graphics mode, the contents are printed in up to four shades of grey. If the resolution is 640 by 200, the screen is printed sideways on the paper, with the upper right corner of the screen printed on the upper left corner of the paper.

COMMAND FORMAT

GRAPHICS

LOCK

Lock a File to Prevent Erasure

ENTERING THE COMMAND

STEP

1) Enter the command **LOCK**.

2) Enter the parameters, edit the command line, if necessary, and then press **Return**.

COMMAND DESCRIPTION

This command locks the contents of a file to prevent unintentional erasure or change. To erase or modify a "locked" file, you must first "unlock" it using the UNLOCK command. The LOCK command does not prevent unauthorized access to your files, and their unlocking and subsequent erasure or modification.

You can still access any locked file to copy or edit it. If you edit a locked file, the .BAK version retains the locked feature. Any attempt to delete it, either directly, or by re-editing the copy made of it (still bearing the original filename and extension) is not allowed by DOS.

COMMAND FORMAT

LOCK [d:][path]file

PARAMETERS

d: The *drivespec*. The location of the file to be locked. If no drivespec is entered, the current drive is assumed.

LOCK
(cont)

path The *path* leading to the file being searched. The *path* consists of a list of directory names, separated by backslashes (\).

file The *filename and extension*, if any. Wildcards may not be used.

SEE ALSO

Files locked with the LOCK command can only be erased or changed by "unlocking" them with the UNLOCK command.

EXAMPLES

Actions allowed by DOS on a Locked File:

a) **EDLIN MYFILE.TXT** – Call up MYFILE.TXT, edit a copy of the file and store it. The locked file is now MYFILE.BAK.

b) **EDLIN MYFILE.TXT** – You can call up MYFILE.TXT, edit a copy of the file and store it, but not under the same name, as this would delete the existing .BAK version which is locked. The new copy must be renamed, MYFILE1.TXT, for instance.

c) **ERASE MYFILE.TXT** – Since this file is a copy of the locked file, it is not protected.

d) **RENAME MYFILE.BAK MYFILE.LOC** – The locked version is now called MYFILE.LOC.

e) **COPY MYFILE.LOC MYFILE.COP** – The locked version is still MYFILE.LOC. MYFILE.COP can be changed or deleted.

(Note, in the above examples, all files are assumed to be in the current directory on the default drive.)

MKDIR

Create a new Subdirectory

ENTERING THE COMMAND

STEP

1) Type the command **MKDIR** (or **MD**).

2) Enter the parameters, edit the command line if necessary, then press the **Return** key.

COMMAND DESCRIPTION

The MKDIR command is used to create a new subdirectory on the specified or default drive. You may create as many subdirectories as you wish, given the amount of disk space you have available. However, you should make sure that the maximum length of any path from the root directory to the desired level is less than or equal to 63 characters, including backslashes.

COMMAND FORMAT

MKDIR [[d:]path]

or

MD [[d:]path]

MKDIR
(cont)

PARAMETERS

d: The *drivespec* of the drive on which the directory being created is located. If no *drivespec* is given, DOS assumes the default drive.

path The *path* defining the new directory. The *path* consists of a list of directory names, separated by backslashes (\), needed to tell DOS the location of the new directory. You cannot create two directories with the same *path* on the same disk.

ERROR MESSAGES

Possible error messages include:

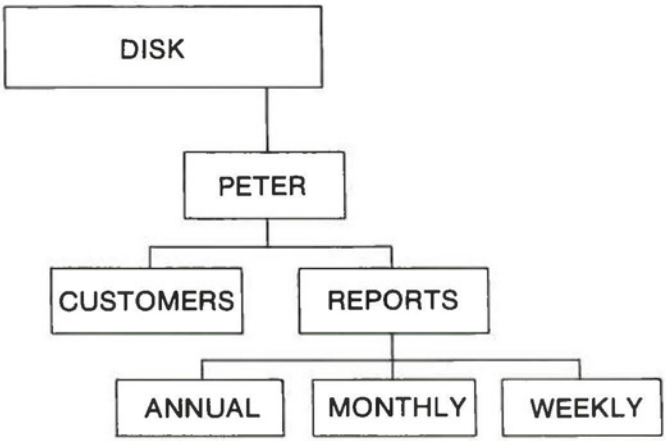
– **Unable to create directory**

The path specified to the directory being created is not valid.

EXAMPLES

Assume that you want to create a group of directories like those shown on the next page as part of your over-all disk directory hierarchy.

MKDIR
(cont)



Example 1) – To create the ***WEEKLY*** *directory when the current directory is the root directory:*

Entering **MD C:\PETER\REPORTS\WEEKLY** creates the WEEKLY directory on drive **C:**. Directories PETER and REPORTS must already exist.

Example 2) – *To create the CUSTOMERS directory when the current directory is WEEKLY:*

The command **MD ..\..\CUSTOMERS** creates the CUSTOMERS directory on drive **C:** The directory REPORTS must exist. The .. tells DOS to back up one level in the directory hierarchy. You must provide a continuous path; each directory must be linked with the directory before and after it.

MODE

Modify Certain System Settings

COMMAND DESCRIPTION

MODE is used to temporarily or permanently modify a number of system settings. These settings include certain display options, printer settings, serial and parallel port settings, size of the RAM disk etc. The MODE command can:

a) ***Show***. Display, on one screen, all current MODE settings (direct MODE only).

b) ***Update from***. Change MODE settings to those previously saved onto a disk in drive A, B or C.

c) ***Save to***. Save all current MODE settings onto disk.

d) ***Screen***. Set display screen options. For example, column width and attribute interpretation.

e) ***Softkeys***. Assign soft key labels (interactive MODE only).

f) ***LPT***. Set printer options. For example, define how data is interpreted by the printer.

MODE
(cont)

g) ***Redirection***. Direct printer output to parallel or serial ports.

h) ***COM***. Set asynchronous port options, such as baud rate, parity, stop bits, etc.

i) ***Ramdisk***. Install or delete the RAM disk and set its size.

j) ***Misc***. Set the number of open files, disk buffers, the break parameter and real time clock option.

A MODE function can be requested directly, like a DOS command, by entering the word MODE followed by the appropriate parameters. Since MODE changes are usually saved to the CONFIG.SYS file, DOS looks for this file on the default drive. If CONFIG.SYS is not there, DOS looks for the system files. If neither CONFIG.SYS nor the system files are found, DOS displays the following message:

Cannot find CONFIG.SYS or system on default drive d:.

where **d:** is the default drive.

The MODE command can also be used interactively by entering the word MODE by itself, without parameters. DOS then calls up a series of MODE menus, enabling you to select the appropriate options by moving a cursor about the screen.

MODE
(cont)

ENTERING THE COMMAND

Mode can be used either interactively (Step 1a) or directly (Step 1b).

STEP

1a) Press the soft key **F4** (MODE) on the DOS soft key line, or type the word **MODE** and press **Return**. Make sure the disk in the default drive contains the MODE and system files.

This displays a menu on the screen and enables you to select MODE options interactively.

A DESCRIPTION OF INTERACTIVE MODE BEGINS ON PAGE II-88.

OR

1b) Type in word **MODE** followed by the parameters described in the DIRECT MODE section of this description. Edit the command line if necessary, then press the **Return** key.

A DESCRIPTION OF THE DOS COMMAND FORMAT FOR DIRECT MODE BEGINS ON PAGE II-111.

MODE
(cont)

USER INTERACTION (INTERACTIVE MODE)

If you press the **F4** soft key (MODE), or type in MODE and press **Return** without any parameters, DOS displays a mini-menu (the main MODE menu) of options, as shown in Fig. II-3.

If the disk in the default drive does not contain the CONFIG.SYS file or the system files, an error message is displayed instead.

If this occurs, you must insert a diskette containing the CONFIG.SYS file or the system files in the default drive, or else change the default drive to that containing the disk with the required files. Once you have done so, re-enter the MODE command. DOS will display the main MODE menu.

The soft key line displayed at the bottom of the screen also changes to display a sequence of cursor movement commands. By pressing the appropriate soft keys or special keyboard keys you can move the cursor about the screen to select the MODE options you wish. Selected items are shown enclosed in parentheses.

Cursor movement is controlled in the same way for all MODE screens. Table II-A lists the keys and their cursor control function. (The soft keys F6 to F9 have different functions on the Softkey Editing screen.)

MODE
(cont)

Table II-A
BLOCK CURSOR MOVEMENT CONTROL WITHIN MODE MENUS

KEYS	EFFECT
Tab, spacebar, → *or* **F9**	Moves right to the next item.
Rub Out, Shift + Tab ← *or* **F8**	Moves left to the previous item.
↑ *or* **F6**	Selects item and moves up to the next line.
↓ *or* **F7**	Selects item and moves down to the next line.
Return	Makes a selection and displays the appropriate menu.
Esc	Returns to the previous menu, or DOS.

The main MODE menu is the first menu displayed when the MODE command is entered (interactive MODE). Nine selections are available from this menu. They are listed on page II-91.

MODE
(cont)

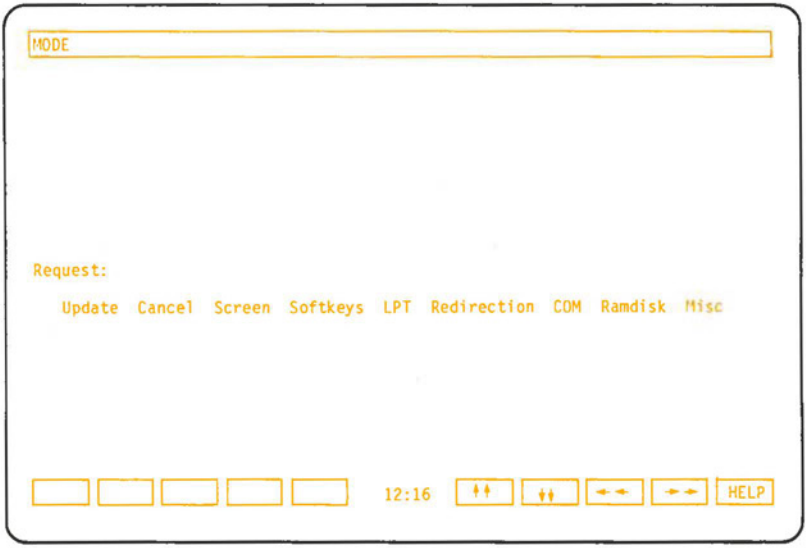


Fig. II-3 – The main MODE menu.

MODE
(cont)

SELECTION	RESULT
Update	Enables you to update the current MODE settings. Displays the Exit menu (Fig. II-4).
Cancel	Enables you to return to DOS without changing the MODE settings. Displays the Cancel menu as shown in Fig. II-5.
Screen	Enables you to select screen display format and keyboard interpretation. Displays the Screen menu as shown in Fig. II-6.
Softkeys	Enables you to modify the labels and functions of the soft keys on the DOS soft key lines. Displays the Softkey Editing menu so you can edit soft key labels and actions (Fig. II-7).
LPT	Enables you to select line width, lines per inch, and output translation. Displays the Select Printer Number menu as shown in Fig. II-8.
Redirection	Enables you to direct the output to three different printers through the three possible Hyperion output ports (both serial and parallel). Displays the Redirection of Printer Output menu as shown in Fig. II-11.
COM	Enables you to set serial communications parameters, such as baud rate, parity, data bits, stop bits, and retry option. Displays the Select Communications Port menu (Fig. II-12).
Ramdisk	Enables you to create a RAM disk and set its size. Displays the RAMdisk menu (Fig. II-14).
Misc	Enables you to set up the number of open files and disk buffers, the break parameter, and set the real time clock option. Displays the Miscellaneous menu as shown in Fig. II-15.

MODE (Update)

Fig. II-4 – Exiting option selections, the Update menu.

MODE (Update)
(cont)

Selecting the ***Update*** option from the main MODE menu changes the menu to allow a selection of three possibilities (as shown below). All possibilities result in a return to DOS.

SELECTION	RESULT
Save to Dos	Exits from the MODE selection procedure back into DOS. The system prompt is displayed in preparation for the entry of another DOS command. If you have not saved the settings to a disk, they remain in effect for the current session, but are erased once you turn off the Hyperion or perform a system restart. Note that certain settings, such as RAM disk size, do not take effect unless stored onto a disk and until DOS is restarted from that disk.
Save X:	Saves the current settings onto the disk in drive **X**, where **X** is the default drive. Using this disk for the next system restart resets the Hyperion to these new values. The file CONFIG.SYS, containing the MODE settings, is updated. If changes are made to the softkey labels and actions, the SFKEY.DAT file is updated (if present) or created.
Update from X:	Change current MODE settings to those stored in the disk in drive **X**, where **X** is the default drive. All MODE settings, including the ones set by this program during this MODE session, are overridden.

MODE (Cancel)

Fig. II-5 – Cancel option selections: the Cancel menu.

MODE (Cancel)
(cont)

Selecting the ***Cancel*** option from the main MODE menu changes the menu to allow a selection of two possibilities (see Fig. II-5).

SELECTION	RESULT
Cancel	Exits from the MODE selection procedure back into DOS. The settings, if they have been changed, are returned to previously set values. Any changes you have made are cancelled for the current session. Any changes you have saved to disk take effect when you use the appropriate disk for a system restart.
MODE	Returns to the main MODE menu.

MODE(Screen)

Selecting the ***Screen*** option from the main MODE menu results in a menu of six items as shown in Fig. II-6. The initial default settings are shown enclosed in parentheses.

The *column width* parameter allows you to select the width (in number of characters) in the display screen. You can set the width to 40 or 80 characters per line.

MODE (Screen)
(cont)

The *Screen Mode* default value is **Color** so the Hyperion can emulate the IBM color graphics card, (even though the screen output remains monochrome). **Black and White** emulates IBM-compatible black and white mode. **Mono** tells the Hyperion to emulate the IBM monochrome graphics card. Use this setting for applications that do not work in color mode. In **Mono/Color** mode, the Hyperion can emulate some features of both color and monochrome cards (address ranges, for example).


```
MODE

Screen:

Column Width                  :  40  80
Screen Mode                   : (Color) Black and White  MONO  MONO/Color
Output Translation            :  Off (ANSI)
Command Line Environment      : (KEY=ON) KEY=OFF
Screen Wait State             : (WAIT=YES) WAIT=NO
Video Adapter Hardware        : (Internal) External

                                12:16                          HELP
```

Fig. II-6 – Screen option selections: the Screen menu.

MODE (Screen)
(cont)

The *Output Translation* parameter can be set to **OFF** or **ANSI.** It controls the way the screen display interprets the keyboard characters and escape sequences.

ANSI and **OFF** call up console filters, used to interpret the keyboard characters and escape sequences. Choosing ANSI sets the screen filter to understand ANSI x3.64-1979 attribute escape sequences. For normal (0), bold (1), underscore (4), subscript (11), and superscript (12) character attributes, only those fonts supported by the set attribute interpretation are displayed.

The *Command line environment* parameter allows you to enable or disable the displaying of the soft key line. The ANSI filter (ANSI.SYS) must be loaded to display the soft key line.

The *Screen wait state* should not be set permanently to **NO.** If it is set to **NO**, then the screen remains permanently on and the screen life may suffer. This option is provided for users who may wish to disable screen blanking for extended periods for purposes such as demonstration sessions.

The *Video Adapter Hardware* option must be set to **Internal** for your system to work. It should only be set to **External** for a color graphics card in an expansion chassis.

Pressing **ESC** exits you back into the main MODE menu. Options that have been selected, are retained, but not selected until you update MODE.

MODE (Softkeys)

Selecting the ***Softkeys*** option from the main MODE menu permits you to change the soft key labels displayed in the four DOS soft key lines, and the commands these labels call up.

You can select the softkey you wish to edit by using the cursor control keys (see Table II-A on Page II-89) to move the block cursor about the screen display from label to label.

Pressing the **Return** key causes the cursor to move to the field labelled *Softkey Name*. The softkey label you chose can now be edited, or re-entered.


```
MODE

Softkey Editing:

Softkey Line 1
 LASTLN   Disks   Files   MODE    DIR/P   CHDIR   PATH    TREE    XPLAIN   HELP

Softkey Line 2
   Dos    Disks   TYPE    DATE    DIR/P   COPY    PRINT   ERASE   RENAME   HELP

Softkey Line 3
   Dos   D-NAME   Files   DATE    DIR/P  D-COPY  D-COMP  FORMAT  CHKDSK   HELP

Softkey Line 4
   Dos     PRN   \PATH\   !MORE   !SORT    A:      B:      C:      D:      Rtn

Softkey Name  :  LASTLN

Softkey Action:  LASTLN

 SAVE  CANCEL  RTN  LASTLN  BEEP   12:16   GOTO 1  GOTO 2  GOTO 3  GOTO 4  HELP
```

Fig. II-7 – The Softkey Editing menu display screen.

MODE (Softkeys)
(cont)

After you enter the softkey label, move your cursor down to the *Softkey Action* line and type in the commands you want displayed on the screen when this soft key is pressed.

In some cases, you may want the parameters softkey line displayed when a command is entered from another line. To add this function to the softkey you are defining, press one of the softkeys **F6** to **F9.** This adds an instruction calling up another softkey line to the end of the action defined for the soft key.

If the soft key action is a command that takes no parameters, you may want to add the **Return** function (**F3** on the soft key line) as part of the action.

SOFTKEY SELECTION		EFFECT
F1	SAVE	Saves changes made and returns the system to the main MODE menu.
F2	CANCEL	Cancels editing session and redisplays the main MODE menu. Any changes are lost.
F3	RTN	Adds a Return function to the softkey action.
F4	LASTLN	Adds a LASTLN command to the softkey action. When pressing the modified soft key labbel from DOS, the previously entered DOS command line is appended to the modified command.
F5	BEEP	Adds a bell signal that shows as a musical note in the SOFTKEY ACTION field, and in the soft key label.
F6-F9	GOTO N	Adds an instruction to display another DOS soft key line to the soft key label. N can be from 1 to 4, since there are four soft key lines in DOS.

MODE (LPT)

```
MODE

Select Printer Number:  LPT1 [PRN]   LPT2  LPT3
```

Fig. II-8 – Select Printer Number menu.


```
MODE

LPT1:  (Printer)

Character Width Horizontal Spacing:  Normal/80  Compressed/132
Lines per Inch Vertical Spacing   : (6) 8
Output Translation                : (Off) STRIP  External:
```

Fig. II-9 – LPT1 parameter selection screen.

```
MODE

LPT2:  (Alternate Printer)

Character Width Horizontal Spacing:  Normal/80  Compressed/132
Lines per Inch Vertical Spacing   : (6) 8
```

Fig. II-10 – LPT2 and LPT3 parameter selection screens.

MODE (LPT)
(cont)

Making the LPT selection from the main MODE menu enables you to set parameters for up to 3 printers connected to the Hyperion. These printers are designated LPT1, LPT2 and LPT3. The default parameters are shown in parentheses in Fig. II-9 and II-10.

These settings should be set to correspond with the abilities of the printer connected to the Hyperion. Once they are correctly set, they should be stored on your disk.

Fig. II-8 shows the ***Select Printer Number*** menu. Pressing Return from this menu displays the appropriate parameter selection menu for the printer selected (Fig. II-9 and II-10). Only LPT1 can have an output translation print filter assigned to it.

The *character width horizontal spacing* and *Lines per inch vertical spacing* parameters only affect printing on certain printers that respond to pre-defined commands to change vertical spacing and perform compressed print (to fit 132 columns on 8 1/2″ wide paper). These are ineffective if the translation is OFF or STRIP, and can only be used if the appropriate print filter for the attached printer is set in the *Output translation* parameter.

Selecting the *Output translation* parameter highlights not only the option itself, but also a space to the right. The flashing cursor waits for you to enter a filename of up to 8 characters into this space.

The filename identifies a print filter that contains a program used by the printer to interpret the characters and escape sequences it receives from the Hyperion. The filter must be contained in a file with that filename, and a filename extension of .LPT on the diskette in drive A in order to take effect.

MODE (LPT)
(cont)

IMPORTANT: LPT1 is the only printer that can be assigned a print filter. The Redirection MODE option is used to assign LTP1 to an output port. Also, the printer for which the print filter was designed must be physically connected to that port for printing to take place.

MODE (Redirection)


```
MODE

Redirection of Printer Output:

Printer Number:  LPT1 [PRN]   LPT2  LPT3
External Port :  Parallel [:] (Serial [¦]/COM1) Serial [¦]/COM2

                                16:27                          HELP
```

Fig. II-11 – Assigning a printer to a Hyperion port: Redirection of Printer Output menu.

MODE (Redirection)
(cont)

Selecting ***Redirection*** from the main MODE menu displays the ***Redirection of Printer Output*** menu (shown in Fig. II-11). This enables you to assign a printer number to any of three possible output ports: a parallel port at the back of the main unit, or one of two serial ports.

Only one printer can function at any one time. If you have three printers each connected to one port, and you want to activate each in turn, you would have to reset the "print direction" each time you change printers. The other printer attributes would be set via the LPT selection from the main MODE menu.

There are two option lines on this screen:

1) Line 1 allows you to select one of three printers to be redirected. In actual usage LPT1 is normally chosen. The printer selected must be compatible with the parameters selected via the LPT MODE option.

2) Line 2 shows the other redirection options. "Parallel" output goes to parallel port: "COM1" output goes to serial port 1: "COM2" output goes to serial port 2. Both serial ports are available through an expansion interface as an option.

MODE (COM)

```
MODE

Select Communications Port:  COM1 [AUX]   COM2
```

Fig. II-12 – Selecting the communications port for serial communications: Select Communications Port menu.


```
MODE

COM1: (Serial Communications [!])

Baud Rate        :  110  150  300  600  1200  2400  4800  9600
Parity           : (None) Odd  Even
Data Bits        :  7 (8)
Stop Bits        : (1) 2
Retry on Timeout:  Yes (No)
Optional Driver : (None) COMSIO  COMACA
```

Fig. II-13 – The parameters screen for COM1.

MODE (COM)
(cont)

Selecting the COM option from the main MODE menu results in a ***Serial Communications Port*** menu (Fig. II-12). Two communication ports are possible: COM1 and COM2. The settings for each must be set separately. The delivered serial communications settings for COM1 are shown in Fig. II-13 as enclosed in parentheses. Settings for COM2 are similar.

The serial output standard supported by the Hyperion is highly flexible, to allow use of many different printers and other serial devices.

These settings must be set to match the settings of the external serial device.

The *baud rate* is the data transfer speed, in bits per second. The *Parity* is a data verification scheme. The *Data Bits* refers to the number of bits that are transferred per character. The *Stop Bits* are part of the communications synchronization.

The *Retry on Timeout* feature is useful when redirecting printer output through a serial port (see Redirection option). It asks the Hyperion to retry if there is no response from the connected device within a certain number of seconds. Although this option appears on both COM1 and COM2 menus it is the same for both and changing one changes the other.

MODE (RAM disk)

Fig. II-14 – Internal memory size option selection after being set to 90 K.

MODE (RAM disk)
(cont)

Selecting the ***Ramdisk*** on from the main MODE menu displays the ***Ramdisk*** menu (Fig. II-14). This screen provides you with the ability to select a RAM disk size. If 0 is selected, then no RAM disk is installed.

All sizes are in units of 1 kilobyte. (1 kilobyte = 1024 bytes.) The smallest RAM disk size permitted is 3K.

If you enter an unacceptable value for RAM disk size, DOS sets the RAM disk to 10K.

Available free RAM memory is displayed for reference. Certain software application packages may demand a large amount of free RAM memory. Check the manual provided with such packages to determine the free memory you need, and set the RAM disk accordingly.

MODE (Misc)

Selecting the ***MISC*** option from the main MODE menu displays the ***Miscellaneous DOS Settings*** menu (Fig. II-15). The settings for a number of parameters in the configuration file can be set here.

```
MODE

Miscellaneous DOS Settings:

Real Time Clock :  Yes  No
Max Open Files  : (8 )
Sensitive Break :  On (Off)
DOS Disk Buffers: (2 )
Optional Driver : (              )

                                12:16                    HELP
```

Fig. II-15 – The Miscellaneous DOS Settings Menu.

MODE (Misc)
(cont)

If you have the real time clock option, move the cursor to the **Yes** field next to the *Real Time Clock:* menu item. This tells DOS to check the real time clock for the time when starting up. If you do not have the real time clock option, press **Return** to leave the default setting of **No.**

The default value maximum number for *Max Open Files* is **8**. This should be enough for most applications. Increase this number if your software applications are sending error messages indicating an insufficient number of open files.

Above the default value of 8, each file opened increases the portion of DOS resident in memory by 39 bytes.

Setting the *Sensitive Break* to **On** causes DOS to check for the **Ctrl + Break** key combination every time it performs a function. Setting this feature to **Off** (the default value) causes DOS to check for the **Ctrl + Break** combination only when accessing the screen, keyboard or printer.

The default number for *DOS Disk Buffers* is **2**. This should be enough for applications not performing a large number of random disk accesses. Users with a hard disk should increase this value to 3. If you have applications performing a large number of random accesses, (data base applications, for example) set the number of buffers to between 10 and 20.

Each extra buffer uses 528 bytes of RAM. Too many buffers reduces the amount of free memory available, and can slow down the system.

The *Optional Driver* menu selection allows you to type in the name of any optional driver you want to install (e.g. for networking).

MODE (SHOW)

```
Screen:
Column Width                    :80
Screen Mode                     :Color
Output Translation              :ANSI
Command Line Environment        :KEY=ON
Screen Wait State               :WAIT=YES
Video Adapter Hardware          :Internal
LPT1:  (Printer)
Character Width Horizontal Spacing:Normal/80
Lines per Inch Vertical Spacing   :6
Output Translation                :Off
Redirection of Printer Output:
Printer Number:LPT1 [PRN]
External Port  :Parallel [¦¦]]
COM1: (Serial Communications [¦]
Baud Rate       :2400
Parity          :None
Data Bits       :8
Stop Bits       :1
Retry on Timeout:No
Optional Driver :None
Ramdisk:
Logical Disk Size:0
A>
LASTLN  Disks  Files  MODE  DIR/P  16:27   CHDIR  PATH  TREE  XPLAIN  HELP
```

Fig. II-16 – MODE SHOW current settings.

MODE (SHOW/UPDATE)
(cont)

COMMAND FORMATS AND PARAMETERS

The **MODE** command can be used directly: it has several formats, depending on the system modifications you wish to make.

1) To display current system settings, enter:

MODE SHOW

2) To update machine mode from drive specified, enter:

MODE UPDATE [=d:]

d: The *drivespec:* must be a drive containing a disk with CONFIG.SYS or the system files. If CONFIG.SYS or the system files are not present, then the following message appears:

Disk error occurred while reading MODE parameters.
Unable to create CONFIG.SYS

MODE displays the following error message if an invalid drive is specified for the UPDATE or SAVE options:

Invalid drive letter specified

Note: if no drive is specified, current settings are saved to the default drive.

MODE (SAVE/SCREEN)

3) To save all current settings, enter:

MODE SAVE [=d:]

Used to save **MODE** settings to a disk drive.

d: The *drivespec*. Must be a drive containing a disk with CONFIG.SYS or the system files. If CONFIG.SYS or the system files are not present, then the following error message appears:

Disk error occurred while writing MODE parameters.
Unable to create CONFIG.SYS

Note: if no drive is specified, current settings are saved to the default drive.

4) To set display screen options, enter:

MODE n[,[d][,[T][,[w][,[k][,TRANS=s]]]]

n *Column width:* enter 40 or 80 characters per line.

d *Screen shift direction:* can be R for right or L for left. If the "T" parameter is not specified afterwards, the display is shifted once and MODE exits.

MODE (SCREEN)
(cont)

T *Test pattern:* enter T to display a test pattern: "0123.....890" (40/80 character wide). A message prompt appears:

Can see the 0/9 on the left/right of the screen ? (Y/N)

Answering **N** causes the screen to shift and the prompt is redisplayed. Answering **Y** exits.

w *Wait state:* can be I for IBM emulation (no wait) or H for Hyperion (wait enabled). No wait leaves screen permanently on; wait enabled causes the screen to blank out after three minutes of inactivity.

k *Softkeys:* enabled when k = H(yperion), no soft keys displayed when k = I(BM).

s *Attribute Translation:* the TRANS parameter, s, can be:

OFF for no character translation, (same as IBM)

ANSI to set the screen filter to understand ANSI x3.64-1979 attribute escape sequences for character attributes (normal (0), bold (1), underscore (4), subscript (11), superscript (12)). Only those fonts supported by the set attribute interpretation are displayed.

Examples:

```
MODE 80,L,,H,I,TRANS = ANSI
MODE 40,,,I,TRANS = OFF
MODE 80,L,T
```

MODE (LPT)

5) To set printer options, enter:

MODE LPT#:[n][,[m][,TRANS = s]]

Controls the line printer attributes.

*Printer number:* can be 1, 2 or 3.

n *Character width horizontal spacing:* sets print size using the loaded "Output translation" print filter. Enter 80 (normal sized print), or 132 (compressed print).

m *Lines per inch vertical spacing:* sets vertical spacing using the loaded "Output translation" print filter. Enter 6 (lines per inch), or 8 (lines per inch).

s *Output Translation:* sets the print filter to define the **TRANS** parameter: **OFF, STRIP,** or a filename of an external print filter.

OFF for no character translation and no definition of horizontal and vertical spacing.

STRIP to absorb the ANSI escape sequences for character attributes, (normal (0), bold (1), underscore (4), subscript (11), superscript (12)); all control characters except for form feed, carriage return and line feed; and all non-ASCII characters. This filter does not provide a definition for horizontal and vertical spacing.

MODE (LPT/REDIRECTION)
(cont)

a filename, an external print filter to provide translation of escape sequences for horizontal and vertical spacing parameters and also to translate character attribute standard escape sequences in the print output stream into specific sequences particular to the attached printer. The EPSON external print filter is delivered on the master DOS diskette. The extension of the filename is not specified.

For example, enter:

```
MODE LPT1:132,6,TRANS=OFF
or MODE LPT2:,,TRANS=EPSON
```

6) To redirect printer output to the serial port, for use with the expansion unit only, enter:

```
MODE LPT#:=COMn
```

*Printer number:* can be 1, 2 or 3.

n *Asynchronous adaptor:* can be 1 or 2.

For example, enter:

```
MODE LPT1:=COM2
```

MODE (COM)

7) To set asynchronous port options, enter:

MODE COMn:[[[[[Baud],Parity],Databits],Stop Bits],P]

Controls serial port characteristics.

n *Communications port adaptor number:* can be, 1 or 2.

Baud Rate *Rate of character transfer:* = 110, 150, 300, 600, 1200, 2400, 4800, 9600, bits per second (you may enter only the first two digits).

Parity Setting must match that of external device = N (none), O (odd), E (even).

Databits These too must match the settings of the external device = 7 or 8.

Stop Bits = 1 or 2 .

P If the *Retry on Timeout* parameter is specified timeout errors are retried indefinitely until successful. If this parameter is not specified, timeouts are not retried. This parameter can only be specified with a serial printer connected to the port. Entering **Ctrl + Break** cancels retries.

For example, enter:

MODE COM1:60,N,7,1
or **MODE COM2:,,,,P**

MODE (RAMDISK/BREAK)

8) To set RAM disk size, enter:

MODE RAMdisk = n

Specifies RAM disk existence and size.

n *Logical disk size* (in bytes). Enter 0 to 640 (K), in increments of 1K. Only 3 digits are looked at by the system. Make sure your current drive is not C: when defining RAM disk size.

Mode does not allow a RAM disk size smaller than 4 K bytes. If you do so, the system displays the following message:

Please specify a Ramdisk size greater than 3K.

You must reboot the system after changing the RAM disk size for the changes to take effect.

9) To set the Break option, enter:

BREAK = ON *or* **BREAK = OFF**

WARNINGS

The MODE settings described in this user guide control how the Hyperion interfaces with various external and internal devices. By altering these settings, the Hyperion can communicate with a wide variety of devices, and can emulate many different types of devices.

Many of the difficulties encountered in running third-party software can be removed by altering the **MODE** settings.

MODE
(cont)

Secondly, **MODE** settings are not permanent until they are saved onto a diskette, usually the Hyperion DOS diskette in drive A. The **MODE** settings are then loaded automatically into the Hyperion when a system restart is done.

THE MODE PROGRAM

As the system boots, the programs resident in DOS are loaded, defining a particular environment and operating parameters for the system.

Usually, the AUTOEXEC.BAT file executes and runs the **MODE** program, setting the system configuration and loading any optional drivers. This AUTOEXEC.BAT file must exist to run the **MODE** program automatically, but an application disk that requires no special drivers can omit running the MODE program, and is identical to an IBM system disk.

MODE MESSAGES

MODE issues messages about successful completion or problems with functions. These messages are descriptive, and relevant only to the function selected. They also describe whether or not **MODE** exits normally, or whether it terminates without the required change accepted.

MORE
Read a Screenful of Data

ENTERING THE COMMAND

STEP

1) Enter the word **MORE**.

2) Enter the parameters, edit the command line, if necessary, and then press **Return**.

COMMAND DESCRIPTION

The MORE command is a filter used to read a screenful of data from a specified file and send it to the specified output device. Filters are special commands that take information from a standard input device, transform (i.e. "filter") it, and send the output to the standard output device. Information can be sent to the MORE filter via "pipes" or by redirection of the standard input device.

After each screenful, DOS pauses and prompts:

— MORE —

When you are ready to continue, press any key to receive the next screenful of data. This continues until all the data in the file has been read.

COMMAND FORMAT

MORE <[d:][path][file]

MORE
(cont)

PARAMETERS

d: *Drivespec*. The location of the file to be read and displayed. If no *drivespec* is entered, the current drive is assumed.

path The *path* defining the current directory. The *path* consists of a list of directory names, separated by backslashes (\), needed to tell DOS the location of the current or specified directory.

file The *filename and extension*, if any. Wildcards may not be used.

EXAMPLES

Example 1) *– Displaying the contents of a file one screenful at a time:*

Entering **MORE<A:\LETTERS\MOM** sends the contents of the file MOM as input to the MORE command for display on the screen.

Example 2) *– Displaying and sorting the contents of a file one screenful at a time:*

Entering **SORT A:CUSTNAME.USA |MORE** sorts the information in CUSTNAME.USA and displays it on the screen. If the file contains U.S. customer names, they are displayed in alphabetical order, a screenful at a time. (Note that there must be a space between the last character of the file name and the "|" character.)

PATH

Search Directories for Commands

ENTERING THE COMMAND

STEP

1) Press the F7 soft key (PATH) on the DOS soft key line, or type in the word **PATH**.
2) Enter the parameters, edit the command line if necessary, then press the **Return** key.

COMMAND DESCRIPTION

The PATH command is used to tell DOS to search specified directories for commands or batch files that were not found by a search of the current directory.

COMMAND FORMAT

PATH[d:]path[;[d:]path]...]

PARAMETERS

– If no parameters are entered, DOS searches the names specified on the previous PATH command (i.e. the search path currently defined to DOS).

PATH
(cont)

; Entering PATH followed only by a semicolon resets the search path to null, which is the default when DOS is started. In this case, DOS searches only the current directory for commands and batch files.

d: The *drivespec* of the drive on which the subdirectories being listed are located. If no *drivespec* is given, DOS assumes the default drive.

path The *path* (series of directory names) leading to the directory DOS will search for any commands not found in the current directory on the specified drive.

EXAMPLE

To instruct DOS to search drives A and B when it cannot find a command or batch file on drive C:

Entering **PATH = A:\;B:** tells DOS to search the root directories of the other drives.

PRINT

Print Files While Doing Other Tasks

ENTERING THE COMMAND

STEP

1) Press the **F7** soft key (PRINT) on the FILES soft key line, or type in the word **PRINT**.

2) Enter the parameters, edit the command line, if necessary, and then press **Return**.

(STEPS continue under USER INTERACTION)

COMMAND DESCRIPTION

This command is used to print a list (queue) of files in the "background" while you use DOS to perform other operations on your Hyperion. Up to 10 files can be listed for printing at one time. The first time you use the PRINT command in a session, the resident size of DOS in memory is increased by about 3,200 bytes.

All tab characters are expanded with blanks to the next eight-column boundary.

COMMAND FORMAT

PRINT [[d:]file] [/T][/C][/P]...]

PRINT
(cont)

PARAMETERS

– If no / parameters are specified in the command line, ***/P*** is assumed and the files listed are queued for printing.

d: The *drivespec*. This is the location of the disk containing the files to be printed.

file The *filename and extension*, if any. Wildcards may be used, and will result in all files matching the filespec being printed. More than one file can be entered in the command line. However, all files must be in the current directory as there is no path parameter in the command format. Once the files are queued, you can change directories without affecting the PRINT operation.

/T This parameter sets the *terminate* mode. All queued files currently in the print queue are cancelled and if a file is currently being printed, printing stops. A cancellation message is printed and the paper advances to the top of the next page.

PRINT
(cont)

/C This parameter sets the *cancel* mode to allow you to select which files to cancel. The preceding filename and all following filenames entered on the command line are cancelled from the queue until a */P* is found on the command line or you press **Return**.

/P The */P* command sets the *print* mode. The preceding filename and all following filenames are added to the queue until a */C* is found on the command line, or you press **Return**.

WARNINGS

Be sure the device you name as the output device is attached to your Hyperion. If you name an unconnected or non-existent device, unpredictable system behavior may result.

Cancel any ASSIGN command statements in effect before using the PRINT command. Using ASSIGN can hide the true device type from DOS when it needs the actual drive information.

USER INTERACTION

The first time you use the PRINT command after you start or re-boot your Hyperion, the following message is displayed:

Name of list device[PRN]:

PRINT
(cont)

STEP *(cont)*

3) Enter the name of your output device (one of **LPT1, LPT2, LPT3, PRN, COM1, COM2, AUX,** etc.) and press **Return**. The default is PRN. This value is used if you press **Return** without typing in a device name.

EXAMPLES

Example 1) – *To queue 3 files for printing:*

Entering **PRINT A:FILE1.TXT A:FILE2 B:FILE3** causes each of the 3 specified files to be printed.

Example 2) – *To cancel a file in the queue and add one for printing:*

Entering **PRINT A:FILE1.TXT/C B:FILE2/P** causes FILE1.TXT to be cancelled. FILE2 is added to the print queue. If any of the files slated to be cancelled is currently being printed, it will terminate printing and displays the following:

– **Filespec cancelled by operator**

PROMPT

Set a new System Prompt

ENTERING THE COMMAND

STEP

1) Enter the word **PROMPT**.

2) Enter the parameters, edit the command line, if necessary, and then press **Return**.

COMMAND DESCRIPTION

This command is used to set a new system prompt. This prompt can be any text string or one of certain values (see metacharacter list on the following page) that you choose.

COMMAND FORMAT

PROMPT [prompt-text]

PARAMETERS

If no parameter is entered, the normal DOS prompt is assumed.

You may include special characters, called "metacharacters" in your prompt-text to include certain special features (the time or date, for example) in your prompt.

PROMPT
(cont)

These special metacharacters can be embedded in the text in the form **$c** where **c** is one of the following:

$	the "$" character
t	the current time
d	the current date
p	the current directory of the default drive
v	the DOS version number
n	the default drive
g	the ">" character
l	the "<" character
b	the "|" character
q	the "=" character
h	a backspace and erasure of the previous character
e	the ESCape character
–	the CR LF sequence (the carriage return sequence, going to the start of the next line)

Any other character is treated as a null character, and no action is taken on it by the PROMPT command. If blanks are deliberately or inadvertently added to the end of the prompt text, they will be made part of the prompt, taking up part of the prompt line. To eliminate unnecessary blanks, the delete key must be used.

SEE ALSO

DOS sets the system prompt to the drivespec followed by the greater-than sign (e.g. A>) whenever you boot (start) the system. If you want to customize the prompt for your system, you may want to insert a PROMPT command in the AUTOEXEC.BAT file. For more information on the AUTOEXEC.BAT file, and batch files in general, see Section 2 of Part V.

PROMPT
(cont)

EXAMPLES

Example 1) – *To set the prompt to the current directory:*

Entering **PROMPT p_ng** sets the prompt to the current directory followed by a carriage return and then the normal DOS prompt. Instead of the normal DOS prompt (e.g. A>) the new prompt is:

```
d:\subdirectory1\subdirectory2
d>
```

where d: is the drivespec and the subdirectories form the path to the current directory. The second line (d>) is the command line, and the cursor is displayed next to the ">".

Example 2) – *To prompt to the normal DOS prompt:*

Entering **PROMPT** without any parameters sets the prompt to the normal DOS (e.g. A>).

RECOVER

Recover Files From a Defective Disk

ENTERING THE COMMAND

STEP

1) Enter the word **RECOVER**.

2) Enter the parameters, edit the command line, if necessary, and then press **Return**.

COMMAND DESCRIPTION

This command is used to recover files from a hard disk that have developed a defective sector. You can recover all the information in the file containing the defective sector except for the data in the bad sector itself. If the disk directory has been damaged, you can still recover all the files on the disk.

The size of the recovered file is a multiple of the DOS allocation unit size. In most cases this is larger than the original file size. Text files will normally require re-editing to remove unwanted data from the end of the recovered file before being used.

COMMAND FORMAT

RECOVER [d:][path][file]

or

RECOVER d:

RECOVER
(cont)

PARAMETERS

d: *Drivespec*. This is the location of the disk where the files are to be recovered. If you do not specify a drive, the default drive is used. If only the *drivespec* is entered RECOVER assumes the directory is damaged, and recovers all files on the specified disk. Before using this second format (with no filename), please read the WARNING on this page.

path The *path* defining the current directory. The *path* consists of a list of directory names, separated by backslashes (\), needed to tell DOS the location of the current or specified directory.

file The *filename and extension*, if any. Wildcards may be used, but only the first file that matches the filename will be recovered. The RECOVER command only recovers one file at a time when filename and extension is entered.

WARNING

The RECOVER command followed only by a drivespec (without a filename) should only be used if the disk directory is damaged. Because DOS has no way to know whether the data in the directory is valid when executing the RECOVER command, it assumes that the entire directory is invalid. As a result, it recovers all files and gives them names of the form:

FILE*nnnn*.REC

where *nnnn* is a sequential number starting with 0001. Each FILEnnnn.REC name becomes the name of one of the recovered files on the disk. Files that still have valid names in the directory being recovered will receive FILEnnnn.REC names as well.

RENAME
Rename a File

ENTERING THE COMMAND

STEP

1) Press the soft key **F9** (RENAME) on the FILES soft key line or type in the word RENAME.
2) Type in the names of the files, edit the command line if necessary, and press the **Return** key.

COMMAND DESCRIPTION

This file management command is used to give existing files a new filename and/or extension. It can be used in the short form REN, or in the long form RENAME.

COMMAND FORMAT

REN[AME] [d:][path]file1 file2

PARAMETERS

d: The *drivespec*, or location of the file to be renamed. There is no need to repeat the *drivespec* for *file2*.

path The *path* (series of directory names) leading to the file to be renamed.

RENAME
(cont)

file1 The *filename plus extension* of the file being renamed. You may include wildcards in certain cases. See example 2.

file2 The *new filename plus extension* being given to the file specified by the *file1* parameter. You may use wildcards to match corresponding characters in *file1*. See examples 1 and 2.

ERROR MESSAGES

Possible error messages are:

- **Invalid drive specification**
- **Invalid parameter**

SEE ALSO

The COPY command can also be used to create a new copy of any file, with a new name. The original version would still exist, of course.

EXAMPLES

Example 1) – *Wildcarding the new name:*

Entering **REN B:LETTER.TXT *.OLD** gives the file LETTER.TXT on drive B the new name LETTER.OLD. Note that the wildcard in file2 (*.OLD) was used to repeat the filename.

Example 2) – *Renaming a sequence of files:*

Entering **RENAME A:L*.TXT E*.BTY** turns all files on drive A beginning with the letter L and having the filename extension TXT into files beginning with the letter E and having the filename extension BTY.

RESTORE

Restore Files From Diskettes to Hard Disk

ENTERING THE COMMAND

STEP

1) Enter the word **RESTORE**.

2) Enter the parameters, edit the command line, if necessary, and then press **Return**.

COMMAND DESCRIPTION

This command is used to restore files from diskettes to a hard disk. The files must have been placed on those diskettes using the BACKUP command.

COMMAND FORMAT

RESTORE d1: [d2:][path][file][/S][/P]

PARAMETERS

d1: The *drivespec*, or location of the backup diskette.

d2: The *drivespec*, or location of the hard disk to which you want to restore the files.

RESTORE
(cont)

path The *path* defining the current directory. The *path* consists of a list of directory names, separated by backslashes (\), needed to tell DOS the location of the current or specified directory.

file The *filename and extension.* Wildcards may be used, and will result in all matching files being restored.

/S This parameter causes backed up files in all associated *subdirectories* to be restored in addition to the files in the specified directory.

/P This parameter results in a *prompt* being shown on the screen for all files that have been changed since they were last backed up, or that are marked read-only. Files can be defined as read-only by certain application programs that interface with DOS internally. The two DOS system files IO.SYS and MSDOS.COM are marked read-only when they are copied to the hard disk by the FORMAT and SYS.

SEE ALSO

The RESTORE command can only be used for files copied to backup diskettes using the BACKUP command.

RMDIR
Remove a Subdirectory

ENTERING THE COMMAND

STEP

1) Type the command **RMDIR** (or **RD**).

2) Enter the parameters, edit the command line if necessary, then press the **Return** key.

COMMAND DESCRIPTION

The RMDIR command is used to remove (delete) a sub-directory on the specified or default drive. You must remove all files from the directory except the "." and ".." entries. These entries contain information on which directory is the "parent" directory and which directories are "children" of the directory containing the entries. The last directory name in a path is the one that is removed.

You cannot remove the root directory or the current directory.

COMMAND FORMAT

RMDIR [[d:]path] or RD [[d:]path]

RMDIR
(cont)

PARAMETERS

d: The *drivespec* of the drive on which the directory being removed is located. If no *drivespec* is given, DOS assumes the default drive.

path The *path* defining the directory being removed. The *path* consists of a list of directory names, separated by backslashes (\), needed to tell DOS the location of the directory you wish to remove. It is the *last subdirectory* named in the *path* that will be removed.

ERROR MESSAGES:

Possible error messages include:

– **Invalid path, not directory, or directory not empty**

EXAMPLES

Example 1) *– To remove the subdirectory WEEKLY, where the path to* ***WEEKLY*** is ***C:\PETER\REPORTS\WEEKLY:***

The command **RD C:\PETER\REPORTS\WEEKLY** removes the WEEKLY subdirectory on drive C:, where the path to WEEKLY is C:\PETER\REPORTS\WEEKLY (see the examples shown with the CHDIR and MKDIR commands).

SET

Insert Text Strings in the Command Processor Environment

ENTERING THE COMMAND

STEP

1) Enter the word **SET**.

2) Enter the parameters, edit the command line, if necessary, and then press **Return**.

COMMAND DESCRIPTION

This command is used to insert text strings into the command processor environment. These strings, along with all other strings in the system environment, are made available to all commands and applications.

With this command, you can enter keywords and parameters that are not meaningful to DOS but can be found and used by applications that access and examine the environment.

COMMAND FORMAT

SET[name = [parameter]]

PARAMETERS

If no *name* is specified, the current set of environment strings is displayed.

SET
(cont)

If a *name* is specified, but the *parameter* is not specified, the current occurrence of the particular *name=parameter* string is removed from the environment.

ERROR MESSAGES

Possible error messages include:

– **Out of environment space**

If you have loaded a program that is resident in the operating system memory (MODE, PRINT, GRAPHICS, etc.) DOS cannot expand the environment area beyond 127 bytes. If the environment is already greater than 127 bytes, DOS cannot expand it any further.

SORT
Sort Text Files

ENTERING THE COMMAND

STEP

1) Enter the word **SORT**.

2) Enter the parameters, edit the command line, if necessary, and then press **Return**.

COMMAND DESCRIPTION

The SORT command is a filter used to sort each line of text in a file in alphabetical order. Filters are special commands that take information from a standard input device, transform (i.e. "filter") it, and send the output to the standard output device. Information can be sent to the SORT filter via "pipes" or by redirection of the standard input device.

Sorts are done using the ASCII collating sequence. Tab characters are not expanded with blanks. The maximum file size that can be sorted is 63K bytes (about 63,000 characters).

COMMAND FORMAT

SORT [/R][/+n] [<[d:][path][file1]] [>[d:][path][file2]]

SORT
(cont)

PARAMETERS

/R This parameter sorts the files in *reverse* order, from "Z" to A" for example, instead of "A" to "Z".

/+n This parameter specifies that the sort will start with *column n*. If no parameters are specified, the sort starts with column 1.

d: The *Drivespec*. The location of the file to be read and displayed. If no *drivespec* is entered, the current drive is assumed.

path The *path* defining the current directory. The *path* consists of a list of directory names, separated by backslashes (\), needed to tell DOS the location of the current or specified directory.

file1 The *filename and extension* of the file being sorted. Wildcards may not be used.

file2 The *filename and extension* of the file which will receive the sorted data. No wild cards are allowed.

SORT
(cont)

EXAMPLES

Example 1) *– To sort the output of the DIR command alphabetically by file extension:*

Entering **DIR A:|SORT/+9** directs the output of the DIR command (a list of all files on drive A) to the SORT command. The result is:

```
FEB    AP  2345    2-04-84  11:08a
JAN    AP  3455    2-04-84   9:14a
JAN    AR  8743    2-04-84  10:33a
FEB    GL  1245    2-04-84  12:44p
JAN    GL  4256    2-04-84   4:07p
      5 File(s)    234678 bytes free
Directory of A:\
Volume in Drive A has label ACME_ACCTS
```

Example 2) *– To sort lines of information typed directly on the screen:*

Enter the command **SORT** and wait until the cursor moves to the line below the command line. Then type in each line of text, pressing **Return** between each line. After the last line has been typed in (and the **Return** key pressed), press the **Ctrl + Z** key combination and then press Return a final time. DOS then sorts the lines of data and displays them on the screen.

SORT
(cont)

Example 3) – *To sort the contents of a file:*

Entering **SORT<UNSORTED.TXT** causes DOS to take the data in UNSORTED.TXT and sort it by the first character of each line. The results are displayed on the screen, since no other output device is specified.

Example 4) – *To sort the contents of a file and send the results to another file:*

SORT<UNSORTED.TXT >SORTED.TXT sorts the data in UNSORTED.TXT and sends the results to SORTED.TXT. If the space between UNSORTED.TXT and the greater-than sign is omitted, DOS displays an error message.

Example 5) – *Displaying and sorting the contents of a file one screenful at a time:*

Entering **SORT A:CUSTOMER.CAN | MORE** sorts the information in CUSTOMER.CAN and displays it on the screen. If the file contains the names of your Canadian customers, they are displayed on the screen, a screenful at a time.

SYS
Transfer MS-DOS System Files

ENTERING THE COMMAND

STEP

1) Type in **SYS d:** then press **Return**.

(STEPS are continued under USER INTERACTION)

COMMAND DESCRIPTION

The SYS command copies MS-DOS system files from the diskette in the default drive to the specified target drive. These files are hidden, and are not listed by the DIR command.

The two system files, IO.SYS and MSDOS.SYS, must be placed in the first sectors of the disk. If other (non-system) files are already there, SYS cannot copy over them. Thus the target disk must be either completely empty, or else formatted using the /S or /B parameters (/B leaves enough space for the files, and /S copies system files to the initial sectors).

When using SYS to an IBM target diskette, the IBM/BIOS and IBM/DOS files are replaced with IO.SYS and MSDOS.SYS.

COMMAND FORMAT

SYS d:

PARAMETERS

d: The *location* of the *disk drive* onto which you want to copy the system files.

SYS
(cont)

USER INTERACTION

If the system files are not found on the default drive, DOS prompts you:

Insert system disk in drive A
and strike any key when ready

STEP *(cont)*

2) Insert the system diskette in drive A and press any character key. DOS responds:

System transferred

WARNING

Do not try to transfer system files to a diskette in drive A if there are no system files on the default drive. In such cases, DOS prompts you to insert the system disk in drive A (see above) and then attempts to overwrite the system files, destroying them in the process.

ERROR MESSAGE

Error messages include:

– **No room for system on destination disk**

TIME

Display or Modify System Time

ENTERING THE COMMAND

STEP

1) Type in the word **TIME.**

2) Press **Return**.

(STEPS are continued under USER INTERACTION)

COMMAND DESCRIPTION

The Hyperion contains a clock to allow the user to display the current time value being used by the Hyperion, and to reset it.

COMMAND FORMAT

TIME [hh:mm:ss.00]

OPTIONAL PARAMETERS

– Not entering a parameter causes the system to prompt for a new time.

TIME
(cont)

hh:mm:ss.00 The new time, which must be entered the form *hh:mm:ss*. *hh* is the *hour (0-23); mm is the minutes (0-59)*; *ss* is the *seconds (0-59)*, and *00* is *hundredths of a second*. Note that the colons must appear exactly as shown.

It is not necessary to always enter the full time definition. If any portion is entered, the rest is assumed to be zeros. For example, if only hours are entered, the minutes and seconds are set to 00:00.

WARNING

It is very important to keep accurate time and date values. DOS uses them to maintain the last change date and time information in each disk directory. As well, certain programs may use these values in other ways.

USER INTERACTION

If a new time is entered on the command line, the system resets the time and redisplays the system prompt in readiness for the next command.

Pressing the **Return** key without entering a parameter, makes the system display the current time only.

```
Current time is hh:mm:ss
Enter new time:
```

TIME
(cont)

STEP *(cont)*

3) Type in the new correct time, if necessary, and press **Return**.

OR

Press **Return** to select the displayed time.

The system prompt reappears, in preparation for the entry of a new command line.

SPECIAL NOTE

To synchronize the Hyperion clock with an external time, enter a time a few seconds into the future, then press **Return** as soon as the time is reached. The Hyperion clock is reset the moment you press the **Return** key.

ERROR MESSAGES

Possible error messages include:

- **Invalid time**
  Enter new time:

SEE ALSO

The DATE command is used to reset the internal clock for day, month, and year.

TREE
Display all Directories

ENTERING THE COMMAND

STEP

1) Type the word **TREE**.

2) Enter the parameters, edit the command line if necessary, then press the **Return** key.

COMMAND DESCRIPTION

The TREE command is used to display all of the directory paths on the specified drive. If you enter the /**F** parameter, this command can also list all the files in each subdirectory. For each directory found, TREE displays the full path along with any of the subdirectories defined within it.

COMMAND FORMAT

TREE[d:]/F]

TREE
(cont)

PARAMETERS

d: The *drivespec* of the drive on which the subdirectories being listed are located. If no drivespec is given, DOS assumes the default drive.

/F If this parameter is entered, all *files* in the subdirectories being listed will be listed as well.

EXAMPLES

Example 1) – *To display the subdirectories and files on drive A:*

Entering **TREE A: /F** produces the following:

```
DIRECTORY PATH LISTING FOR VOLUME ACME_DISK

Path:          \PETER

Sub-directories: CUSTOMER
                 REPORTS

Files:           MEMO.JBF
                 MEETINGS.JAN

Path:          \PETER\CUSTOMER

Sub-directories: None

Files:           NEWCUST.LST
```

TYPE

Display the Contents of a File

ENTERING THE COMMAND

STEP

1) Press the soft key **F3** (TYPE) from the FILES soft key line or type in the word **TYPE**.

2) Press **Ctrl + Print** to have the text displayed on the screen sent to the printer and printed. This copies everything displayed on the screen to the printer until **Ctrl + Print** is pressed again. (Note: this step is optional, and is used only if you want to print out the file.)

3) Enter a parameter, edit the command line if necessary, then press **Return**.

The system displays (and prints, if step 2 is followed) the file, as specified by the parameters entered, then redisplays the system prompt in preparation for the entry of a new command.

COMMAND DESCRIPTION

This command tells DOS to display the contents of a file on the Hyperion screen. It can also be used to print the file, if **Ctrl + Print** keys are pressed. The action of the **Print** key is described in Part I, Section 11 of this guide.

TYPE
(cont)

Only files that contain text (ASCII coded files) can be properly displayed using this command. The contents of files that contain programs and program generated data (binary files) can not be properly displayed.

COMMAND FORMAT

TYPE [d:][path]file

PARAMETERS

d: The *drivespec*, or disk drive location of the file to be displayed (or printed).

path The *path* (series of directory names) leading to the file.

file The *filename and extension*, if any, of the file to be displayed (or printed).

WARNING

Some files are not legible when displayed using the TYPE command. They contain characters that cannot be displayed on the screen.

TYPE
(cont)

Be prepared to use the **Ctrl + Num Lock** keys to tell the Hyperion to pause when displaying a large file. To recommence the display (or printing) after pressing the **Ctrl + Num Lock** key combination, press any key. To terminate the displaying of a file, press **Ctrl + C**.

ERROR MESSAGES

Possible error messages are:

- Invalid drive specification
- Invalid parameter

SEE ALSO

The **Print** key is described in Part I, Section 11. The PRINT command, described in this section, provides a more convenient way of printing files.

You may wish to use the MORE command as an alternative to the TYPE command in order to display the contents of a file a screenful at a time.

UNLOCK
Unlock a Locked File

ENTERING THE COMMAND

STEP

1) Enter the command **UNLOCK**.

2) Enter the parameters, edit the command line, if necessary, and then press **Return**.

COMMAND DESCRIPTION

This command "unlocks" a file that has been "locked" by the LOCK command to prevent unintentional erasure or change. To erase or modify a "locked" file, you must first "unlock" it using the UNLOCK command.

You do not need to unlock a locked file to copy or edit it. If you edit a locked file, the .BAK version retains the locked feature. Any attempt to delete it, either directly, or by re-editing the copy made of it (still bearing the original filename and extension) is not allowed by DOS. See the LOCK command for details.

COMMAND FORMAT

UNLOCK [d:][path]file

UNLOCK
(cont)

PARAMETERS

d: *Drivespec*. The location of the file to be unlocked. If no *drivespec* is entered, the current drive is assumed.

path The *path* leading to the file being searched. The *path* consists of a list of directory names, separated by backslashes (\).

file The *filename and extension*, if any. Wildcards may not be used.

SEE ALSO

The UNLOCK command is used on files that have been previously locked with the LOCK command.

VER
Display the DOS Version Number

ENTERING THE COMMAND

STEP

1) Enter the command **VER**.

2) Press **Return**.

COMMAND DESCRIPTION

This command is used to display the version number of the DOS version with which you are working. The version number consists of the major version number, followed by a period and a two-digit revision level number. For example:

MS-DOS Version 2.11

where the "2" is the version number and the "11" is the revision level.

COMMAND FORMAT

VER

VERIFY

Verify Data Is Properly Written

ENTERING THE COMMAND

STEP

1) Enter the word **VERIFY**.

2) Enter the parameters, edit the command line, if necessary, and then press **Return**.

COMMAND DESCRIPTION

This command checks that the data being written to a disk (as a result of a COPY command, for example) is being correctly recorded. When set to ON, VERIFY remains on until turned off by a SET VERIFY System Call or a VERIFY OFF command.

If VERIFY is set to ON, disk operations could be slowed down by a noticeable amount in certain cases, since DOS must spend time checking the data when writing to a disk.

COMMAND FORMAT

VERIFY [{ON}{OFF}]

PARAMETERS

Entering the VERIFY command without any parameters results in a message indicating the current state of the VERIFY feature (ON or OFF).

VOL
Display Disk (Volume) Name

ENTERING THE COMMAND

STEP

1) Enter the command **VOL** and a **drivespec**, if desired.

2) Press **Return**.

COMMAND DESCRIPTION

This command displays the volume name (diskname) of the specified disk. The volume name can be set when the diskette is formatted, or, in the case of hard disks, when the DOS partition is created. The volume name can also be set or changed using the DISKNAME command.

COMMAND FORMAT

VOL [d:]

VOL
(cont)

PARAMETERS

d: The volume name of the *disk* located here is displayed. If no *drivespec* is entered, the current drive is assumed.

SEE ALSO

The VOL and DISKNAME commands perform the same function.

Section 3

SUMMARY OF DOS (2.11) COMMANDS

The disk operating system commands described in this Part of the Hyperion DOS 2.11 Guide are:

continued

SUMMARY OF DOS (2.11) COMMANDS (continued)

DOS COMMANDS		PAGE
PATH	Search directories for commands	II-121
PRINT	Print files while doing other tasks	II-123
PROMPT	Set a new system prompt	II-127
RECOVER	Recover files from a defective disk	II-131
RENAME	Rename a file	II-133
RESTORE	Restore files from diskette to hard disk	II-135
RMDIR	Remove a subdirectory	II-137
SET	Insert text strings in the command environment	II-139
SORT	Sort text data	II-141
SYS	Transfer MS-DOS system files.	II-145
TIME	Display or modify system time.	II-147
TREE	Display all directories	II-151
TYPE	Display the contents of a file	II-153
UNLOCK	Unlock a locked file.	II-157
VER	Display the DOS version number	II-159
VERIFY	Verify data is properly written	II-161
VOL	Display disk (volume) name	II-163

Section 1

INTRODUCTION TO PART III

1.1 THE EDLIN TEXT EDITOR

EDLIN is a single-line text editing system which may be used to create files or update existing files. EDLIN may also be used to delete, edit, insert and display single or multiple lines of text.

Text in files created or edited by EDLIN is separated into numbered lines. Each line can be up to 255 characters in length. Line numbers are generated sequentially during the editing process but are not present when saved lines of text are subsequently displayed. When lines are inserted or deleted, line numbers are automatically increased or decreased. Consequently, lines are always numbered consecutively in a file of text.

1.2 TO ACCESS EDLIN FROM DOS

To access EDLIN, enter:

EDLIN [*drivespec*]:<*filename*>.[*extension*]

and press **Return**.

The *drivespec* specified the disk drive where the file is to be located. The *filename.extension* identifies the file. The *drivespec* and *extensions* are optional. The *filename* is mandatory.

If you are creating a new file, the *filename* is that for the new *file* you are creating. EDLIN searches on the *filespec*. If no file is found it creates a new file and displays the message:

```
New File
*
```

If you are accessing an existing file, the *filespec* is that for the existing file you want to display or edit. EDLIN searches and finds the file and loads it into memory. If the whole file can be loaded into memory, EDLIN displays the message:

```
End of input file
*
```

If the file is too large to fit into memory, EDLIN loads lines until 3/4 of memory is full. The asterisk(*) prompt is displayed but not the "End of input file" message. The part of the file now in memory is ready to edit.

To edit the remainder of the file, some of the edited lines must be saved to diskette and the unedited lines loaded from diskette into memory. This procedure is described in the APPEND and WRITE commands in Section 3.

The EDLIN Prompt

The system prompt for EDLIN is an asterisk (*).

The appearance of the asterisk prompt indicates that the system is ready to accept EDLIN commands.

1.3 ENTERING AN EDLIN COMMAND

When the asterisk prompt is displayed, you can enter an EDLIN command and its parameters. The commands and parameters are described in the Sections 2 and 3.

EDLIN commands belong to two types: **intraline** and **interline**. Intraline commands perform editing functions *within* a line. The commands used to perform intraline editing are described in Section 2. Note that some of these are the same commands that are used to edit the DOS command lines.

Interline editing commands are described in Section 3, and perform editing functions on a line-by-line basis.

General Information That Applies to All EDLIN Commands

The following applies to all EDLIN commands:

a) Most commands are either single-letter or single-key commands.

b) With the exception of END and QUIT, commands are usually preceded and/or followed by parameters.

c) Commands can be entered in upper case or lower case letters, or a combination of both.

d) A delimiter (blank space or comma) may be used to separate commands and parameters from each other. Delimiters are not required, however. A delimiter is only required between two adjacent line numbers.

e) Interline commands become effective only after you press the **Return** key.

f) You can stop the execution of any command by pressing **Ctrl + Break**.

g) For commands that produce a constantly scrolling screen display, pressing **Ctrl + Num Lock** will stop the display so that you can read it. Pressing any character key then restarts the display.

...continued

General Information That Applies to All EDLIN Commands (cont)

h) It is possible to refer to line numbers relative to the current line. Use the minus sign (−) and a number to indicate a line before the current line; use a plus sign (+) and a number to indicate a line after the current line.

i) Multiple commands can be entered on one command line (before pressing **Return**). Each command must then be separated from the next by a semicolon.

j) Control characters can be inserted into the text, or can be used in the strings for SEARCH and REPLACE commands. To enter a control character, press **Ctrl + V**, then enter the desired control character in upper case.

Entering Text

To begin entering text, EDLIN must be in the Insert Mode. The insert command, **I**, brings you from the Command Mode to Insert Mode. To exit from insert mode, enter **F6** or the **Ctrl + Z** on a new line.

1.4 EXITING FROM EDLIN BACK INTO DOS

To exit from EDLIN back into DOS, wait until the asterisk prompt reappears, then enter either an **E** or a **Q** and press **Return**. Entering **E** saves the modified file to diskette. Entering **Q** exits from EDLIN without saving the modified file.

Part III

Section 2

INTRALINE COMMANDS

Section 2

INTRALINE COMMANDS

Intraline commands include the special editing functions and the control character functions: only the special editing functions are discussed here.

Table III-A summarizes the commands, codes, and functions. Descriptions of the special editing functions follow the table.

Table III-A
SPECIAL EDITING COMMANDS

COMMAND	CODE	FUNCTION
Copy one character	**F1** or →	Copy one character from the template to the new line.
Copy up to character	**F2**	Copy all characters from the template to the new line up to the character specified.
Copy Template	**F3**	Copy all remaining characters in the template to the new line.
Skip one character	**Del**	Do not copy (skip over) a character in the template.
Skip up to character	**F4**	Do not copy (skip over) the characters in the template up to the character specified.
Quit Input	**Esc**	Void the current input; leaves the template unchanged.
Insert mode	**Ins**	Allows you to insert characters within a line; pressing <Ins> again exits insert mode.
New Template	**F5**	Make the new line the new template.

F1 *(soft key)*
Copy One Character

FUNCTION

Pressing the F1 soft key copies one character from the template (upper line), to the edit line (lower line). When the F1 key is pressed, one character is copied to the edit line and insert mode is automatically turned off. Use the F1 key to advance the cursor one column across the line.

EXAMPLE

Assume you have just displayed a line for editing using the [line] command described on Page III-19. The screen shows:

```
1:*This is a sample file.
1:*_
```

At the beginning of the intraline edit, the cursor is positioned at the beginning of the edit line (indicated by the underline). Pressing the **F1** key copies the first character (T) from the template to the edit line displayed:

```
1:*This is a sample file.
1:*T_
```

Each time the **F1** key is pressed, one more character appears:

```
1:*This is a sample file.
1:*Th_

1:*This is a sample file.
1:*Thi _

1:*This is a sample file.
1:*This_
```

F2 *(soft key)*
Copy Up to Character X

FUNCTION

Pressing the **F2** soft key copies all characters up to a given character from the template to the edit line. The given character is the next character typed and is not copied or shown on the screen. Pressing the **F2** key causes the cursor to move to the single character that is this command's only parameter. If the template does not contain the specified character, nothing is copied. Pressing **F2** also automatically turns off insert mode.

EXAMPLE

Assume the screen shows:

```
1:*This is a sample file.
1:*_
```

At the beginning of the intraline edit, the cursor is positioned at the beginning of the edit line (indicated by the underline). Pressing the **F2** key copies all characters from the template up to the character pressed immediately after the **F2** key. Pressing **F2** first then **p** will show on the screen as:

```
1:*This is a sample file.
1:*This is a sam_
```

F3 *(soft key)*
Copy Remaining Characters

FUNCTION

Pressing the **F3** soft key copies all remaining characters from the template to the edit line. Regardless of the cursor position at the time the **F3** key is pressed, the rest of the line appears, and the cursor is positioned after the last character on the line.

EXAMPLE

Assume the screen shows:

```
1:*This is a sample file.
1:*_
```

At the beginning of the intraline edit, the cursor is positioned at the beginning of the edit line (indicated by the underline). Pressing the **F3** key copies all characters from the template (shown in the upper line displayed) to the line with the cursor (the lower line displayed):

```
1:*This is a sample file.
1:*This is a sample file._
```

Also, insert mode is automatically turned off if it was on.

DEL *(keyboard key)*
Skip One Character

FUNCTION

Pressing the **Del** key skips over one character in the template. Each time you press the **Del** key, one character is deleted (not copied) from the template. The action of the **Del** key is similar to the **F1** soft key, except that **Del** skips a character in the template rather than copies it to the edit line. (Note: to use the **Del** key, the keyboard must *not* be in **Num Lock** mode.)

EXAMPLE

Assume the screen shows:

```
1:*This is a sample file.
1:*_
```

At the beginning of the intraline edit, the cursor is positioned at the beginning of the edit line (indicated by the underline). Pressing the **Del** key skips over the first character ("T")

```
1:*This is a sample file.
1:*_
```

The cursor position does not move, only the template is affected. To see how much of the line has been skipped over, press the **F3** soft key, which copies the remaining characters from the template.

```
1:*This is a sample file.
1:*his is a sample file._
```

F4 *(soft key)*
Skip Up to Character X

FUNCTION

Pressing the **F4** soft key skips over all characters up to a given character in the template. The given character is the next character typed, and is not copied and not shown on the screen. If the template does not contain the specified character, nothing is skipped over. The action of the **F4** key is similar to the **F2** key, except that **F4** skips over characters in the template rather than copies them to the edit line.

EXAMPLE

Assume the screen shows:

```
1:*This is a sample file.
1:*_
```

At the beginning of the intraline edit, the cursor is positioned at the beginning of the edit line (indicated by the underline). Pressing the **F4** key skips over (deletes) all the characters in the template up to the character pressed after the **F4** key. Press **F4** and then the letter **p:**

```
1:*This is a sample file.
1:*_
```

The cursor position does not move. To see how much of the line has been skipped over, press the **F3** key to copy the template. This moves the cursor beyond the last character of the line:

```
1:*This is a sample file.
1:*ple file._
```

All the letters up to the letter **p** have been deleted.

ESC *(keyboard key)*
Stop Input and Clear Edit Line

FUNCTION

Pressing the **Esc** key clears the edit line, but it leaves the template unchanged. **Esc** also prints a back slash (\), carriage return, and line feed, and turns insert mode off if it was on. The cursor is positioned at the beginning of the line. Pressing the **F3** key copies the template to the edit line just as the line was before **Esc** was pressed.

EXAMPLE

Assume the screen shows:

```
1:*This is a sample file.
1:*_
```

At the beginning of the intraline edit, the cursor is positioned at the beginning of the edit line (indicated by the underline). Assume you want to replace the line by typing Sample File:

```
1:*This is a sample file.
1:*Sample File_
```

Now, to re-edit the line, press **Esc**:

```
1:*This is a sample file.
1:*Sample File\
    _
```

Return can now be pressed to keep the original line or to perform any other intraline editing functions. If **F3** is pressed, the original template is copied to the edit line:

```
1:*This is a sample file.
1:*Sample file\
This is a sample file._
```

INS *(keyboard key)*
Insert Mode

FUNCTION

The **Ins** key is a toggle switch which moves from replace mode (the default) to insert mode and back to replace mode when the **Ins** key is pressed a second time. (Note: to use the **Ins** key, the keyboard must *not* be in **Num Lock** mode.)

On entry into insert mode the current position in the template is not changed. The cursor does move as each character is inserted. However, when you have finished inserting characters, the cursor is positioned at the same character as it was before the insertion began. Thus, characters are inserted *before* the character the cursor points to.

Pressing the **Ins** key again causes you to exit from insert mode and enter into replace mode. All characters entered now overstrike and replace characters in the template. When you start to edit a line, this mode is in effect. Each character typed replaces a character in the template. If the **Return** key is pressed, the remainder of the template is truncated.

EXAMPLE

Assume the screen shows:

```
1:*This is a sample file.
1:*_
```

At the start of the intraline edit, the cursor is at the start of the edit line. Press the **F2** key and then **p** key:

```
1:*This is a sample file.
1:*This is a sam_
```

INS
(cont)

Now press the **Ins** key and type the characters "sonite":

```
1:*This is a sam
1:*This is a samsonite_
```

If you now press the **F3** key, the rest of the template is copied to the edit line:

```
1:*This is a samsonite_
1:*This is a samsoniteple file._
```

If you were to press the **Return** key instead of the **F3** key in the above step, the remainder of the template would be truncated, and the edit line would read:

```
1:*This is a samsonite
```

If you type in characters that extend beyond the length of the template, the remaining characters in the template are automatically appended when you type **F3**.

F5 *(soft key)*
New Template

FUNCTION

Pressing the **F5** key copies the current contents of the edit line to the template. The contents of the old template are then destroyed. Pressing **F5** outputs an "at" sign (@), a carriage return, and a line feed. The edit line is also emptied and insert mode is turned off.

NOTE

F5 performs the same functions as the **Esc** key, except that the template is changed and an "at" sign (@) is printed instead of a backslash (\).

EXAMPLE

Assume the screen shows:

```
1:*This is a sample file.
1:*_
```

F5
(cont)

At the beginning of the intraline edit, the cursor is positioned at the beginning of the line (indicated by the underline). Assume that you enter **F2** then the letter **p:**

```
1:*This is a sample file.
1:*This is a sam_
```

Now enter Ins and the character string **sonite**:

```
1:*This is a sample file.
1:*This is a samsonite_
```

Now press the **F3** soft key:

```
1:*This is a sample file.
1:*This is a samsoniteple file.
```

At this point, assume that you want this line as the new template, so you press the **F5** key:

```
1:*This is a sample file.
1:*This is a samsoniteple file.@
_
```

Additional editing can now be done using the above new template.

Part III

Section 3

INTERLINE COMMANDS

Section 3

INTERLINE COMMANDS

Interline commands perform editing functions on whole lines at a time. The interline commands are summarized in the following table and are described in detail with examples following the description of command parameters.

Table III-B
INTERLINE COMMANDS

COMMAND	PURPOSE
<line>	Edit Line
A	Append Lines
C	Copy Lines
D	Delete Lines
E	End Editing
I	Insert Text
L	List Text
M	Move Lines
Q	Quit Editing
R	Replace Text
S	Search Text
T	Transfer Lines
W	Write Lines

Interline Command Parameters

Each interline command accepts some optional parameters. The following list of parameters indicates their form. The effect of a parameter depends on the command with which it is used.

Table III-C
INTERLINE COMMAND PARAMETERS

PARAMETER	DEFINITION
[*line*]	[*line*] indicates a line number to be entered by the user. Line numbers must be separated from other line numbers, other parameters, and the command. Use a comma or space to separate. [*line*] may be specified one of four ways: ***Number*** — Any integer less than 65534. If a *number* larger than the largest existing line number is specified, then <*line*> indicates the line after the last line number. ***Period (.)*** — A *period* indicates the "current" line number. The "current" line is the last line edited, and is marked on your screen by an asterisk (*) between the line number and the first character. ***Octothorpe (#)*** — The *octothorpe* indicates the line following the last line number. Specifying # for <line> has the same effect as specifying a number larger than the last line number. ***Return*** — *Return* entered without any parameter directs EDLIN to use an appropriate default value.
?	The *question mark* parameter directs EDLIN to ask the user if the correct string has been found. The *question mark* is used only with the Replace and Search commands.
[*string*]	[*string*] represents text to be found, to be replaced, or to replace other text. The [*string*] parameter is used only with SEARCH and REPLACE.

Interline Commands

[line]
Edit Line

COMMAND FORMAT

[**line**]

FUNCTION

When a line number is entered, EDLIN displays the line number and text, then, on the line below, reprints the line number. The line is then ready for editing. You may use any of the available intraline commands to edit the line. The existing text of the line serves as the template until the **Return** key is pressed.

If no line number is entered (that is, only the **Return** key is pressed), the line after the current line, marked with an asterisk (*), is edited. If no changes of the current line are needed and the cursor position is at the beginning or end of the line, press the **Return** key to accept the line as is.

WARNING

If the **Return** key is pressed while the cursor is on a character other than the first or last character of the line, the remainder of the line is truncated.

[line]
(cont)

EXAMPLE

Assume the following file exists and has been listed with the L(ist) command:

```
1: This is a sample file
2: used to demonstrate
3: the editing of line
4:*four.
```

To edit line 4, enter:

```
4
```

The contents of the line are displayed along with a cursor below the line:

```
4:*four.
4:*_
```

Now enter **Ins** and enter the word **number**:

```
4:*four.
4:*number_
```

Now press the **F3** soft key and press **Return**:

```
4:*four
4:*number four_
```

A[PPEND]

Append Lines From the Input File

COMMAND FORMAT

[*n*] **A**

where *n* is the number of lines to be appended.

FUNCTION

This command is used for large files that do not fit into memory all at one time. If the command is typed without parameters, lines are appended to memory until available memory is 3/4 full. To edit the remainder of the file that doesn't fit into memory, edited lines in memory must first be written to disk. (See the WRITE command for this procedure.) The rest of the unedited lines in the file may then be loaded into memory using the APPEND command. The message "End of input file", is displayed when the last line of the file has been read into memory using the APPEND command.

EXAMPLE

After editing the first 300 lines of a file that was too large for EDLIN to load, use the WRITE command to write the edited lines onto disk and free up memory so that the APPEND command can be used. Enter: **200 W**. This writes the first 200 lines of your file back to disk. To access the unedited portion of file that you have not yet edited, enter: **100 A**. This loads 100 lines of the unedited portion of your file into memory. If the "End of input file" message did not appear on your screen, there are more lines remaining to be edited on the disk The WRITE-APPEND process must be repeated when you have finished editing the lines you have just loaded.

C[OPY]
Copy Lines

COMMAND FORMAT

[*line1*],[*line2*],*line3* **C**

where *line1* and *line2* define the block of text to be copied. The block of text is copied to a location in front of *line3*.

FUNCTION

The COPY command is used to copy a block of text to a specified point. If the *line1* and *line2* parameters are not specified, they are defaulted to the current line.

The line numbers of the block being copied must not overlap the line defining where the block is to be copied to.

After a COPY command is executed, the first line of the copied block becomes the current line.

EXAMPLE

Assume the following file exists and is ready to edit:

```
1: This is a sample file.
2: Use: to demonstrate dynamic line numbers
3: See what happens when you
4: Use the Copy command.
5: First line to be copied
6: Second line to be copied
7:*Line numbers
```

To copy a block of text, enter: **5,6,3C**

C[OPY]
(cont)

The result is:

```
1: This is a sample file.
2: Use: to demonstrate dynamic line numbers
3:*First line to be copied
4: Second line to be copied
5: See what happens when you
6: Use the Copy command.
7: First line to be copied
8: Second line to be copied
9: Line numbers
```

If the parameters defining the block to be copied were not specified, they would be defaulted to the current line. This copies the current line to the specified location. For example:

```
,,3C
```

When performed on the original sample, the result is:

```
1: This is a sample file.
2: Use: to demonstrate dynamic line numbers
3:*Line numbers
4: See what happens when you
5: Use the Copy command.
6: First line to be copied
7: Second line to be copied
8: Line numbers
```

D[ELETE]

Delete Lines

COMMAND FORMAT

[*line1*][,*line2*] **D**

where *line1* and *line2* are the numbers of lines in the file.

FUNCTION

This command deletes the specified lines and all lines in between. If *line1* is omitted, it defaults to the current line (the line with the asterisk next to the line number). If *line2* is omitted, then just *line1* is deleted. When lines have been deleted, the line immediately after the deleted section becomes the current line and has the same line number as *line1* had before the deletion occurred.

EXAMPLE

Assume the following file exists and is ready to edit:

```
: This is a sample file.
2: Use: to demonstrate dynamic line numbers
3: See what happens when you
4: Delete and Insert
•
•
25: (The D and I commands)
26: (Use <Ctrl + C> to exit insert mode)
27:*Line numbers
```

To delete multiple lines, enter: **5,24 D**

D[ELETE]
(cont)

The result is:

```
1: This is a sample file.
2: Use: to demonstrate dynamic line numbers
3: See what happens when you
4: Delete and Insert
5:*(The D and I commands)
6: (Use <Ctrl + C> to exit insert mode)
7: Line numbers
```

To delete a single line, enter: **2 D**. The result is:

```
1: This is a sample file.
2:*See what happens when you
3: Delete and Insert
4: (The D and I commands)
5: (Use <Ctrl + C> to exit insert mode)
6: Line numbers
```

Next, delete a range of lines from the following file, beginning on the current line (line 2). Enter **,5D**. The result is:

```
1: This is a sample file.
2:*Line numbers
```

Notice how this actually deletes one line less than the actual number specified: ,5D deletes four lines.

E[ND]
End the Editing Session (Save Edited Text)

COMMAND FORMAT

E

FUNCTION

This command ends the editing session, saves the edited file on disk and exits to the DOS command level. The original input file is renamed "filename.BAK".If the file was created during the editing session, no ".BAK" file is created.

The E command takes no parameters. Therefore, you cannot tell EDLIN on which drive to save the file. The drive must be selected when the editing session is invoked. If the drive is not designated when EDLIN is invoked, the file is saved on the disk in the default drive. (It is still possible to COPY the file to a different drive. However, this is done automatically if the drive is designated during access.)

You must be sure that the disk contains enough free space for the entire file to be written. If the process of writing the file to disk is aborted, the version of the file that has just been edited is lost. Only a part of that file is written to disk before the procedure aborts.

EXAMPLE

The only possible command is:

E followed by a **Return**

After execution of the E command, control is returned to DOS and the DOS (drivespec) prompt is displayed.

I[NSERT]

Insert Text Before Specified Line

COMMAND FORMAT

[*line*]**I**

where *line* is the number of the line before which text is to be inserted.

FUNCTION

This command inserts line(s) of text immediately before the specified line. If you are creating a new file, the INSERT command must be given before text can be inserted. In such a case, the inserted lines begin with line number 1.

EDLIN remains in insert mode until a **Ctrl+Z** or F6 is entered (i.e followed by pressing the **Return** key) or **Ctrl + C** combination is pressed. Successive line numbers appear automatically each time **Return** is pressed. When the insert is finished and insert mode has been exited, the line which immediately follows the newly inserted lines becomes the current line. All line numbers following the inserted section are incremented by the number of lines inserted. If a line number is not specified, the default is the current line number (the lines are inserted immediately before the current line). If the line number is an integer larger than the last line number in the file, or if the octothorpe (#) is specified as the line number, the inserted lines are appended to the end of the file. In this case, the last line inserted becomes the current line. (This is the same as when a file is being created.)

I[NSERT]
(cont)

EXAMPLE

Assume that the following file exists and is ready to edit:

```
1: This is a sample file.
2: Use: to demonstrate dynamic line numbers
3: See what happens when you
4: Delete and Insert
5: (The D and I commands)
6: (Use the <Ctrl + C> to exit insert mode)
7:*Line numbers
```

To insert text before a specific line (not the current line), enter:

```
4 I.
```

The result is:

```
4:*_
```

Now, enter the new text for lines 4 and 5:

```
4:*fool around with
5:*those very useful commands that
```

Then, to end the insertion, press **F6** and then press **Return**.

```
6:*<F6>
```

I[NSERT]
(cont)

Now enter the command **L** to list the file:

```
1: This is a sample file.
2: Use: to demonstrate dynamic line numbers
3: See what happens when you
4: fool around with
5: those very useful commands that
6:*Delete and Insert
7: (The D and I commands)
8: (Use the <Ctrl + C> to exit insert mode)
9: Line numbers
```

To insert lines immediately before the current line, enter: **I**. The result is:

```
6:*_
```

Now, insert the following text, press **F6** at the end, and then press **Return**:

```
6:*perform the two major editing functions,
7:*<F6>
```

Then, enter the command **L** to list the file:

```
 1: This is a sample file.
 2: Use: to demonstrate dynamic line numbers
 3: See what happens when you
 4: fool around with
 5: those very useful commands that
 6: perform the two major editing functions,
 7:*Delete and Insert
 8: (The D and I commands)
 9: (Use the <Ctrl + C> to exit insert mode)
10: Line numbers
```

I[NSERT]
(cont)

To append new lines to the end of the file, enter: **11 I.** This produces the following:

```
11:*_
```

Now, enter the following new lines:

```
11:*The insert command can place new lines
12:*anywhere in the file; there's no space problems
13:*because the line numbers are dynamic;
14:*They'll slide all the way to 65533.
```

End insertion by pressing the **Ctrl + C** combination on a new line. The new lines appear at the end of all previous lines in the file. Now list the result:

```
 1: This is a sample file.
 2: Use: to demonstrate dynamic line numbers
 3: See what happens when you
 4: fool around with
 5: those very useful commands that
 6: perform the two major editing functions,
 7: Delete and Insert
 8: (The D and I commands)
 9: (Use the <Ctrl+C> to exit insert mode)
10: Line numbers
11: The insert command can place new lines
12: anywhere in the file; there's no space problems
13: because the line numbers are dynamic;
14: They'll slide all the way to 65533.
```

L[IST]
List a Range of Lines

COMMAND FORMAT

[*line1*][,*line2*] **L**

where *line1* and *line2* are line numbers.

FUNCTION

This command lists the specified range of lines, including the two lines named. If *line1* is omitted, it defaults to line number 1. If *line2* is omitted, 23 lines are listed, starting with *line1*; or, if *line1* is the current line, the eleven lines before the current line and the eleven lines after the current line are displayed. If the current line is one of the lines listed, the list contains an asterisk between the line number and the first character.

EXAMPLE

Assume the following file exists and is ready to edit:

```
 1: This is a sample file.
 2: Use: to demonstrate dynamic line numbers
 3: See what happens when you
 4: Delete and Insert
 5: (The D and I commands)
      .
      .
      .
13:*The current line contains an asterisk.
      .
      .
      .
35: (Alternately, use <Ctrl+C> to exit insert mode
```

L[IST]
(cont)

To list a range of lines without reference to the current line, enter: **2,5 L**. The result is:

```
2: Use: to demonstrate dynamic line numbers
3: See what happens when you
4: Delete and Insert
5: (The D and I commands)
```

To list a range of lines beginning with line number 1, enter: **,5 L**. The result is:

```
1: This is a sample file.
2: Use: to demonstrate dynamic line numbers
3: See what happens when you
4: Delete and Insert
5: (The D and I commands)
```


To list a range of 23 lines following a specified line, enter: **11, L**. The result is:

```
11: The specified line is listed first in the range.
12: The current line is unchanged by the L command
13:*The current line contains an asterisk.
      .
      .
      .
33: <F6> also exits from interline insert.
```

L[IST]
(cont)

To list a range of 23 lines centered around the current line, enter only: **L**. The result is:

```
2: Use: to demonstrate dynamic line numbers
3: See what happens when you
4: Delete and Insert
5: (The D and I commands)
      .
      .
      .
11: The specified line is listed first in the range.
12: The current line is unchanged by the L command

13:*The current line contains an asterisk.
      .
      .
      .
24: <Ctrl + C> exits interline insert command mode.
```

M[OVE]

Move Lines to a New Location

COMMAND FORMAT

[*line1*],[*line2*],*line3* **M**

where *line1* and *line2* define the block of text to be moved. The block of text is moved to a position in front of *line3*. A value for *line3* must be entered.

FUNCTION

This command moves line(s) of text immediately before the line specified by *line3*. If either of the first two parameters *(line1, line2)* are not entered, they default to the current line.

After a MOVE command is executed, the first line of the the block of text moved becomes the current line. The lines are renumbered to reflect their new position.

EXAMPLE

Assume that the following file exists and is ready to edit:

```
1: This is a sample file.
2: Use: to demonstrate dynamic line numbers
3: See what happens when you
4: Move a block of text
5: (The M command)
6: First line of text block
7:*Second line of text block
8: Third line of text block
```

To move a block of text, enter: **6,8,2M**

M[OVE]
(cont)

The result is:

```
1: This is a sample file.
2:*First line of text block
3: Second line of text block
4: Third line of text block
5: Use: to demonstrate dynamic line numbers
6: See what happens when you
7: Move a block of text
8: (The M command)
```

If no values were specified for the *line1* and *line2* parameters, the MOVE command line would look like this when performed on the original example:

```
,,2M
```

The result is:

```
1: This is a sample file.
2:*Second line of text block
3: Use: to demonstrate dynamic line numbers
4: See what happens when you
5: Move a block of text
6: (The M command)
7: First line of text block
8: Third line of text block
```

Q[UIT]

Quit the Editing Session (Do Not Save Edited Text)

COMMAND FORMAT

Q

FUNCTION

This command quits the editing session without saving any editing changes, and exits to the DOS operating system. No ".BAK" file is created. The QUIT command takes no parameters. It is simply a fast means of exiting an editing session. As soon as the Quit command is given, EDLIN displays the message:

```
Abort edit (Y/N)?_
```

Press **Y** to quit the editing session; press **N** (or any other key) if you decide to continue the editing session.

EXAMPLE

Assume the following file exists and is ready to edit:

```
1: This is a sample file.
2: Use: to demonstrate dynamic line numbers
3: Compare the before and after
4: See what happens when you
5: Delete and Insert
6: Line numbers
```

Q[UIT]
(cont)

Now, to delete line 3, enter:

```
3 D
```

To list the file, enter "L":

```
1: This is a sample file.
2: Use: to demonstrate dynamic line numbers
3: See what happens when you
4: Delete and Insert
5: Line numbers
```

Now, to ignore the changes and quit the editing session, enter:

```
Q
```

The system prompts:

```
Abort edit (Y/N)?_
```

Enter **Y** to exit to the operating system command level:

```
Abort edit (Y/N)?Y
```

R[EPLACE]

Replace Text

COMMAND FORMAT

[*line1*][,*line2*][?]**R**<*string1*>[*F6*]<*string2*>]

where *line1* and *line2* are line numbers; *F6* is a soft key; *string1* and *string2* are text strings; and *?* generates a prompt.

FUNCTION

This command replaces all occurrences of *string1* in the specified range (from line1 to line2) with *string2*. As each occurrence of *string1* is found, it is replaced by *string2*. Each line in which a replacement occurs is displayed. If a line contains two or more replacements of *string1* with *string2*, then the line is displayed once for each occurrence. When all occurrences of *string1* in the specified range are replaced by *string2*, the REPLACE command terminates and the asterisk prompt reappears.

If *string2* is omitted, then *string1* is deleted from all lines in the range. If *string1* is simply to be deleted, then *string1* may be terminated with either a combination **F6 Return**, or simply a **Return**. If *string2* is used as a replacement for *string1*, it may be terminated either with a **F6 Return** combination or with a **Return**.

If *line1* is omitted, then *line1* defaults to the line after the current line. If *line2* is omitted, *line2* defaults to an octothorpe (#). Remember that the octothorpe (#) indicates the line after the last line of the file.

R[EPLACE]
(cont)

If the question mark (?) parameter is given, the REPLACE command stops at each line with a string that matches *string1*, displays the line with *string2* in place, and prompts "**O.K.?**". If the user enters **Y** or the **Return** key, then *string2* replaces *string1*, and the next occurrence of *string1* is found. Again, the "**O.K.?**" prompt is displayed. This continues until the end of the range or until the end of the file. After the last occurrence of *string1*, EDLIN returns the asterisk prompt.

If you press any key besides **Y** or **Return** after the "O.K.?" prompt, the *string1* is left as it was in the line, and the replace goes to the next occurrence of *string1*. If *string1* occurs more than once in a line, each occurrence of *string1* is replaced individually, and the "**O.K.?**" prompt is displayed after each replacement. In this way, only the desired *string1* is replaced, and you prevent replacement of embedded strings.

EXAMPLE

Assume the following file exists and is ready to edit:

```
 1: This is a sample file.
 2: Use: to demonstrate dynamic line numbers
 3: See what happens when you
 4: fool around with
 5: those very useful commands that
 6: perform the two major editing functions,
 7: Delete and Insert
 8: (The D and I commands)
 9: (Use <Ctrl+C> to exit insert mode)
10: Line numbers
11: The insert command can place new lines
12: anywhere in the file; there's no space problems
13: because the line numbers are dynamic;
14: They'll slide all the way to 65533
```

R[EPLACE]
(cont)

To replace all occurrences of *string1* with *string2* in a specific range, type in:

```
2,12Rand
```

then press **F6** and type in "**or**". Your command line should be:

```
2,12Rand ^Zor
```

Now press **Return**. The result is:

```
 5: those very useful commors that
 7: Delete or Insert
 8: (The D or I commors)
11: The insert commor can place new lines
```


If you want to change the lines back to the correct version (i.e. with "command" for "commor"),a global replace of "or" would also change the occurrences of the word "or" as in line 8: "the D or I commors". To avoid this, and confirm each replacement, use the question mark (**?**) parameter. First type in:

```
2,12?Ror
```

then press **F6** and type in "**and**". Your command line should be:

```
2,12?Ror ^Zand
```

R[EPLACE]
(cont)

Now press **Return**. The result is:

```
5: those very useful commands that
O.K.? Y
8: (The D and I commors)
O.K.? N
11: The insert command can place new lines
O.K.? Y
*_
```

Now, use the LIST command (**L**) to see the result of all these changes:

```
      .
      .
 5: those very useful commands that
      .
 7: Delete and Insert
 8: (The D or I commands)
      .
11: The insert command can place new lines
      .
      .
```

S[EARCH]
Find Text

COMMAND FORMAT

[*line1*][,*line2*][?]**S**[*string*]

where *line1* and *line2* are line numbers; *string* is the text string being searched for; and ? generates a prompt.

FUNCTION

This command searches the specified range of lines for the specified string. The text string must be terminated with a **Return**. The first line that matches *string* is displayed, and becomes the current line. The search command terminates when a match is found. If no line contains a match for *string*, the message "**Not Found**" is displayed.

If the optional parameter, question mark (?) is included in the command, EDLIN displays the first line with a matched string; then prompts the user with the message "**O.K.?**". If the user enters **Y** or **Return**, the line becomes the current line, and the search terminates. If the user presses any other key, the search continues until another match is found, or until all lines have been searched (the "**Not Found**" message is displayed).

If *line1* is omitted, *line1* defaults to the line after the current line. If *line2* is omitted, *line2* defaults to the octothorpe (#). If *string* is omitted, the string used with the previous S[EARCH] or R[EPLACE] command is used. If no previous string has been used in this session, the message "**Not Found**" is displayed.

S[EARCH]
(cont)

EXAMPLE

Assume the following file exists and is ready to edit:

```
 1: This is a sample file.
 2: Use: to demonstrate dynamic line numbers
 3: See what happens when you
 4: fool around with
 5: those very useful commands that
 6: perform the two major editing functions,
 7: Delete and Insert
 8: (The D and I commands)
 9: (Use <Ctrl+C> to exit insert mode)
10: Line numbers
11: The insert command can place new lines
12: anywhere in the file; there's no space
    problems
13: because the line numbers are dynamic
14:*They'll slide all the way to 65533
```

To search for the first occurrence of a string, enter:

```
2,12 Sand
```

and press **Return**. The result is:

```
5: those very useful commands that
```

To find the "and" in line 7, modify the search command by pressing the **Del** key, the **F3** soft key and then the **Return** key. The command entered is:

```
,12Sand
```

S[EARCH]
(cont)

The search then continues from the line after the current line (line 5), since no first line is given. The result is:

```
7: Delete and Insert
```

To search through several occurrences of a string until the correct string is found, enter:

```
1, ? Sand
```

The result is:

```
5: those very useful commands that
O.K.?N_
7: Delete and Insert
O.K.?Y_
*_
```

When you find the occurrence you are looking for, press **Y** to stop the search command. To continue searching, press **N**.

T[RANSFER]

Transfer Text from a File to Current File

COMMAND FORMAT

[*line*]**T**[*d:*][*file*]

where *d:* is the disk drive location of the file, and *file* is the *filename* and *extension* of the file being transferred to the current file. The text is added to a location in front of the specified *line*.

FUNCTION

The TRANSFER command is used to copy the contents of another file to the current file you are working on. The *file* being transferred must be in the same directory as the current file (i.e. as defined by the path used in the EDLIN command line). If a *file* is requested from a disk drive other than the default drive, it must reside in the root directory.

If the TRANSFER command is used to request text from a file in a different directory, EDLIN responds with the message:

File not found

If the *line* parameter is not specified, *line* defaults to the current line.

When text is transferred into the current file, the first line of the new text becomes the current line.

EXAMPLES

Assume that the following file, called EXTERNAL.FIL exists on the same disk drive and in the same directory as the current file you're working on:

T[RANSFER]
(cont)

```
1: First line of external file
2: Second line of file
     .
     .
10: Last line of file.
```

Assume as well that you're currently working on the following file:

```
1: First line of current file
2: which is an exercise showing
3: how to use the Transfer command
4: We'll add the other file in here
     .
     .
25:*Last line of current file
```

To copy an external file that resides on drive B (not the current drive), enter: **4,T,B:EXTERNAL.FIL**

The result is:

```
1: First line of current file
2: which is an exercise showing
3: how to use the Transfer command
4:*First line of external file
5: Second line of file
     .
     .
13: Last line of file.
14: We'll add the other file in here
     .
     .
35: Last line of current file
```

W[RITE]
Write to Output File

COMMAND FORMAT

[*n*] **W**

where *n* is the number of lines to be written to diskette.

FUNCTION

The WRITE command is used when editing files that are larger than available memory. By executing the WRITE command, lines are written from the edited file in memory to diskette. Space is created in memory for lines to be appended. If *W* is entered with no *n* parameter, then lines are written until memory is 1/4 full.

If the *n* parameter is given, then *n* lines are written out. Note that lines are written out beginning with the start of the file; subsequent lines in the editing buffer are renumbered beginning with one. A later APPEND command will append lines to any remaining lines in the editing buffer.

Part III

Section 4

EDLIN ERROR MESSAGES

Section 4

EDLIN ERROR MESSAGES

4.1 ERRORS WHEN ACCESSING EDLIN

The following errors can occur when accessing EDLIN from DOS:

- **Cannot edit .BAK file—rename file**

  The user tried to edit a file with the filename extension ".BAK." ".BAK" files cannot be edited as this extension is reserved for backup copies.

  If the user needs the ".BAK" file for editing purposes, the user must either RENAME or COPY the file, giving it a different extension.

- **Invalid drive or file name**

  Check the EDLIN access command line for illegal filename or illegal disk drive specifications.

4.2 ERRORS WHILE EDITING

Errors that can be incurred during an EDLIN editing session are:

- **Entry Error**

  The last command entered contained a syntax error. Re-enter the command with the correct syntax.

- **Line too long**

  If the strings used with the REPLACE commands cause the line to expand beyond 253 characters, EDLIN aborts the command.

  Use the command to replace only part of the text string, and then replace the rest of the text with a subsequent REPLACE command.

- **Disk Full—file write not completed**

  The user entered the END command, but the disk did not contain enough free space for the whole file. EDLIN aborts the END command and returns the user to the operating system. Some of the file may have been written to the disk.

  Only a portion of the file has been saved. Any part of the file not written out to the disk has been permanently lost. You should probably delete the portion of the file that was saved and restart the editing session. Use the version of the file with the .BAK extension (i.e. if the file was OVERFLOW.TXT, use the backup version, OVERFLOW.BAK. You must rename it with an extension other than .BAK to use it, however. Always be sure that the disk has sufficient free space for the file to be written, before you begin your editing session.

- **File not found**

  A file has been specified with the TRANSFER command that does not exist, or that doesn't reside in the same directory as the file being edited. Exit from the file you're working on, and check the filename and extension using the DIR command in DOS.

Section 5

SUMMARY OF EDLIN COMMANDS

The EDLIN commands available on the Hyperion are:

COMMAND/KEY		PAGE
A[PPEND]	Append lines from the input file to the editing buffer.	III-21
C[OPY]	Copy lines of text.	III-22
DEL	Skip a character in the template.	III-9
D[ELETE]	Delete lines from the editing buffer	III-24
E[ND]	End the editing session and save the changes to the file.	III-26
ESC	Clear the input line, but leave the template unchanged.	III-11
F1 or →	Copy one character from the template to the input line.	III-6
F2	Copy all characters up to "x" from the template to the input line.	III-7
F3	Copy all remaining characters from the template to the input line.	III-8
F4	Skip up to character "x" in the template.	III-10
F5	Replace the contents of the template with the contents of the input line.	III-14
INS	Insert characters in the input line without affecting the template.	III-12
I[NSERT]	Insert new text lines into the editing buffer.	III-27
[line]	Displays the line indicated.	III-19
L[IST]	List lines from the editing buffer	III-31
M[OVE]	Move lines to a new location	III-34
Q[UIT]	End the editing session and discard the changes to the file.	III-36
R[EPLACE]	Replace one text string with another.	III-38
S[EARCH]	Find a text string.	III-42
T[RANSFER]	Transfer text from a file to current file	III-45
W[RITE]	Write text lines from the editing buffer to the output file.	III-47

Section 1

INTRODUCTION TO PART IV

DOS, the disk operating system, provides an environment from which many other programs can be loaded into memory. Those programs found to be the most useful have been included on the DOS system diskette and are described in this part of the DOS guide.

- ***Section 2*** describes **LINK**, a program which joins specially-formatted programs or program segments together. The general LINK options are described. For those options that are specific to BASIC, FORTRAN, Pascal, Macro Assembler, and other programming languages, however, you should consult the applicable programmer manual for the specific language.

- ***Section 3*** describes **DEBUG**. This is an interactive machine language debugging program, permitting a user to load, examine, and alter a program's machine language. Any disk or memory location may be accessed, and all memory locations except ROM may be altered.

- ***Section 4*** describes **LIB**, a program that enables the user to store subroutines in a library that can be accessed by more than one program as required.

Part IV

Section 2

LINK

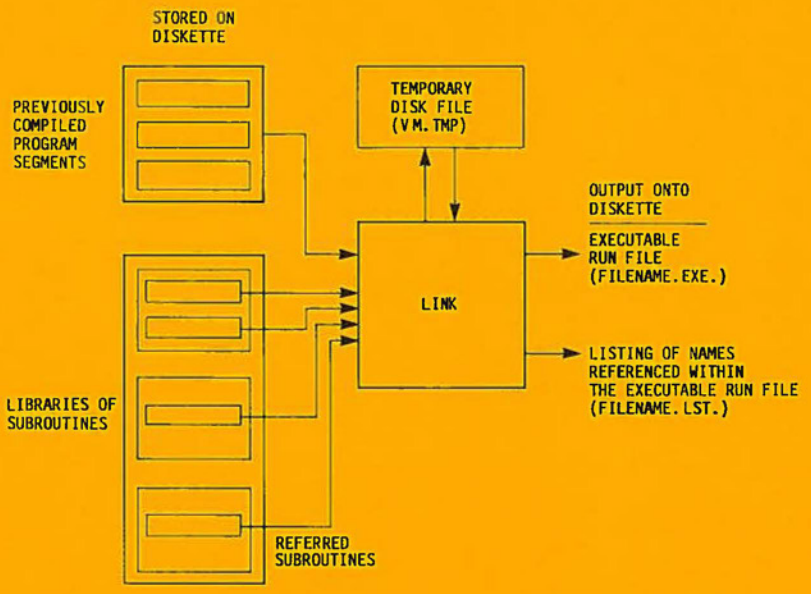


Fig. IV-1 – A block diagram of the LINK program function.

Section 2

LINK

2.1 WHAT IS LINK

In most high-level programming languages, as well as in Assembler, a text editor such as EDLIN is used to enter the instructions and data, referred to as "source code". Source code can be either a single block of coding, or it can be a group of modules, used as a single program. Using a modular approach creates smaller, more easily-assembled programs, as it permits the use of a single module in a number of different programs.

LINK also allows you to "mix" languages because of its operation at machine-language level, permitting you to hook an Assembler language program up to run with a module written in a another language such as BASIC.

The program assembler creates an intermediate file which is then used by the LINK program to generate two new files: one is machine executable or "object" code, which is referred to as the *run file*; the other is a *list file* listing the results of creating the object code. The list file also contains any reference labels and subroutine names, cross-referenced to their appropriate destinations in a program.

LINK provides a system work area for preparing object code for entry into a core-image library. The object program will have its specific base address, assigned by the programmer. This may, however, conflict with the supervisor (the primary control program of the operating system), and a new base address is then assigned by LINK.

LINK performs other valuable services, including: creating temporary workspace, searching libraries for required program modules, and loading data in the proper sequence.

Temporary workspace can be created by LINK in the form of a VM.TMP file (a temporary virtual memory file). As the name indicates, this file contains addressable space that *appears* to be real storage, but is not. The file exists for the LINK session only, and is deleted at the end of the session.

Relocatable libraries can be set up by the programmer to store commonly-used routines. When these routines are required, naming the particular library to LINK by means of a "libswitch" will cause the contents of that library to be loaded into an available main storage location, hence the term "relocatable" library.

LINK also loads the program to be executed and any required modules into memory locations according to a specific heirarchy. LINK also generates error messages, permitting the programmer to detect and correct errors and to rerun LINK.

2.2 WHEN TO USE LINK

LINK is a program produced by Microsoft that can link previously compiled programs together into one that can be run on the computer.

In order to use LINK, your computer must have at least 50 K bytes of memory space free: 40 K to store the program code and data; 10 K to be available for run space.

LINK is able to link program files whose combined size is less than or equal to 384 K bytes.

Single-Diskette Drive Considerations

A single-diskette drive Hyperion can be used if and only if the output from LINK is sent to the same drive from which the input is taken. LINK does not allow time to swap diskettes during operation on a one-drive Hyperion. Therefore, two diskette drives provides a more practical configuration.

Brief Description

The LINK program must be used in order to create a program that can be run on the Hyperion. You would use LINK when you have one or more programs that you have coded in a language such as BASIC, Assembler, Pascal, C, etc., and which you have compiled (i.e., translated into its Assembler equivalent) and which you wish to combine into one executable program. LINK will combine these subprograms into one program, change the references where necessary to make sure that all references direct the pointer to the correct destination, translate the Assembler into executable machine code, and output the result into a file called *filename*.EXE.

A block diagram of the LINK program function is given in Fig. IV-1.

2.3 HOW TO USE LINK

To use LINK, you would:

STEP

1) Insert the diskette containing the file LINK.EXE. This file contains the LINK program.
2) Direct the attention of the operating system to the drive containing the diskette by entering the drive letter followed by a colon and then pressing the Return key.

...continued

STEP *(cont)*

3) Enter the word **LINK**, press **Return**, and respond to the system prompts as described in Section 2.7.1, "Method 1: Interactive LINK".

OR

Enter the word **LINK**, followed by the appropriate parameters, i.e. LINK, C+TEST, TEST, TEST/MAP, etc. Edit the command line as required, and then press **Return**. The procedure is described in Section 2.7.2, "Method 2: Command Line LINK".

OR

Enter the word **LINK**, followed by a path and filespec, i.e. LINK @\BANK\R.RSP, and then press **Return**. The filespec is the filename and extension of the file that contains the parameters to the LINK command. The procedure is described in Section 2.7.3, "Method 3: Batched LINK".

As indicated, the LINK command can be entered in one of three ways. Each is described in detail in Section 2.7. Command parameters and switches are described in Section 2.8.

2.4 WHAT LINK DOES – DESCRIPTION OF THE LINK PROGRAM

The LINK program scans the object code contained in the files it has been requested to link and loads this code into memory in the following way·

a) The object code has been defined, either by the programmer (in Assembler), or by the higher level program (BASIC, Pascal, etc.) compiler, into ***segments.*** Each segment has a ***name*** and a ***combine type***. Each segment may further have a ***class***(*ification*) name and/or a ***group*** name. Each segment of code may not exceed 64 K bytes in size.

b) LINK loads segments into memory, alphabetically by segment name, according to class. It then reorders classes according to the first segment name of each class.

c) LINK then scans the object code in memory, checking for segments that belong to the same group.

d) The coded statements in the segments being linked are readdressed starting from offset 0000H for the beginning of the first segment loaded. The addresses of labels that are referenced from within segments are renumbered, if necessary. The addressing depends on the combine type.

e) LINK can also be directed to look at compiled code that was previously stored in "library" files, loading portions of these libraries that were referenced from the segments of object code being linked together, into the program being LINKed. The library search is done via a directory-indexed library search method, which greatly reduces the link time for sessions involving library searches.

f) LINK uses available memory as much as possible. When available memory is exhausted, LINK creates a temporary file called VM.TMP, and puts it onto the diskette in the default drive. The linking process continues, with the segments now being loaded into this VM.TMP file on diskette. When linking is finished, the contents of VM.TMP and memory are both transferred into the run file *filename*.EXE, which contains the executable program.

LINK

OBJECT CODE

SEGMENT NAME	SEGMENT CLASS
MAGGIE	APPLE
PETR	KEEP
MINOU	HOPS
HAPPY	QUILT
HYPERION	ZEBE
STEP	ANDREW
PINEAPPLE	JUICE
SANDWICH	BEEF
TOP	UPSIDE
Z	A
DISKS	BLACK
SPOON	YOG
BOOK	KEEP
LITTLE	BEEF
COW	JUICE
FARM	KEEP
LAND	UPSIDE
MEADOW	BEEF
X	BLACK
B	YOG
C	BEEF
Y	YOG
W	JUICE
V	HOPS
D	QUILT
K	ZEBE
J	QUILT
M	ANDREW
CANADA	ANDREW
IRA	A

L I N K

EXECUTABLE CODE

SEGMENT NAME	SEGMENT CLASS
B	YOG
SPOON	YOG
Y	YOG
BOOK	KEEP
FARM	KEEP
PETER	KEEP
C	BEEF
LITTLE	BEEF
MEADOW	BEEF
SANDWICH	BEEF
CANADA	ANDREW
M	ANDREW
STEP	ANDREW
COW	JUICE
PINEAPPLE	JUICE
W	JUICE
D	QUILT
HAPPY	QUILT
J	QUILT
DISKS	BLACK
X	BLACK
HYPERION	ZEBE
K	ZEBE
IRA	A
Z	A
LAND	UPSIDE
TOP	UPSIDE
MAGGIE	APPLE
MINOU	HOPS
V	HOPS

Fig. IV-2 – LINK loads segments from the object code programs in the order of their class.

2.5 CERTAIN DEFINITIONS

The terms mentioned in the previous subsection are: segment, name, combine type, class, and group. type. In addition, there are three combine types: private, public and stack, and common. These terms summarize the underlying functioning of LINK. An understanding of the concepts that define these terms provides a basic understanding of the way LINK works.

Segment

A segment is a series of coded instructions of up to 64 K bytes in length. A segment must have a name, and may also be assigned a class(ification) name, a group name, and a combine type. When loaded into memory, the segment must be so coded as to begin on a "paragraph" (16-byte boundary). The coding of segments and other coded instructions are described, for the Hyperion, in the *Hyperion BASIC Guide* (for BASIC), and in the appropriate programmer guides for other programming languages.

Name

The name of the segment can be up to 30 characters long. However, if the name is also used as a filename, it must conform to normal DOS rules for filenames.

Combine Types

In addition to a class name and a group name, a segment may also be assigned a combine type. There are three possible combine types: private, public and stack, and common.

If the combine type of a segment is *private*, it means that no coded instructions in that segment reference anything outside that segment. The segment exists as an independent program. LINK therefore loads it (and the executable program) into memory like an independent program. The beginning address offset is always 0000H, and the offset of the segment that follows is returned to 0000H.

If the combine type of a segment is *public and stack*, then there are coded instructions in the segment that reference variables, data, and coded instructions in other public segments or in the stack segment. Consequently, there is only one base address for public segments with the same segment name and class name.

If the combine type is *common*, then that indicates that the object code of the smaller segments is exactly the same as the same amount of object code in the larger segments. As a result, the segments are all loaded on top of one another in memory. The resulting memory area taken up will only be as long as the longest segment loaded. There is only one base address for all common segments of the same name and class.

Class

Assigning a class(ification) name to a segment controls the order and relative placement of segments in memory during LINKing time. In Assembler, the class name is entered by the programmer. In higher-level languages (BASIC, FORTRAN, COBOL, etc.), class naming is done by the compiler.

When called up, the LINK program scans the object code files in the order they are entered on the LINK command line. LINK places the first segment of code found into memory, then searches for other segments with the same class name, loading them into memory as they are found.

When all segments from one class are loaded, LINK returns to the beginning of the object code files to load the first segment (of the next class). The process continues until all code has been loaded. The executable program segments will thus end up being "gathered" into classes. Good gathering minimizes execution time and makes for a better program. Fig. IV-2 attempts to illustrate this process.

If the combine type of a segment is *private*, it means that no coded instructions in that segment reference anything outside that segment.

If you are writing Assembler programs, you can control the ordering of classes by writing a dummy module at the beginning of the first program to be linked. This dummy module will consist of the series of segment instructions, each assigned to a class. LINK will then organize the segments into the classes in the same order as in the dummy module.

WARNING: This method will work only when coding in Assembler. Do not use this method with higher-level languages. You should also be careful to declare all classes to be used in your program, if you want to retain absolute control over the ordering of classes.

Group

A segment may also be assigned a group name. Note that segments belonging to the same group may be scattered throughout the program. They need not be contiguous.

Once all the segments are loaded into memory, LINK scans the resulting program. The limitation measured by LINK is the distance from the beginning of the first segment of any group, to the end of the last segment of that group. This distance must be less than 64 K bytes. This limitation includes any segments belonging to other groups if those segments are interspersed between the first and last segments of the first group.

For example, if the segment order is "groups ABCCDBEAABCFF", then ABCCDBEAA must be less than 64 K bytes long, BCCDBEAAB must be less than 64 K bytes long, and CCDBEAABC must be less than 64 K bytes long. If these conditions are not met, then LINK will display an error message and not produce an executable program.

The purpose and advantage of assigning group names to segments would be determined by the programmer.

Again, a user may manually assign group names to segments in Assembler only. Higher level languages have group and class names assigned automatically by their own respective compilers.

2.6 FILES USED BY AND PRODUCED BY LINK

LINK works with one or more input files, produces two output files, may create a virtual memory file, and may be directed to search one to eight library files. For each type of file, the user may give a three-part file specification. The format for LINK file specifications is:

DRIVE:\PATH\FILENAME.EXTENSION

where: ***DRIVE:*** is the *drive designation*. Permissible drive designations for LINK are A: through O:. The colon is always required as part of the drive designation.

PATH is the *series of directory names* leading to the file.

FILENAME is any legal *filename* of one to eight characters.

EXTENSION is a *one- to three-character extension* to the filename. A period is always required as part of the extension.

The file specification must be entered as one word, with no blank spaces between parts. An example would be:

```
B:MAIN.OBJ
```

Input Files

If no extensions are given in the input (object) file specifications, LINK recognizes by default:

INPUT FILE	DEFAULT EXTENSION
Object	.OBJ
Library	.LIB

Output Files

If no extensions are given in the output (run or list) files, LINK appends the following default extensions:

OUTPUT FILE	DEFAULT EXTENSION
Run	.EXE (you cannot override extension)
List	.MAP (you can override extension)

VM.TMP File

LINK uses available memory for the link session. If the files to be linked create an output file that exceeds available memory, LINK creates a temporary file and names it VM.TMP. If LINK needs to create VM.TMP, it displays the message:

```
VM.TMP has been created
Do not change diskette in drive, x
```

Once this message is displayed, the user must not remove the diskette from the default drive until the link session ends. If the diskette is removed, the operation of LINK is unpredictable, and LINK might return the error message:

Unexpected end of file on VM.TMP

LINK uses VM.TMP as a virtual memory. The contents of VM.TMP are subsequently written to the file named following the Run File: prompt. VM.TMP is a working file only and is deleted at the end of the linking session.

WARNING: Do not use VM.TMP as a file name for any file. If the user has a file named VM.TMP on the default drive and LINK requires the VM.TMP file, LINK will delete the VM.TMP on disk and create a new VM.TMP. Thus, the contents of the previous VM.TMP file will be lost.

2.7 THREE METHODS OF ENTERING THE LINK COMMAND

LINK may be invoked three ways:

1) ***Method 1: Interactive LINK***. By this method, the user enters the word LINK and waits for the system to prompt for individual parameters.

2) ***Method 2: Command Line LINK***. By the second method, the user enters the word LINK and all necessary parameters on the same command line.

3) ***Method 3: Batched LINK***. By this third method, the user enters the word **LINK** followed by a filespec. In this last method, the user must first create a file (specified by the *filespec*) to contain all the necessary parameters.

In all three cases, in addition to the command word **LINK**, the user must be familiar with the *parameters* and *switches* that modify the function of the LINK command. Parameters and switches are described in Section 2.8.

2.7.1 Method 1: Interactive LINK

STEP

1) Enter the word **LINK**, and press **Return**.

LINK will be loaded into memory. Then, LINK returns a series of four text prompts that appear one at a time. The first prompt is:

Object Modules [.OBJ]:

Remember to specify the drivespec (location of the files to be linked) and path, unless the files are on the current drive and in the current directory.

STEP

2) Enter the .OBJ files to be linked, separated by a blank space or plus sign (+).

   Entering a plus sign and pressing **Return**, causes the system to redisplay the prompt and enables you to continue entering .OBJ files to be linked. There is no default. At least one object filename must be entered.

...continued

STEP *(cont)*

3) Type in any switches that are required. Each switch must be preceeded by a slash (/).

4) Use the cursor control and keyboard keys to edit the characters entered, if necessary (this step is the same for any user input, and will therefore not be restated).

5) Press **Return**.

   The system displays the next prompt:

   Run File [Object-file.EXE]

6) Type in the filename (or a filename plus extension) for the file to contain the executable machine code for the program being linked.

   If no name is entered, the system uses the filename of the first object file entered in Step 2. If no filename extension is entered, the system automatically appends the extension .EXE.

7) Type in applicable switches and press **Return**.

   The system displays the next prompt:

   List File [Run-File.MAP]:

8) Enter a filename for the file to contain a listing of the names of all routines and labels referenced within the linked program.

   If no filename is entered, the system assumed no .MAP file is to be created.

...continued

The system displays the next prompt:

STEP *(cont)*

9) Type in a filename for the file to contain a listing of the names of all routines and labels referenced within the linked program.

   If no filename is entered, the system assumed no .MAP file is to be created.

10) Type in any switches that may apply and press **Return**.

    The last prompt is displayed:

    Libraries [.LIB]:

11) In order to locate required modules, the user must specify the libraries where these are located. To do this, enter the filenames of the library files (up to 8) to be searched. Separate each filename from the next by either a blank space or a plus sign (+).

    If you enter a plus sign followed by a **Return**, the system redisplays the prompt to allow you to continue entering library filenames.

12) Type in any switches that may apply and press **Return**.

    If no filenames are entered before pressing **Return**, the system does not search any.

The LINK program will then begin the linking procedure. This may take from a few seconds to a few minutes, depending on the size of the programs being linked.

A sample entry-response session is shown in Fig. IV-3.

Variations

If the response to the first prompt is terminated by a semicolon (;) before pressing **Return**, then the LINK program is invoked immediately. The remaining three prompts are not displayed. The system assumes the default values. This feature saves time and overrides the need to enter a series of carriage returns.

To stop the LINK session at any time, press **Ctrl + Break.**


```
A>link

   Microsoft Object Linker V2.01 (Large)
(C) Copyright 1982,1983 by Microsoft Inc.

Object Modules [.OBJ]: RIPON
Run File [RIPON.EXE]:
List File [NUL.MAP]:
Libraries [.LIB]:

LASTLN  Disks  Files  MODE  DIR/P  00:53  CHDIR  PATH  TREE  XPLAIN  HELP
```

Fig. IV-3 – A sample entry-response session displayed on the screen.

2.7.2 Method 2: Command Line LINK

STEPS

1) Enter the following command line:

   LINK <**object list**>[/**switch(es)**],
   [**run file**][/**switch(es)**],
   [**list file**][/**switch(es)**],
   [**library list**][/**switch(es)**]

2) Edit the command line, if necessary, and press the **Return** key.

The parameter enclosed in angle brackets (<,>) must be entered. The parameters enclosed in square brackets ([,]) are optional. The parameters are:

- ***object list*** is a *list of object modules*, separated by plus signs. If no filename extensions are specified, the system assumes a filename extension of .OBJ.

- ***run file*** is the *name of the file* to receive the executable output. If no filename extension is specified, the system appends the extension .EXE.

- ***list file*** is the *name of the file* to receive the listing.

- ***library list*** is a *list of library modules* to be searched. If no filename extensions are specified, the system assumes the extension is .LIB.

- ***/switch(es)*** are *optional switches*, which may be placed following any of the response entries (just before any of the commas or after the <lib-list>, as shown).

The parameters and switches are described in detail in Section 2.8.

To select the default for a field, simply enter a second command without spaces in between (see the example in Fig. IV-4).

The example shown in Fig. IV-4 causes LINK to be loaded, then causes the object modules FUN.OBJ, TEXT.OBJ, TABLE.OBJ, and CARE.OBJ to be loaded. LINK then pauses (caused by the /P switch). When the user presses any key, LINK links the object modules, produces a global symbol map (the /M switch), defaults to FUN.EXE run file, creates a list file named FUNLIST.MAP, and searches the library file COBLIB.LIB.


```
A>LINK FUN TEXT TABLE CARE /P/M,,FUNLIST,COBLIB

LASTLN Disks Files MODE DIR/P 00:09 CHDIR PATH TREE XPLAIN HELP
```

Fig. IV-4 – An example of a LINK command line entered on screeen.

2.7.3 Method 3: Batched LINK

STEP

1) Enter the command line:

LINK @<filespec>

2) Edit the command line, if necessary, and press the **Return** key.

The parameter *filespec* is the name of a response file. A response file contains answers to the LINK prompts (shown under method 1 for invoking) and may also contain any of the switches.

IMPORTANT: Before using method 3 to invoke LINK, the user must first create the response file.

A response file has text lines, one for each prompt. Responses must appear in the same order as the command prompts appear.

Use switches and special command characters in the response file the same way as they are used for responses entered on the terminal keyboard.

When the LINK session begins, each prompt will be displayed in turn with the responses from the response file. If the response file does not contain answers for all the prompts, either in the form of filenames or the semicolon special character or carriage returns, LINK will halt after displaying the prompt which does not have a response and wait for the user to enter a legal response. When a legal response has been entered, LINK continues the link session.

For example, a response file may contain:

```
FUN TEXT TABLE CARE /PAUSE/MAP
FUNLIST
COBLIB.LIB
```

This response file will cause LINK to load the four files. LINK will pause before creating and producing a public symbol map to permit the user to swap diskettes (see discussion under /PAUSE in Section 2.8.2. before using this feature). When the user presses any key, the output files will be named FUN.EXE and FUNLIST.MAP. LINK will search the library file COBLIB.LIB, and will use the default settings for the flags.

2.8 COMMAND PARAMETERS AND SWITCHES

2.8.1 Parameters

When entering the LINK command, you must enter four sets of parameters. In method 1, a set is entered after each of the four system prompts. In method 2, all four sets are entered on the same command line and separated by commas. In method 3, the four sets are entered into a response file.

The four sets of parameters are:

1) Object Modules,
2) Run File,
3) List File,
4) Library List.

Object Modules (.OBJ)

Enter a filename of the object modules to be linked. LINK assumes by default that the filename extension is .OBJ. If an object module has any other filename extension, the extension must be given here. Otherwise, the extension may be omitted. Modules must be separated by plus signs (+).

Remember that LINK loads segments into classes in the order encountered (see Section 2.5, "Certain Definitions"). Use this information for setting the order in which the object modules are entered.

Run File (.EXE)

The *filename* entered will be created to store the *run file* (executable object code) that results from a link session. A *run file* receives the filename extension .EXE, even if the user specifies an extension (the user specified extension is ignored).

If no *filename* is entered, LINK uses the first filename entered in the previous parameter as the RUN filename.

List File (.MAP)

The *filename* entered here causes LINK to create a file with that *filename* and *filename extension* .MAP to store a list of all reference labels and subroutine names cross-referenced to the appropriate destinations in the program.

Libraries (.LIB)

The valid responses are one to eight library *filenames*. Not entering a parameter assumes no library search. Library files must have been created by a LIB program (see Section 4). If no *filename extension* is entered, LINK assumes by default that the filename extension is .LIB for library files.

Library filenames must be separated by blank spaces or plus signs (+).

LINK searches the library files in the order listed to resolve external references. When it finds the module that defines the external symbol, LINK processes the module as another object module.

If LINK cannot find a library file on the diskettes in the disk drives, it returns the message:

Cannot find library *<library-name>*
Enter new drive letter:

Simply press the letter for the drive designation (for example, B).

LINK does not search within each library file sequentially. LINK uses a method called Dictionary Indexed Library Search. This means that LINK finds definitions for external references by index access rather than searching from the beginning of the file to the end for each reference. This indexed search reduces substantially the link time for any sessions involving library searches.

2.8.2 Switches

The word "switch" is another term for "parameter". The idea behind the word "switch" is that it is the sort of parameter that turns something, such as a subroutine, on or off. Inserting a "switch" causes the LINK program to do, or not do, a function.

The six switches monitor alternate linker functions. Switches may be entered at the end of each parameter, or may be entered as a group after the last parameter. Each switch must be preceeded by a slash (/).

All switches may be abbreviated, from a single letter through the whole switch name. The only restriction is that an abbreviation must be a sequential sub-string from the first letter through the last entered; no gaps or transpositions are allowed. For example:

LEGAL	ILLEGAL
/D	/DSL
/DS	/DAL
/DSA	/DLC
/DSALLOCA	/DSALLOCT

/DSALLOCATE

Use of the /DSALLOCATE switch directs LINK to load all data (DGroup) at the high end of the data segment. Otherwise, LINK loads all data at the low end of the Data Segment. At run time, the DS pointer is set to the lowest possible address and allows the entire DS segment to be used. Use of the /DSALLOCATE switch in combination with the default load low (that is, the /HIGH switch is not used), permits the user application to allocate dynamically any available memory below the area specifically allocated within DGroup, yet to remain addressable by the same DS pointer. This dynamic allocation is needed for Pascal and FORTRAN programs.

NOTE: The user's application program may dynamically allocate up to 64 K bytes (or the actual amount available) less the amount allocated within DGroup.

/HIGH

Use of the /HIGH switch causes LINK to place the Run image as high as possible in memory. Otherwise, LINK places the Run file as low as possible.

NOTE: Do not use the /HIGH switch with Pascal or FORTRAN programs.

/LINENUMBERS

Use of the /LINENUMBERS switch directs LINK to include in the list file the line numbers and addresses of the source statements in the input modules. Otherwise, line numbers are not included in the list file.

NOTE: Not all compilers produce object modules that contain line number information. In these cases, of course, LINK cannot include line numbers.

/MAP

/MAP directs LINK to list all public (global) symbols defined in the input modules. If /MAP is not given, LINK will list only errors (which includes undefined globals).

The symbols are listed alphabetically. For each symbol, LINK lists its value and its segment:offset location in the Run file. The symbols are listed at the end of the List file.

/PAUSE

The /PAUSE switch causes LINK to pause in the link session when the switch is encountered. Normally, LINK performs the linking session without stopping from beginning to end. The pause allows the user to swap the diskettes before LINK outputs the Run (.EXE) file.

When LINK encounters the /PAUSE switch, it displays the message:

```
About to generate .EXE file
Change disks <hit ENTER>
```

LINK resumes processing when the user presses the **Return** key (Return is synonymous with Enter).

CAUTION: Do not swap the diskette which will receive the List file, or the diskette used for the VM.TMP file, if created.

/STACK: <*number*>

number represents any positive *numeric value* (in hexadecimal radix) up to 65536 bytes. If the /STACK switch is not used for a link session, LINK calculates the necessary stack size automatically.

If a value of 1 to 511 is entered, LINK uses 512.

All compilers and assemblers should provide information in the object modules that allows the linker to compute the required stack space.

At least one object (input) module must contain a stack allocation statement. If not, LINK displays the following error message:

```
Warning: No STACK segment
```

2.9 ERROR MESSAGES

All errors cause the link session to abort. Therefore, after the cause is found and corrected, LINK must be rerun.

Table IV-A
LINK ERROR MESSAGES

ATTEMPT TO ACCESS DATA OUTSIDE OF SEGMENT BOUNDS, POSSIBLY BAD OBJECT MODULE

Cause: probably a bad object file.

BAD NUMERIC PARAMETER

Cause: Numeric value not in digits.

CANNOT OPEN TEMPORARY FILE

Cause: LINK is unable to create the file VM.TMP because the disk directory is full.

Cure: insert a new diskette into a spare drive. Do not change the diskette that will receive the list.MAP file.

CANNOT OPEN RESPONSE FILE

Cause: response file not opened before using LINK.

Cure: open response file.

ERROR: DUP RECORD TOO COMPLEX

Cause: DUP record in assembly language module is too complex.

Cure: simplify DUP record in assembly language program.

...continued

Table IV-A *(cont)*
LINK ERROR MESSAGES

ERROR: FIXUP OFFSET EXCEEDS FIELD WIDTH

Cause: an assembly language instruction refers to an address with a short instruction instead of a long instruction.

Cure: edit assembly language source and reassemble.

INPUT FILE READ ERROR

Cause: probably a bad object file.

INVALID OBJECT MODULE

Cause: object module(s) incorrectly formed or incomplete (as when assembly was stopped in the middle).

SYMBOL DEFINED MORE THAN ONCE

Cause: LINK found two or more modules that define a single symbol name.

PROGRAM SIZE OR NUMBER OF SEGMENTS EXCEEDS CAPACITY OF LINKER

Cause: the total size may not exceed 384K bytes and the number of segments may not exceed 255.

REQUESTED STACK SIZE EXCEEDS 64K

Cure: specify a size less than or equal to 64K bytes with the /STACK switch.

...continued

Table IV-A *(cont)*
LINK ERROR MESSAGES

SEGMENT SIZE EXCEEDS 64K

Cause: 64 K bytes is the addressing system limit.

SYMBOL TABLE CAPACITY EXCEEDED

Cause: too many very long names entered; exceeding approximately 25K bytes.

TOO MANY EXTERNAL SYMBOLS IN ONE MODULE

Cause: the limit is 256 external symbols per module.

TOO MANY GROUPS

Cause: the limit is 10 groups.

TOO MANY LIBRARIES SPECIFIED

Cause: the limit is 8.

TOO MANY PUBLIC SYMBOLS

Cause: the limit is 1024.

TOO MANY SEGMENTS OR CLASSES

Cause: the limit is 256 (Segments and Classes taken together).

...continued

Table IV-A *(cont)*
LINK ERROR MESSAGES

UNRESOLVED EXTERNALS: <*list*>

Cause: The external symbols listed have no defining module among the modules or library files specified.

VM READ ERROR

Cause: a disk problem; not LINK caused.

WARNING: NO STACK SEGMENT

Cause: none of the object modules specified contains a statement allocating stack space, but the user entered the /STACK switch.

WARNING; SEGMENT OF ABSOLUTE OR UNKNOWN TYPE

Cause: a bad object module or an attempt to link modules LINK cannot handle (e.g., an absolute object module).

WRITE ERROR IN TMP FILE

Cause: no more disk space to expand VM.TMP file.

WRITE ERROR ON RUN FILE

Cause: usually not enough disk space for Run file.

Part IV

Section 3

DEBUG

Section 3

DEBUG

3.1 INTRODUCTION

DEBUG is a debugging program used to provide a controlled testing environment for binary and executable object files. Note that text editors such as the *Hyperion Text Editor* are used to alter source files; DEBUG is the counterpart for binary files. DEBUG eliminates the need to reassemble a program to see if a problem has been fixed by a minor change. It allows you to alter the contents of a file or the contents of a CPU register, and then to immediately re-execute the program to check the validity of the changes.

All DEBUG commands may be aborted at any time by pressing **Ctrl + Break**. **Ctrl + Num Lock** suspends the display, so that the user can read it before the output scrolls away. Pressing any key other than **Ctrl + Break** or **Ctrl + Num Lock** restarts the display. All of these commands are consistent with the control character functions available at the DOS command level.

3.2 INVOCATION

To invoke DEBUG:

STEP

1a) Enter the word **DEBUG** and press **Return**,

OR

...continued

STEP *(cont)*

1b) enter the command line:

DEBUG [file]

and press **Return**,

OR

1c) enter the command line:

DEBUG [file][arglist]

and press **Return.**

Entering the word DEBUG by itself invokes the DEBUG program (step 1a).

Entering the word DEBUG followed by *file (a filename* and *extension)* loads the corresponding file into memory and invokes DEBUG (step 1b).

Entering the word DEBUG followed by *file* (a *filename* and *extension)*, and then the *arguments* loads the corresponding file into memory, substitutes the arguments and switches into the file, and invokes the DEBUG program. The *arglist* includes the filename parameters and switches that are to be passed to the file when loaded into memory (step 1c).

In all three cases, after DEBUG has been loaded into the Hyperion, a hyphen (-) is displayed as a prompt and the user can enter the DEBUG *commands* and command *parameters* that are described on the following pages.

3.3 PARAMETERS

All DEBUG commands accept parameters, except the Quit command. Parameters may be separated by delimiters (spaces or commas), but a delimiter is a must only between two consecutive hexadecimal values. Thus, the following commands are equivalent.

```
dcs:100 110
d cs: 100 110
d,cs:100,110
```

Also, entries may be made in any combination upper or lower case.

Table IV-B
DEBUG COMMAND PARAMETERS

PARAMETERS	DEFINITION
drive	*Diskette drive*. A one-digit hexadecimal value to indicate which *drive* a file will be loaded from or written to. The valid values are 0-3. These values designate the drives as follows: 0 = A, 1 = B, 2 = C, 3 = D. Drive D is the hard disk.
byte	*Byte*. A two-digit hexadecimal value to be placed in or read from an address or register.
record	*Logical record number or number of sectors*. A 1- to 3-digit hexadecimal value. When used in pairs, the first number identifies a record number, and the second identifies a number of records. A record represents 512 bytes; a storage space of 512 bytes on a diskette is called a "sector".

...continued

Table IV-B *(cont)*
DEBUG COMMAND PARAMETERS

PARAMETER	DEFINITION
record *(cont)*	Logical record numbering starts with zero (0) for the first record on track 0, and increases to the last record on a diskette (approximately 560 = 350H) Used with the Load and Write commands, to indicate the location and quantity of information to be loaded from diskette, or written to diskette.
value	***Value.*** A *hexadecimal value* up to four digits used by various commands to specify a port number or a quantity.
address	***Memory address location.*** A two-part designation consisting of either an alphabetic segment register designation (DS, CS, SS, ES) or a four-digit segment address plus an offset value. The segment designation or segment address may be omitted, in which case the default segment is used. DS is the default segment for all commands except Go, Load, Trace, Unassemble, and Write, for which the default segment is CS. All numeric values are hexadecimal. For example: **CS:0100** points to offset 100 in the current code register; **04BA:0100** points to memory address 04BA0 + 0100 = 04CA0 The colon is required between a segment designation (whether numeric or alphabetic) and an offset.

...continued

DEBUG

Table IV-B *(cont)*
DEBUG COMMAND PARAMETERS

PARAMETER	DEFINITION
range	*Memory address range*. Points to, or locates, a section of memory by giving its first and last address, or by giving its first address and specifying how many bytes of memory are involved. The range can be expressed in three ways: *<address>* *<address>* *<address>* **L** *<value>* *<address>* *The first address* is the beginning address of the range. The second *address* is the last address. *"Value"* specifies the amount of memory in the range, in bytes. If nothing follows the first *address*, then L80 is assumed. The two *addresses* must be separated by a space or comma. The letter "L" does not need to be preceded or followed by a space or comma. Examples: CS:100 110 CS:100 L 10 CS:100 The following is illegal: CS:100 CS:110 ^ error The limit for *<range>* is 10000 hex. Use 0000 or (0) to specify a *<value>* of 10000 hex within four digits.
list	*List of byte values*. A series of *<byte>* values or characters in a *<string>* line. *<list>* must be the last parameter on the command line. Example: fcs:100 42 45 52 54 41 The values "42 45 52 54 41" have been subsituted for the "list" parameter.

...continued

DEBUG

Table IV-B *(cont)*
DEBUG COMMAND PARAMETERS

PARAMETER	DEFINITION
string	***Character string***. Any number *of characters* enclosed in quote marks. Quote marks may be either single (') or double ("). Within <string>s, the opposite type of quote marks may be used freely and are considered by the system as characters. If the same type of quote marks as the delimiter quote marks must appear within a <string>, the quote marks must be doubled or changed to the other type. For example, the following strings are legal: 'This "string" is okay.' "This ' 'string' ' is okay.' However, this string is illegal: 'This 'string' is not.' Similarly, these strings are legal: "This 'string' is okay." "This " "string"" is okay." However, this string is illegal: "This "string" is not." Note that the double quotations are not necessary in the following strings where the quotes are changed. "This ' 'string' ' is not necessary." "This " "string" " is not necessary." The ASCII values of the characters in the string are used as a <list> of byte values.

DEBUG

3.4 COMMANDS

Each DEBUG command line consists of a single letter (the command) followed by one or more parameters. When entering the command line, the cursor keypad and keyboard keys can be used to edit the line before pressing **Return**. Pressing **Return** enters the command line into the system.

If a syntax error occurs in a DEBUG command line, DEBUG reprints the command line and indicates the error with a caret (^) and the word "error".

For example:

```
dcs:100 cs:110
  ^
error
```

All commands and parameters may be entered in either upper or lower case. Any combination of upper and lower case may be used in commands.

The DEBUG commands summarized below are described in detail, on the following pages. The DEBUG commands are:

Assemble	**M**ove
Compare	**N**ame
Dump	**O**utput
Enter	**Q**uit
Fill	**R**egister
Go	**S**earch
Hex	**T**race
Input	**U**nassemble
Load	**W**rite

In the command formats which follow, the DEBUG commands are each followed by one or more parameters. The parameters enclosed by angle brackets (<,>) are mandatory. Parameters enclosed by square brackets ([,]) are optional. Parameters must be entered in the order shown.

A
Assemble Statements into Memory

Format: **A**[***address***]

Function: Assembles statements directly into memory starting at [*address*].

Comments: Any numeric input for this commands must be in hexadecimal notation. If a memory location (address) isn't specified, statements are assembled starting at the location following the previous Assemble command.
If no Assemble command has been entered previously, statements start at CS:0100.

Two pseudo-instructions have been included in Assemble: DW, which assembles word values into memory, and DB, which assembles byte values.

All forms of register indirect commands and opcode synonyms are supported.

When all the statements are entered, press **Return** to obtain a DEBUG prompt.

Mnemonics: The segment override mnemonics are: CS:, DS:, ES:, and SS:.
The mnemonic for far return is RETF.
String mnemonics must state the string size explicitly.
Use MOVSB to move byte strings and MOVESW to move word strings.
Prefix mnemonics are entered in front of the opcode they reference. You can also enter them on a separate line.

A
(cont)

Short, near and far jumps and calls are automatically assembled, depending on their byte displacement from the destination address. To override them, use the NEAR (or NE) and FAR prefixes.

WAIT and FWAIT prefixes are explicitly specified for 8087 opcodes.

When using other operands while assembling statements, use square brackets to enclose operands that refer to memory. For some operands, you must also specify byte memory location (with the prefix BY or BYTE PTR) or word memory location (with the prefix WD or WORD PTR).

Example: Use the following as an example of how to use Assemble:

```
A>DEBUG
-A100
2292:0101 MOV AX,20            ;ADD TWO NUMBER
2292:0103 MOV BX,10F           ;TOGETHER
2292:0106 ADD AX,BX            ;
2292:0108 OR AX,AX             ;IF RESULT IS ZERO,
2292:010A JZ 01110A            ;JUMP TO 0111
2292:010C PUSH AX              ;COPY AX INTO
2292:010D POP CX               ;CX
2292:010E JMP SHORT 0113
2292:0111 MOV WORD PTR [SI],AX
2292:0113 RETF                 ;RETURN FROM FAR
2292:0114                      ;CALL
```

C
Compare Portions of Memory

Format: **C<*range*> <*address*>**

Function: Compares the portion of memory specified by <*range*> to a portion of the same size beginning at <*address*>.

Comments: If the two areas of memory are identical, there is no display and DEBUG returns with the DEBUG prompt. If there *are* differences, they are displayed in four columns as:

```
<address1> <byte1> <byte2> <address2>
<address1> <byte1> <byte2> <address2>
     .        .      .        .
     .        .      .        .
```

Example: The following commands have the same effect:

```
C100,1FF 300
```

or

```
C100L100 300
```

Each command compares the block of memory from 100 to 1FFH with the block of memory from 300 to 3FFH.

D
Display Contents of Memory (Dump)

Format: **D**[***range***]

Function: Displays the contents of the specified region of memory.

Comments: If a *range* of addresses is specified, the contents of the *range* are displayed. If the Dump command is entered without parameters, 128 bytes are displayed at the first address (DS:100) after that displayed by the previous Dump command.

The dump is displayed in two portions: a hexadecimal dump (each byte is shown as a hexadecimal value) and an ASCII dump (the bytes are shown in ASCII characters). Nonprintable characters are denoted by a period (.) in the ASCII portion of the display. Each display line shows sixteen bytes with a hyphen between the eighth and ninth bytes. At times, displays are split in this manner to fit them on the page. Each displayed line, except possibly the first, begins on a 16-byte boundary.

Example: If the user enters the command:

```
DCS:100 110
```

DEBUG displays the first 16 (10H) bytes of memory starting at the current pointer location.

```
04BA:0100 42 45 52 54 41 ... 4E 44 BERTA T.
```

DEBUG

D
(cont)

If the following command is entered:

the first 128 bytes of memory found, starting from the current pointer location, are displayed as 8 lines of 16 bytes each. Each line of the display begins with an address; incremented by 16 from the address on the previous line. Each subsequent Dump (entered without parameters) displays the bytes immediately following those last displayed.

If the user enters the command:

```
DCS:100 L 20
```

the display is formatted as described above, but 23 (20H) bytes are displayed.

If the user enters the command:

```
DCS:100 115
```

the display is formatted as described above, but all the bytes in the range of lines from 100H to 115H in the CS segment are displayed.

E
Enter Individual Byte Values Into Memory

Format: **E**<***address***> [***list***]

Function: Enters byte values into memory at the specified <*address*>.

Comments: If the optional <*list*> of values is entered, the replacement of byte values occurs automatically. (If an error occurs, no byte values are changed.) If the <*address*> is entered without the optional <*list*>, DEBUG displays the address on the next line and the value of the first byte followed by a period. The cursor waits to the right of the period for the user's input. At this point, the Enter command waits for you to perform one of the following actions:

1. Type in the value after the period. If the value typed in is not a legal hexadecimal value or if more than two digits are typed, the illegal or extra character is not echoed.

2. Press the **space bar** to advance to the next byte. To change the value, simply enter the new value as described in (1.) above. If the user spaces beyond an eight-byte boundary, DEBUG starts a new display line with the address displayed at the beginning.

3. Type a hyphen (-) to return to the preceding byte. If the user decides to change a byte behind the current position, typing the hyphen returns the current position to the previous byte. When the hyphen is typed, a new line is started with the address and its byte value displayed.

4. Press the **Return** key to terminate the Enter command. The **Return** key may be pressed at any byte position.

E
(cont)

Example: Assume the following command is entered:

```
ECS:100
```

DEBUG displays:

```
04BA:0100 EB._
```

To change this value to 41, enter "41" as shown:

```
04BA:0100 EB.41_
```

To step through the subsequent bytes, press the **space bar** to see:

```
04BA:0100 EB.41    10.    00.    BC._
```

To change BC to 42:

```
04BA:0100 EB.41    10.    00.    BC.42_
```

Now, realizing that 10 should be 6F; enter the hyphen as many times as needed to return to byte 0101 (value 10), then replace 10 with 6F:

```
04BA:0100 EB.41    10.    00.    BC.42-
04BA:0102 00.-
04BA:0101 10.6F
```

Pressing the **Return** key ends the Enter command and redisplays the DEBUG prompt.

F
Fill a Range of Addresses

Format: **F<*range*> [*list*]**

Function: Fills the addresses in the <*range*> with the values in the <*list*>.

Comments: If the <*range*> contains more bytes than the number of values in the <*list*>, the <*list*> will be used repeatedly until all bytes in the <*range*> are filled. If the <*list*> contains more values than the number of bytes in the <*range*> the extra values in the <*list*> will be ignored. If any of the memory in the <*range*> is not valid (bad or nonexistent), the error will be propagated into all succeeding locations.

Example: Assume that the following command is entered:

```
F04BA:100 L 100 42 45 52 54 41
```

DEBUG fills memory locations 0B4A:100 through 0B4A:1FF with the bytes specified. The five values are repeated until all 100H bytes are filled.

G
Execute Program (Go)

Format: **G**[=[***address***] [***address...***]]

Function: Executes the program currently in memory.

Comments: If the Go command is entered alone, the program executes as if the program had run outside DEBUG.

If =[*address*] is set, execution begins at the address specified. The equal sign (=) is required, so that DEBUG can distinguish the start address from the breakpoint addresses.

With the other optional *addresses* set, execution stops at the first address encountered, regardless of the position of that address in the list of addresses to halt execution, no matter which branch the program takes. When program execution reaches a breakpoint, the registers, flags, and the decoded instruction are displayed for the last instruction executed. (The result is the same as if you had entered the Register command for the breakpoint address.)

Up to ten breakpoints may be set. Breakpoints may be set only at addresses containing the first byte of an opcode. If more than 10 breakpoints are set, DEBUG returns the message "**BP Error**".

G
(cont)

The user stack pointer must be valid and have six bytes available for this command. The Go command uses an IRET instruction to cause a jump to the program under test. The user stackpoint is set, and the user flags, code segment register, and instruction pointer are pushed on the user stack. (Thus, if the user stack is not valid or is too small, the operating system may crash.) An interrupt code (0CCH) is placed at the specified breakpoint address(es). When an instruction with the breakpoint code is encountered all breakpoint addresses are restored to their original instructions. If execution is not halted at one of the breakpoints, the interrupt codes are not replaced with the original instructions.

Example: Assume the following command is entered:

```
GCS:7550
```

The program current in memory executes up to the address 7550 in the CS segment. Then DEBUG displays registers and flags, after which the Go command is terminated.

After a breakpoint has been encountered, if you enter the Go command again, then the program executes just as if the user had entered the filename at the MS-DOS command level. The only difference is that program execution begins at the instruction after the breakpoint rather than at the usual start address.

H
Perform Hexadecimal Arithmetic

Format: **H**<***value***> <***value***>

Function: Performs hexadecimal arithmetic on the two parameters.

Comments: First, DEBUG adds the two parameters, then subtracts the second parameter from the first. The result of the arithmetic is displayed on one line; first the sum, then the difference.

Negative values are represented as subtracted from 0000. For example, −2 would be FFFE.

Example: Assume the following command is entered:

```
H19F 10A
```

DEBUG performs the calculations and then returns the results:

```
02A9 0095
```

I
Read a Port Address and Display Byte Found (Input)

Format: I<***value***>

Function: Reads a port (input address) and displays the byte found at that address.

Comments: A 16-bit port address is allowed.

Example: Assume the following command is entered:

```
I2F8
```

Assume also that the byte at the port is 42H. DEBUG inputs the byte and displays the value:

```
42
```

DEBUG

L
Load File into Memory

Format: **L**[***address***] [***drive***] [***record***] [***record***]

Function: Loads a file into memory at the address specified and sets the BX,CX register combination to the number of bytes read.

Comments: The file must first have been named with either the DEBUG invocation command or with the Name command. Both the invocation and the Name commands format a filename properly in the normal format of a file control block at CS:5C. If the Load command is given without any parameters, DEBUG loads the file into memory beginning at address CS:100 and sets BX:CX to the number of bytes loaded. If the Load command is given with an address parameter, loading begins at the memory <*address*> specified. If Load is entered with all parameters, absolute disk sectors are loaded, not a file. Each <*record*> is taken from the <*drive*> specified (the drive designation is numeric here – 0 = A:, 1 = B:, 2 = C:, etc.); DEBUG begins loading with the first <*record*> specified, and continues until the number of sectors specified in the second <*record*> have been loaded.

Example: Assume the following commands have been entered:

```
A:DEBUG
NFILE.COM
```

Now, to load FILE.COM, enter:

```
L
```

L
(cont)

DEBUG loads the file and returns the DEBUG prompt. Assume you want to load only portions of a file or certain records from a disk. To do this, enter:

```
L04BA:100 2 0F 6D
```

DEBUG then loads 109 (6D hex) records beginning with logical record number 15 (0FH) into memory beginning at address 04BA:0100. When the records have been loaded, DEBUG simply redisplays the prompt.

If the file has a .EXE extension, then it is relocated to the load address specified in the header of the .EXE file: the address parameter is always ignored for .EXE files. Note that the header itself is stripped off the .EXE file before it is loaded into memory. Thus, the size of a .EXE file on disk will differ from its size in memory.

If the file named by the Name command or specified on on invocation is a .HEX file, then entering the Load command with no parameters causes loading of the file beginning at the address specified in the .HEX file. If the Load command includes the option address, DEBUG adds the address specified in the Load command to the address found in the .HEX file to determine the start address for loading the file.

M
Move Block of Memory

Format: **M<*range*> <*address*>**

Function: Moves the block of memory specified by <*range*> to the location beginning at the <*address*> specified.

Comments: Overlapping moves (moves where part of the block overlaps some of the current addresses) are always performed without loss of data. Addresses that could be overwritten are moved first. The sequence for moves from higher addresses to lower addresses is to move the data beginning at the block's lowest address and working towards the highest. The sequence addresses is to move the data beginning at the block's highest addresses and working towards the lowest.

Note that if the addresses in the block being moved will not have new data written to them, the data there before the move will remain; that is, the Move command really copies the data from one area into another, in the sequence described, and writes over the new addresses. This is why the sequence of the move is important.

Example: Assume you enter:

```
MCS:100 110 CS:500
```

DEBUG first moves address CS:110 to addresses CS:510, then CS:10F to CS:50F, and so on until CS:100 is moved to CS:500. You should enter the Dump command, using the <*address*> entered for the Move command, to review the results of the move.

N
Name a File and Assign Parameters

Format: **N**[***filename***] [***filename...***]

Function: Sets filenames

Comments: The Name command performs two distinct functions, both having to do with filenames. First, Name is used to assign a *filename* for a later Load or Write command. Thus, if you invoke DEBUG without naming any file to be debugged, then the **N**[*filename*] command must be given before a file can be loaded. with the Load command. Second, Name is used to assign *filename parameters* to the file being debugged. In this case, the Name command accepts a list of parameters that are used by the file being debugged.

These functions overlap. Consider the following set of DEBUG commands:

```
-NFILE1.EXE
-L
-G
```

Because of the two-pronged effect of the Name command, the following happens:

1. Name assigns the filename FILE1.EXE to the filename to be used in any later Load or Write commands.

2. Name also assigns the filename FILE1.EXE to the first filename parameter to be used by any program that is later debugged.

3. Load loads FILE1.EXE into memory.

4. Go causes FILE1.EXE to be executed with FILE1.EXE as the single filename parameter (that is, FILE1.EXE is executed as if FILE1.EXE had been typed at the command level).

N
(cont)

A more useful chain of commands might go like this:

```
-NFILE1.EXE
-L
-NFILE2.DAT FILE3.DAT
-G
```

Here, Name sets FILE1.EXE as the filename for the subsequent Load command. The Load command loads FILE1.EXE into memory, and then the Name command is used again, this time to specify the parameters to be used by FILE1.EXE. Finally, when the Go command is executed, FILE1.EXE is executed as if FILE1.DAT FILE2.DAT FILE3.DAT had been typed at the MS-DOS command level.

Note; if a Write command were executed at this point, then FILE1.EXE (i.e. the file being debugged) would be saved with the name FILE2.DAT. To avoid such undesired results, you should always execute a Name command before either a Load or a Write.

There are four distinct regions of memory that can be affected by the Name command:

CS:5C FCB for file 1
CS:6C FCB for file 2
CS:80 Count of characters
CS:81 All characters entered

N
(cont)

A File Control Block (FCB) for the first filename parameter given to the Name command is set up at CS:5C. If a second filename parameter is given, then an FCB is setup for it beginning at CS:6C. The number of characters typed in the Name command (exclusive of the first character, "N") is given at the location CS:80. The actual stream of characters given by the Name command (again, exclusive of the letter "N") begins at CS:81. Note that this stream of characters may contain switches and delimiters that would be legal in any command typed at the MS-DOS command level.

Example: A typical use of the Name command would be:

```
DEBUG PROG.COM
-NPARAM1 PARAM2/C
-G
-
```

In this case, the Go command executes the file in memory as if the following command line had been entered:

```
PROG PARAM1 PARAM2/C
```

Testing and debugging therefore reflect a normal run time environment for PROG.COM.

O
Output Byte to Port

Format: **O**<***value***> <***byte***>

Function: Sends the <*byte*> specified to the output port specified by <*value*>.

Comments A 16-bit port address is allowed.

Example: Enter:

```
O2F8 4F
```

DEBUG outputs the byte value 4F to output port 2F8.

Q
Exit DEBUG without Saving File (Quit)

Format: **Q**

Function: Exits from the DEBUG program.

Comments: The Quit command takes no parameters and exits DEBUG without saving the file currently being operated on. You are returned to the MS-DOS command level.

Example: To end the debugging session, enter **Q** and press **Return**. DEBUG is terminated, and control returns to the DOS command environment.

R
Display Contents of Registers

Format: **R**[***register-name***]

Function: Displays the contents of one or more CPU registers.

Comments: If no <*register-name*> is entered, the Register command dumps the register save area and displays the contents of all registers and flags.

If a register name is entered, the 16-bit value of that register is displayed in hexadecimal, and then a colon appears as a prompt. The user then either enters a <*value*> to change the register, or simply presses the **Return** key if no change is wanted.

The only valid register-names are:

AX	BP	SS
BX	SI	CS
CX	DI	IP
DX	DS	PC
SP	ES	F

(IP and PC both refer to the instruction pointer.)

Any other entry for <*register-name*> results in a message "**BR Error**".

If F is entered as the <*register-name*>, DEBUG displays each flag with a two character alphabetic code. To alter any flag, enter the opposite two letter code. The flags are either set or clear.

R
(cont)

The flags with their codes for get and clear are listed below:

FLAG NAME	SET	CLEAR
Overflow	OV	NV
Direction	DN	Decrement UP Increment
Interrupt	EI	Enabled DI Disabled
Sign	NG	Negative PL Plus
Zero	ZR	NZ
Auxiliary Carry	AC	NA
Parity	PE	Even PO Odd
Carry	CY	NC

Whenever the user enters the command RF, the flags are displayed in the order shown above in a row at the beginning of a line. At the end of the list of flags, DEBUG displays a hyphen (-). You may enter new flag values as alphabetic pairs. The new flag values can be entered in any order. You are not required to leave spaces between the flag entries. To exit the Register command, press the **Return** key. Flags for which new values were not entered remain unchanged.

R
(cont)

If more than one value is entered for a flag, DEBUG returns a message **"DF Error"**. If you enter a flag code other than those shown above, DEBUG returns a message **"BF Error"**. In both cases, the flags up to the error in the list are changed; flags at and after the error are not. At start up, the segment registers are set to the bottom of free memory, the Instruction Pointer is set to 0l00H, all flags are cleared, and the remaining registers are set to zero.

Example: Enter: **R**

DEBUG displays all registers, flags, and the decoded instruction for the current location. If the location is CS:11A, then DEBUG might display:

```
AX=0E00 BX-OOFF CS-0007 DX=01FF SP=039D
BP=0000
SI=005C    DI=0000    DS=04BA    ES=04BA
SS=04BA CS=04BA
IP=011A NV UP DI NG NZ AC PE NC
04BA:011A CD21      INT     21
```

If you enter **RF**

DEBUG displays the flags:

```
NV UP DI NG NZ AC PE NC -
```

Now enter any valid flag designation, in any order, with or without the DEBUG prompt. To see the changes, either enter the R(Register) or RF(Register Flag) command. If you enter **RF**

DEBUG displays:

```
NV UP EI PL NZ AC PE CY -
```

Press the **Return** key to leave the flags this way, or to enter different flag values.

DEBUG

S
Search Range for List of Bytes

Format: **S**<***range***> <***list***>

Function: Searches the range specified for the <*list*> of bytes specified.

Comments: The <*list*> may contain one or more bytes, each separated by a space or comma. If the <*list*> contains more than one byte, only the first address of the byte string is returned. If the <*list*> contains only one byte, all addresses of the byte in the <*range*> are displayed.

Example: If you enter:

```
SCS:100 110 41
```

DEBUG might respond:

```
04BA:0104
04BA:010D
-
```

DEBUG

T
Execute Instruction and Display (Trace)

Format: **T**[**=address**] [***value***]

Function: Executes one instruction and displays the contents of all registers, flags, and the decoded instruction.

Comments: If the original =<*address*> is entered, tracing occurs at the =<*address*> specified. The optional <*value*> causes DEBUG to execute and trace the number of steps specified by <*value*>.

The Trace command uses the hardware trace mode of the microprocessor. Consequently, the user may also trace instructions stored in ROM.

Example: Enter:

DEBUG returns a display of the registers, flags, and decoded instruction for that one instruction. Assume that the current position is 04BA:011A; then DEBUG might return the display:

```
AX=0E00  BX=00FF  CX=0007  DX=01FF
SP=039D
BP=0000  SI=005C  DI=0000  DS=04BA
ES=04BA
SS=04BA CS=04BA IP=011A NV UP DI NG NZ
AC
PE NC 04BA:011A CD21 INT 21
```

Now enter: **T.036011A 10**

DEBUG executes sixteen (10 hex) instructions beginning at 011A in the current segment and then displays all registers and flags for each instruction as it is executed. The display scrolls away until the last instruction is executed. Then the display stops, and you can see the register and flag values for the last few instructions performed. Remember that **Ctrl+Num Lock** suspends the display at any point, so that you can study the registers and flags for any instruction.

U
Display Source Statements (Unassemble)

Format: **U[*range*]**

Function: Unassembles bytes and displays the source statements that correspond to them, along with addresses and byte values.

Comments: The display of unassembled code looks like a listing for an assembled file. If you enter the Unassemble command without parameters, 20 hexadecimal bytes are unassembled at the first address after that displayed by the previous Unassemble command. If you enter the Unassemble command with the <*range*> parameters, then DEBUG disassembles all bytes in the range. If the <*range*> is given as an <*address*> only, then 20H bytes are unassembled, not 80H.

Example: Enter:

```
U04BA:100 L10
```

DEBUG unassembles 16 bytes beginning at address 04BA:0100:

```
04BA:0100 206472   AND [SI+72],AH
04BA:0103 69       DB  69
04BA:0104 7665     JBE 016B
04BA:0106 207370   AND [BP+DI+70],DH
04BA:0109 65       DB  65
04BA:010A 63       DB  63
04BA:010B 69       DB  69
04BA:010C 66       DB  66
04BA:010D 69       DB  69
04BA:010E 63       DB  63
04BA:010F 61       DB  61
```

U
(cont)

If you enter:

```
U04bA:0100 0108
```

the display shows:

```
04BA:0100 206472   AND [SI+72],AH
04BA:0103 69       DB  69
04BA:0104 7665     JBE 016B
04BA:0106 207370   AND [BP+DI+70],DH
```

If the bytes in some addresses are altered, the unassembler alters the instruction statements. The Unassemble command can be entered for the changed locations, the new instructions viewed, and the unassembled code used to edit the source file.

W
Write File to Disk

Format: **W[*address*] [*drive*] [*record*] [*record*]**

Function: Writes the file being debugged to a disk file.

Comments: If only the Write appears, BX:CX must already be set to the number of bytes to be written; the file is written beginning from CS:100. If the Write command is given with just an *address*, then the file is written beginning at that *address*. If a Go or Trace command was used, BX:CX must be reset before using the Write command without parameters. (Note that if a file is loaded and modified, the name, length, and starting *address* are all set correctly to save the modified file as long as the length has not changed.)

The file must have been named either with the DEBUG invocation command or with the Name command (see Name above). Both the invocation and the Name commands format a file name properly in the normal format of a file control block at CS:5C.

If the Write command is given with parameters, the write begins from the memory *address* specified; (the *drive* designation is numeric here – 0 = A:, 1 = B:, 2 = C:, etc.); DEBUG writes the file beginning at the logical record number specified by the first *record*; and continues until the number of sectors specified in the second *record* have been written.

WARNING: Writing to absolute sectors is EXTREMELY DANGEROUS because the process bypasses the file handler.

W
(cont)

Example: Enter:

```
W
```

DEBUG writes out the file to disk and displays the DEBUG prompt:

```
W
-_
```

Another example:

```
CS:100 1 37 2B
```

DEBUG writes out the contents of memory, beginning with the address CS:100 to the disk in drive B:. The data written out starts in disk logical record number 37H and consists of 2BH records. When the write is complete, DEBUG displays the prompt:

```
WCS:100 1 378 2B
```

3.5 ERROR MESSAGES

During the DEBUG session, you may receive any of the following error messages. Each error terminates the DEBUG command with which it is associated, but does not terminate DEBUG itself.

Table IV-C DEBUG ERROR MESSAGES

ERROR CODE	DEFINITION
BF	***Bad Flag*** The user attempted to alter a flag, but the characters entered were not one of the acceptable pairs of flag values. See the Register command for the list of acceptable flag entries.
BP	***BP Too many Breakpoints*** The user specified more than ten breakpoints as parameters to the Go command. Re-enter the Go command with ten or fewer breakpoints.
BR	***Bad Register*** The user entered the Register command with an invalid register name. See the Register command for the list of valid register names.
DF	***Double Flag*** The user entered two values for one flag. The user may specify a flag value only once per RF (register flag) command.

Part IV

Section 4

LIB

Section 4

LIB

4.1 WHEN TO USE LIB

LIB is a program produced by Microsoft that can create libraries for a variety of programs or for specific programs. These are libraries of program modules that have previously been compiled. When the time comes to link programs and modules together into one executable program, the LIB command can scan the appropriate libraries and extract the necessary modules.

In order to use LIB, your computer must have at least 38 K bytes of memory space free: 28 K to store the program code and data; 10 K to be available for run space.

Single-Diskette Drive Considerations

A single-diskette drive Hyperion can be used if and only if the output from LIB is sent to the same drive from which the input is taken. LIB does not allow time to swap diskettes during operation on a one-drive Hyperion. Therefore, two diskette drives provides a more practical configuration.

Brief Description

The LIB program can be used to accomplish any of the following tasks:

1) Create a library file.

2) Add, delete, or replace object files (modules) in a library file.

3) Extract modules from a library file and place them into separate object files. The user can then make any modifications to the code that are necessary, and return the object file to the end of the library file.

4.2 HOW TO USE LIB

To use LIB, you would:

STEP

1) Insert the diskette containing the file LIB.EXE. This file contains the LIB program.

2) Set the default drive to the drive containing the diskette.

3) Enter the word **LIB**, press **Return**, and respond to the system prompts as described in Section 4.4.1, *"Method 1: Interactive LIB"*.

   OR

   Enter the word **LIB**, followed by the appropriate parameters. Edit the command line as required, and then press **Return**. The procedure is described in Section 4.4.2, *"Method 2: Command Line LIB"*.

   OR

   Enter the word **LIB**, followed by a filespec, and then press **Return**. The filespec is the filename and extension of the file that contains the parameters to the LIB command. The procedure is described in Section 4.4.3, *"Method 3: Batched LIB"*.

As indicated, the LIB command can be entered in one of three ways. Each way is described in detail in Section 4.4. Command parameters and operators are listed and described in Section 4.5.

4.3 WHAT LIB DOES – DESCRIPTION OF THE LIB PROGRAM

The LIB program performs five library manager functions:

1) It creates the library file, joining the object files (modules of compiled code) into one file. The filename and object modules are entered as parameters to the LIB command.

2) It adds object files to the end of an existing library file.

3) It can delete object modules from a library file.

4) It can replace object modules within a library file.

5) It can extract object modules from a library file and put them back into object files. The code in these files can then be edited and changed as necessary.

The difference between the "object file" and "object module" is that the "object module" is considered to be the sequence of coded instructions by itself (program module), and the object file is that sequence put into its own file with a unique filename.extension.

Index and Cross Reference Listing

LIB also creates an index, in each library file, of the modules in that file. This index minimizes the time taken to find any module or public symbol. LIB will output a cross-reference listing of the public symbols in the library file, if such a listing is requested.

4.4 THREE METHODS OF ENTERING THE LIB COMMAND

LIB may be invoked three ways:

1) ***Method 1: Interactive LIB***. By this method, the user enters the word LIB and waits for the system to prompt for individual parameters.

2) ***Method 2: Command Line LIB***. By the second method, the user enters the word LIB and all necessary parameters on the same command line.

3) ***Method 3: Batched LIB***. By this third method, the user enters the word LIB followed by a filespec. In this last method, the user must first create a file to contain all the necessary parameters. The filespec entered is that for this file.

In all three cases, in addition to the command word LIB, the user must be familiar with the parameters and operators that modify the function of the LIB command. Parameters and operators are described in Section 4.5.

4.4.1 Method 1: Interactive LIB

STEP

1) Enter the word LIB, and press **Return.**

...continued

LIB will be loaded into memory. Then LIB returns a series of three text prompts that appear one at a time. The first prompt asks you for the *Library File* parameter:

Library Name:

STEP *(cont)*

2) Enter the *filespec* (drive:filename.extension) of the library file to be created or edited. If no extension is entered, the system assumes an extension of .LIB.

   There is no default. The *filename*, at least, must be entered. If the library file specified does not exist, the LIB program prompts:

   Library does not exist. Create?

   Enter Y and press **Return** to create a new library file. Enter **N** and press **Return** to abort the library session and return to DOS.

3) Use the cursor control and keyboard keys to edit the characters entered, if necessary (this step is the same for any user input, and will therefore not be restated).

4) Press **Return**.

   The system displays the next prompt:

   Operations:

...continued

```
B>LIB

     Microsoft Library Manager V2.00
(C) Copyright 1983 by Microsoft Inc.

Library Name: LIB1
Operations: +MODUL1+MODUL2+MODUL3-OLDMOD*MODMOD
List file:

LASTLN Disks Files MODE DIR/P 00:53 CHDIR PATH TREE XPLAIN HELP
```

Fig. IV-5 – A sample entry-response session displayed on the screen.

STEP *(cont)*

6) Enter the parameters and operators (described in Section 4.5) that define the LIB function to be performed.

7) Edit the command line, if necessary.

8) Press **Return**.

   The system displays the next prompt:

   List file:

9) Enter a *filespec* for the file to contain the cross-reference listing, and press **Return**.

   If **Return** is pressed without entering a *filespec*, no cross-reference listing file is created.

The LIB program will then begin executing the LIB function requested. This may take from a few seconds to a few minutes, depending on the function.

A sample entry-response session is shown in Fig. IV-5.

To stop the LIB session at any time, press **Ctrl + Break**.

4.4.2 Method 2: Command Line LIB

STEP

1) Enter the following command line:

   LIB <*library file*><*operations*>[*list file*]

2) Edit the command line, if necessary and press **Return** (see Fig. IV-6).

The parameter enclosed in angle brackets (<,>) must be entered. The parameters enclosed in square brackets ([,]) are optional. The parameters are:

- ***library file*** is a *filespec* (drive:filename.extension) of the library file to be created or edited.

+ ***operations*** is the sequence of *parameters and operators* that define the LIB function being called up.

- ***list file*** is the *filespec* (drive:filename.extension) of the cross-reference file.

If an extension to the library filename is not entered, the system assumes .LIB to be the extension.

Parameters and operators are described in Section 4.5. If the library file specified does not exist, the LIB program will prompt:

Library does not exist. Create?

Enter **Y** and press **Return** to create a new library file. Enter **N** and press **Return** to abort the library session and return to DOS.

If you enter a semicolon (;) immediately after the library filename, the LIB program will read through the library file and perform a consistency check. No changes will be made to the modules in the library.

If you enter the library filename followed immediately by a comma and a list filename, the system will perform a consistency check of the library file, and then produce a cross reference list file.

If you have many operations to perform during a library session, use the ampersand (&) symbol at the end of the command line before pressing **Return**. This returns the cursor to the beginning of the next line and enables you to continue entering the command line. The ampersand enables you to extend the command line beyond the physical one-line limit.

```
B>LIB NEWLIB +MOD1+MOD2+MOD3 LIST
```

Fig. IV-6 – An example of a LIB command line entered on screen.

4.4.3 Method 3: Batched LIB

STEP

1) Enter the command line:

   LIB @<*filespec*>

2) Edit the command line, if necessary.

3) Press **Return**.

The parameter *filespec* is the name of a response file. A response file contains answers to the LIB prompts (shown under method 1, "Interactive LIB").

IMPORTANT: Before using method 3 to invoke LIB, the user must first create the response file.

A response file has text lines, one for each prompt. Responses must appear in the same order as the command prompts appear.

Use parameters and operator characters in the response file the same way as they are used for responses entered on the terminal keyboard.

When the LIB session begins, each prompt will be displayed in turn with the responses from the response file. If the response file does not contain answers for all the prompts, LIB will use the default responses.

4.5 COMMAND PARAMETERS AND OPERATORS

4.5.1 Parameters

When entering the LIB command, you must enter three sets of parameters. In method 1, a set is entered after each of the three system prompts. In method 2, all three sets are entered on the same command line and separated by commas. In method 3, the three sets are entered into a response file.

The three sets of parameters are:

1) library file,
2) operations,
3) list file.

Library File

This is the *filespec* of the library file you want to manipulate. LIB assumes the *filename extension* is .LIB, unless otherwise specified. Because LIB can manage only one library file at a time, only one *filename* is allowed.

If you enter a library *filename* and follow it immediately with a semicolon (;), LIB will perform a consistency check only, then return to the operating system. Any errors in the file will be displayed.

Operations

Any of three operations may be specified: add, delete, extract.

Each operation is specified by an operator: plus sign (+) for add; minus sign (−) for delete; and asterisk (*) for extract and place into an external object file.

Each module or object filename must be preceeded by an operator which tells LIB what to do with it.

The string of object filenames, modules, and operators, must follow each other without any delimiters, i.e., no blank spaces or commas.

Entering an ampersand (&) as the last character in a line enables you to continue the operation parameter (string of modules and operators) onto the next line.

List File

Entering a *filespec* for the list file causes the system to create a cross-reference file for the library specified.

The cross-reference listing contains two lists. The first is an alphabetical listing of all public symbols. Each symbol name is followed by the name of the module it is contained in. The second list is an alphabetical list of the modules in the library. Under each module name is an alphabetical listing of the public symbols in that module.

4.5.2 Operators

LIB provides six operators. These determine how the LIB program is to function.

Plus sign (+)

Use the *plus sign*, followed by an object filespec (drive:filename.extension) to append the object file as the last module in the library file.

When LIB sees the *plus sign*, it assumes that the filename extension is .OBJ. You may override this assumption by specifying a different extension.
LIB strips the drivespec and extension from the filespec, leaving only the object filename. For example, if the filespec is B:CURSOR.OBJ, LIB will remove the B: and the .OBJ, leaving only CURSOR. This becomes a module named CURSOR in the libary.

Minus sign (−)

Use the *minus sign*, followed by a module name, to delete a module from the library file. LIB then "closes up" the diskette space left empty by the deletion. This cleanup action prevents the library file from growing larger than necessary.

Remember that new modules, even replacement modules, are added to the end of the library file, not put into spaces vacated by deleted modules.

Asterisk (*)

Use the *asterisk*, followed by a module name, to extract a module from the library file and place it into a separate object file. The module will still exist in the library.

...continued

Operators (cont)

The module name is used as the filename. LIB adds the default drive designation and an extension of .OBJ. The drivespec and extension cannot be user-specified. However, once the object file is created, the user can rename the file, give it a different extension or copy it to a different disk.

Semicolon (;)

Use a single *semicolon*, followed immediately by a **Return**, at any time to select the default responses to the remaining prompts or command line.

This feature saves time by overriding the need to answer additional prompts.

Once the *semicolon* is entered, the command entry is terminated. You cannot use the semicolon to skip over parts of the LIB command entry process.

Ampersand (&)

Use the *ampersand* to extend the current line. This operator is only used when entering the Operation parameter.

The line length for a response is limited to the line length of the system. For a large number of responses, enter the *ampersand* at the end of the current line and press **Return**. This moves the cursor to the beginning of the next line, enabling you to continue entering the Operation parameter.

Ctrl+C

Entering *Ctrl+C* at any time aborts the current LIB session and returns you to the operating system.

Entering *Ctrl+C* during command entry (before pressing **Return** for that parameter) deletes the parameter entered and enables you to re-enter the parameter.

Table IV-D summarizes the LIB operators.

Table IV-D
THE LIB PROGRAM OPERATORS

OPERATOR	ACTION
Plus sign (+)	Appends an object file to the end of a library file.
Minus sign (−)	Deletes a module from a library file.
Asterisk (*)	Extracts a module from a library file, and places the module into an object file. The module is not deleted from the library.
Ampersand (&)	Enables the entry of an operation sequence that is longer than one physical line on the video display.
Semicolon (;)	Begins immediate execution of the LIB command. The system assumes default responses to any remaining prompts.
Ctrl+C	Aborts the LIB session, or deletes the current parameter being entered.

4.6 EXAMPLES

If the response to the LIB prompts are:

```
Library File: FUN
Operation: +CURSOR-HEAP+HEAP*FOIBLES&
Operation: *INIT+ASSUME+RIDE;
```

the system will delete the module HEAP; extract the two modules FOIBLES and INIT (creating two files called FOIBLES.OBJ and INIT.OBJ); then append the object files CURSOR, HEAP, ASSUME and RIDE. You may enter your operations parameter in any order, and you may use the ampersand as often as needed. No cross-reference listing is produced.

If the LIB command line is:

```
LIB A:BEGIN +MANCH-ALBERT*CRIMP BAILEY
```

the LIB program adds the module **MANCH** to the end of the library; deletes the module **ALBERT**; and extracts the module **CRIMP**, placing it into a file called **CRIMP.OBJ**. LIB also creates a cross reference listing and stores it in a file called **BAILEY**.

If contents of a response file called RESPONSE.TO is:

```
BAGAL
+COLLECT*BATCH*NUMBER-1
CROSS.ONE
```

and the command line LIB @RESPONSE.TO is entered, then the LIB program is called up to manipulate the library file BAGAL.LIB. The module COLLECT is added to the end of the library, and BATCH and NUMBER-1 are extracted and put into their own object files BATCH.OBJ and NUMBER-1.OBJ. A cross reference is created and stored in a file called CROSS.ONE.

4.7 ERROR MESSAGES

All errors cause the library session to abort. Therefore, after the cause is found and corrected, LIB must be rerun.

Table IV-E
LIB ERROR MESSAGES

<symbol> is a multiply defined PUBLIC. Proceed?

Cause: Two modules define the same public symbol. You are asked to confirm the removal of the definition of the old symbol.

Cure: Remove the PUBLIC declaration from one of the object modules and recompile or reassemble. If you respond "No", the library will be left in an indeterminate state.

Allocate error on VM.TMP

Cause: Out of disk space.

Cannot create extract file

Cause: No room in directory for extract file.

Cannot create list file

Cause: No room in directory for library file.

Cannot nest response file

Cause: @filespec in response (or indirect) file.

...continued

Table IV-E *(cont)*
LIB ERROR MESSAGES

LIB cannot open VM.TMP

Cause: There is no room for VM.TMP in disk directory.

Cannot write library file

Cause: Out of disk space.

Close error on extract file

Cause: Out of disk space.

Error: An internal error has occurred

Contact Bytec-Comterm Inc.

Fatal Error: Cannot open input file

Cause: You mistyped an object filename.

Fatal Error: No library file specified

Cause: You didn't specify a library.

Fatal Error: Cannot open response file

Cause: You didn't create the response file before using batched lib.

...continued

Table IV-E *(cont)*
LIB ERROR MESSAGES

Fatal Error: Module filename is not in the library

Cause: You tried to delete a module that is not in the library.

Input file read error

Cause: Bad object module or faulty disk.

Invalid object module/library

Cause: Bad object module and/or library.

Library disk is full

Cause: No more room on disk.

Listing file write error

Cause: Out of disk space.

No library file specified

Cause: No response to Library File prompt.

Read error on VM.TMP

Cause: Disk not ready for read.

...continued

Table IV-E *(cont)*
LIB ERROR MESSAGES

Symbol table capacity exceeded

Cause: Too many public symbols (about 30 K characters in symbols).

Too many object modules

Cause: More than 500 object modules.

Too many public symbols

Cause: 1024 public symbols maximum.

Write error on library/extract file

Cause: Out of disk space.

Write error on VM.TMP

Cause: Out of disk space.

Section 1

INTRODUCTION TO PART V

This part, *Part V, Useful Procedures*, is a description of some advanced techniques of Hyperion usage which may be useful.

Part V is organized according to the application:

— ***Section 2*** describes the batching of DOS commands. "Batching" is putting a series of commands in one file, which when activated, performs each command. Special commands to increase the flexibility of batch files are included.

— ***Section 3*** outlines the commands available to modify the Configuration File used by DOS when starting up the system.

— ***Section 4*** is an examples section with expanded descriptions of some Hyperion commands and functions. Examples of splitting files using EDLIN, concatenating files using /A and /B switches, and IBM DOS emulation are described.

Part V

Section 2

BATCHING DOS COMMANDS

Section 2

BATCHING DOS COMMANDS

2.1 GENERAL INTRODUCTION

After using the Hyperion DOS for a while, you may find yourself using the same sequences of DOS commands over and over. In such cases it may be useful for you to put all these DOS commands into a file and simply call up the file each time you want the sequence of DOS commands executed. Putting DOS commands into a file is called *batching* DOS commands.

The Batch File

The file into which these commands are put, is called a *batch file.*

You may assign any filename you wish to this batch file, as long as the name does not already exist on the diskette with the file extension COM or EXE. The file extension must always be .BAT. This is how the system recognizes that this is a batch file.

Typing in the filename of the batch file along with any parameters, and pressing **Return** causes the system to execute all of the commands stored in the batch file automatically. The batch file filename can therefore be considered as a "command", a command which you have created. (See "Executing a Batch".)

Inserting Remarks and Pauses

It is possible to insert remarks into a batch file, to provide useful feedback to the user. It is also possible to cause the batch process to pause at any point and wait for the user to hit any key before proceeding.

Inserting Parameters

Parts of the command lines contained in a batch file may be left incomplete. You would structure batch files in this way, if you wanted them to behave differently depending on the value substituted for the incomplete parts (variables) in the file.

When using a batch file containing variables, you can add a number of parameters to the command line. During execution, the system substitutes these parameters for the variables in the file.

The AUTOEXEC Batch File

Whenever a system restart is performed, the commands found in a file called AUTOEXEC.BAT are automatically entered into the system.

Since AUTOEXEC.BAT is a batch file, you can modify it to create your own AUTOEXEC, which would then be automatically executed at every system restart. Be careful when changing AUTOEXEC.BAT. If the commands it contains are not executed, the normal Hyperion operating environment may be changed.

2.2 BATCH FILE COMMANDS

There are 7 subcommands that you can use to make batch files more flexible. Described on pages V-9 to V-22, they are:

ECHO – Display Batch Commands While Executing
FOR – Use the FOR Loops in Batches
GOTO – Transfer Control to Another Line
IF – Use the IF Statement in Batches
PAUSE – Halt Processing to Display Message
REM – Display Remark from within File
SHIFT – Use Additional Parameters

2.3 CREATING A BATCH FILE

Batch files are created in the same way as any other text file, using either EDLIN or a text editor. You can also create batch files directly from the keyboard by using the COPY command:

STEP

1) Enter the following command line:

   COPY CON d:filename.BAT

   where *d:* is a drivespec, and *filename* is any legal filename. Warning: if filename.BAT already exists, the COPY command destroys the previous contents of the file when used in this way.

2) Type the DOS command lines that are to form this batch, exactly as you would type them normally for immediate execution. Press the **Return** key between each command. Editing on a line is identical to the line editing available in EDLIN in Insert mode. Once **Return** is pressed the line cannot be changed.

3) When all of the commands needed for the batch have been entered, press **Ctrl + Z** followed by **Return** on a new line to end the batch. The COPY can be aborted at any time by pressing **Ctrl + Break.**

The commands typed into the batch file "filename.BAT" can now automatically be executed at any time, by typing **filename** when DOS prompts for a command.

2.4 EXECUTING A BATCH

Once a ".BAT" file has beeen created, all of the commands it contains can be executed simply by typing the filename of the ".BAT" file on a command line. Only the filename, and not the ".BAT" extension, should be typed.

To execute a batch of DOS commands:

STEP

1) Type in the command line:

   filename *parameters*

2) Press the **Return** key.

Up to 9 *parameters* may be entered after the *filename*. A parameter is a value that is going to be substituted for variables previously entered into the batch file.

2.5 PASSING PARAMETERS

As mentioned earlier, it is often useful to create batch files that contain variables to be replaced at the time they are executed. For example, a useful batch file for diskette and file management purposes might be one that displays the contents of a file on the screen, and then deletes it only if the user so decides. The batch file would look like this:

```
TYPE %1
REM Enter Ctrl + Break to SAVE this file,
PAUSE or hit any other key to ERASE it.
ERASE %1
```

The batch file does not contain the name of the file to be typed and then optionally erased. Instead, it contains a reference to the first (%1) parameter on the command line that called up the batch. This batch file would be called up by typing:

```
filename TYPEFILE.TXT
```

The name of the file to be typed (TYPEFILE.TXT) would then replace each occurrence of %1 in the batch file.

This simple example only refers to one command line parameter, but in fact any batch file can refer to up to ten words on the command line. They are always referred to as %0, %1,...,%9. %0 is the actual filename, exactly as it was typed on the command line. %1 through %9 are up to nine command line parameters.

If you have 10 or more parameters, you can access them by using the SHIFT subcommand.

Occasionally, it may be necessary to have a real percent sign appear in a batch file. DOS assumes that percent signs are parameter references, unless they are typed twice:

```
REM This is a percent sign (%%).
```

During execution of a batch containing the above line, the remark would be displayed on the screen as

```
REM This is a percent sign (%).
```

EXAMPLE

The following batch file used to check all three drives (drive C can be either a RAM disk or a hard disk) for a file and types it if found. It provides a good example of a batch file that uses several of the batch file subcommands. The comments in brackets on the side are not part of the batch file, but are included to explain how the file works.

```
ECHO OFF                          (turn echo off)
IF NOT EXIST A:%1 GOTO CHKA
TYPE A:%1
GOTO DONE                         (type action done, go to end)
:CHKA                             (DOS jumps here from line two
IF NOT EXIST B:%1 GOTO CHKB         if file not found on drive A)
TYPE B:%1
GOTO DONE
:CHKC
IF NOT EXIST C:%1 GOTO NOFIND
TYPE C:%1
GOTO DONE
:NOFIND                           (if file isn't found, a message
ECHO Cannot find file %1            is displayed.)
:DONE
ECHO ON                           (all commands done, turn echo
                                    back on)
```

ECHO

Display Batch Commands While Executing

ENTERING THE SUBCOMMAND

STEP

1) Type in the word **ECHO.**

2) Type in the parameters, edit the command line if necessary, then press the **Return** key.

SUBCOMMAND DESCRIPTION

The ECHO batch processing subcommand causes DOS to display the commands executed in the batch file when ECHO is set to ON. ECHO does not affect messages generated by the command being executed in the batch file.

ECHO can also be used to have messages displayed while the commands in the batch file are being executed. Messages can be any string up to 123 characters long.

SUBCOMMAND FORMAT

ECHO [{ON}{OFF}{string}]

PARAMETERS

– If no parameters are entered, DOS displays the *current* ECHO value (ON or OFF).

ECHO
(cont)

string The *string of characters* you want displayed as a message. When DOS reaches the ECHO subcommand line, the message is displayed on the screen, even if ECHO is set to OFF.

EXAMPLE

This example shows how the ECHO subcommand works. The following file,

```
ECHO OFF
REM REMARKS ARE NOT DISPLAYED WITH ECHO = OFF
ECHO BUT MESSAGES AFTER "ECHO" ARE
ECHO ON
REM REMARKS ARE DISPLAYED WITH ECHO = ON
ECHO SO ARE MESSAGES AFTER "ECHO"
```

produces the following display:

```
ECHO OFF
BUT MESSAGES AFTER "ECHO" ARE
REM REMARKS ARE DISPLAYED WITH ECHO = ON
SO ARE MESSAGES AFTER "ECHO"
```

FOR
Use FOR Loop in Batches

ENTERING THE SUBCOMMAND

STEP

1) Type in the word **FOR**.
2) Type in the parameters, edit the command line if necessary, then press the **Return** key.

SUBCOMMAND DESCRIPTION

The FOR subcommand is used to allow repeated (iterative) processing of DOS commands. However, FOR subcommands cannot be nested (i.e. with more than one FOR subcommands on the same subcommand line. If filenames are entered as part of the FOR subcommand line, no path names will be accepted.

SUBCOMMAND FORMAT

FOR %%variable IN(set) DO command

PARAMETERS

%%variable During each iteration of a FOR loop, this *variable parameter* is set to a member of the list of files contained in the set you specify.

FOR
(cont)

set This parameter contains the names of the files on which DOS performs the *command* you specify in the FOR subcommand line.

command This *command* will be performed on every file named in the *set* you specify. Path names cannot be specified for the files, so all files must be in the current directory.

EXAMPLE

The following batch file,

```
FOR %%V IN (A B C) DO DIR %%V:
```

results in the following DOS commands being run:

```
DIR A:
DIR B:
DIR C:
```

GOTO
Transfer Control to Another Line

ENTERING THE SUBCOMMAND

STEP

1) Type in the word **GOTO**.

2) Type in the parameters, edit the command line if necessary, then press the **Return** key.

SUBCOMMAND DESCRIPTION

The GOTO batch processing subcommand is used to transfer control to another line in the program. In effect, DOS jumps from the GOTO subcommand line to the line containing the *label* you specify. DOS then continues executing the commands in the file, starting with the command in the line *following* the label.

A label is inserted into a batch file as a colon (:) followed by the label name. If more than one label is used in a file, all labels must be unique.

If no label is entered, DOS stops processing the commands in the batch file and sends you an error message (see following page).

However, labels that are not referenced by a GOTO subcommand are ignored by DOS and do not affect the execution of the batch file. This means that you can use label lines without references for your own comments in the batch file.

GOTO
(cont)

SUBCOMMAND FORMAT

GOTO label

PARAMETERS

label When DOS executes a GOTO subcommand, it searches for the line bearing this *label* preceded by a colon (:). If no *label* corresponding to the one in the subcommand line is found, DOS stops executing commands in the batch file and issues an ERROR MESSAGE. If a *label* is longer than eight characters, DOS issues an error message.

ERROR MESSAGE

If no label is provided, DOS stops executing the commands in the batch file and issues the following error message:

Label not found

If you receive such an error message, check the batch file you are using to make sure the label is present, and properly formatted.

IF
Use the IF Statement in Batches

ENTERING THE SUBCOMMAND

STEP

1) Type the word **IF**.

2) Enter the parameters, edit the command line if necessary, then press the **Return** key.

SUBCOMMAND DESCRIPTION

The IF subcommand is used when you want to set conditions on the execution of a command in a batch file. If the conditions are met, the command is executed, if not, DOS does not perform the command and goes on to the next line in the file.

SUBCOMMAND FORMAT

IF [NOT] condition command

PARAMETERS

NOT A *NOT condition* statement is true if the *condition* is false.

IF
(cont)

condition The *condition* parameter can be one of the following format:

a) ERRORLEVEL number

where *number* is the program exit code. ERRORLEVEL is true when the program has an exit code equal to or greater than *number*;

b) string1 = = string2

which is true when both strings of characters are identical;

c) EXIST filespec

which is true if *filespec* is the name of a file found on the specified drive. Path names are not allowed in using the IF subcommand.

command The *command* may be any DOS command except the IF subcommand. If the condition statement (including the *NOT*, if present) is true, DOS will execute the command.

PAUSE

Halt Processing to Display a Message

ENTERING THE SUBCOMMAND

STEP

1) Type the word **PAUSE**.

2) Enter the parameters, edit the command line if necessary, then press the **Return** key.

SUBCOMMAND DESCRIPTION

The PAUSE subcommand is used to suspend the execution of commands by DOS to allow you to perform certain user tasks such as exchanging diskettes between commands. When DOS encounters a PAUSE command, it halts processing and displays any string of characters you may have entered along with the subcommand (see SUBCOMMAND FORMAT below) and the message:

Strike a key when ready...

to prompt you to strike a key (any key except the **Ctrl + Break** combination, which ends processing) to recommence processing.

If you want to have the option of stopping the execution of a batch file, you can insert PAUSE subcommands at strategic points in the batch file along with messages indicating how much of the file DOS has completed. At each PAUSE, DOS gives you some time to decide whether or not to end processing. To stop processing simply enter the **Ctrl + Break** combination.

PAUSE
(cont)

SUBCOMMAND FORMAT

PAUSE [**string**]

PARAMETERS

string The string of characters you want displayed as a message when the PAUSE subcommand is executed.

EXAMPLE

The following batch file demonstrates two uses of the PAUSE subcommand:

```
:START
DIR A: ¦FIND "DIR">DIRECT.LST
TYPE DIRECT.LST
COPY DIRECT.LST + DIRSTOR.LST DIRSTOR.LST
PAUSE PRESS CTRL+C TO STOP SEARCH
DIR B: ¦FIND "DIR">DIRECT.LST
TYPE DIRECT.LST
COPY DIRECT.LST + DIRSTOR.LST DIRSTOR.LST
PAUSE PRESS CTRL+C TO STOP SEARCH
PAUSE REPLACE DISKETTES IN DRIVES A AND B
GOTO :START
```

This file uses the FIND command to locate any lines in the directory listing (generated by DIR) that refer to directories. These lines, with the caption <DIR> are then piped to the file DIRECT.LST. DIRECT.LST is then copied to DIRSTOR.LST using the COPY command. By displaying DIRECT.LST, you can see which, if any, directories are on the diskette being searched. DIRSTOR.LST provides a permanent record of all the directories found.

NOTE: This batchfile was designed to be executed from drive C (RAM disk or hard disk).

REM

Display a Remark From Within a File

ENTERING THE SUBCOMMAND

STEP

1) Type the word **REM**.

2) Enter the parameters, edit the command line if necessary, then press the **Return** key.

SUBCOMMAND DESCRIPTION

This subcommand is used to display a remark from within a batch file. Remarks can be any string up to 123 characters long.

SUBCOMMAND FORMAT

REM [string]

PARAMETERS

string The *string of characters* you want displayed as a message when DOS executes a REM subcommand. The *string* can be up to 123 characters long.

SHIFT

Use Additional Parameters

ENTERING THE SUBCOMMAND

STEP

1) Type the word **SHIFT**.

2) Enter the parameters, edit the command line if necessary, then press the **Return** key.

SUBCOMMAND DESCRIPTION

The SHIFT subcommand is used when more than 10 parameters are specifed with a batch command (see "Passing Parameters").

When you enter a SHIFT subcommand, each parameter variable is reassigned to the next parameter to the right in the list of parameters entered with the batch file command.

SUBCOMMAND FORMAT

SHIFT

SHIFT
(cont)

EXAMPLES

Example 1) *– If the parameter list were:*

A B C D E F G H I J K

the parameter variables **%0** to **%9** would correspond to the variable parameters as follows:

A	B	C	D	E	F	G	H	I	J	K
%0	%1	%2	%3	%4	%5	%6	%7	%8	%9	

If a **SHIFT** subcommand is entered, the variable parameters would correspond as follows:

A	B	C	D	E	F	G	H	I	J	K
	%0	%1	%2	%3	%4	%5	%6	%7	%8	%9

Part V

Section 3

MODIFYING THE CONFIGURATION FILE

Section 3

MODIFYING THE CONFIGURATION FILE

3.1 INTRODUCTION

Every time you start your system with DOS, it reads the configuration file (called CONFIG.SYS) to set certain parameters to define how the system works.

These parameters are pre-set by DOS (i.e. default values are given) to provide the optimum performance for most typical applications. However, there may be cases where some of these values should be changed to improve performance, or to allow certain software programs to run. These programs may reconfigure the system automatically, or provide instructions for you on which configuration parameters to change.

3.2 CONFIGURATION COMMANDS

There are also five commands that can be entered into the CONFIG.SYS file to change certain system parameters. The five commands are:

BREAK – *Check for Control Break during DOS Functions*
BUFFERS – *Set the Memory Buffer Size*
DEVICE – *Specify Name of Device Driver File*
FILES – *Specify Maximum Number of Open Files*
SHELL – *Specify Top-Level Command Processor*

All these commands, expect for BREAK, must be added to the CONFIG.SYS file for DOS to use them. If they are entered directly into the system, DOS displays an error message.

In addition to being directly entered in to the CONFIG.SYS file using a text editor, these commands (except the SHELL command) can be changed interactively using the MODE command.

If you change any of the commands in CONFIG. SYS, the changes take effect the *next* time DOS is started.

The one exception to this is the BREAK command. When entered directly into the DOS, it takes effect immediately.

BREAK

Check for Control Break during DOS Functions

ENTERING THE COMMAND

STEP

1) Type the word **BREAK**.

2) Enter the parameters, edit the command line if necessary, then press the **Return key**.

COMMAND DESCRIPTION

The BREAK command should only be used once in the configuration file. If BREAK is set to ON, DOS will check for a **Ctrl + Break** combination entered from the keyboard while performing any function in a program. This allows you to stop, or break out of, programs that have few accessess to the screen, keyboard or printer (running compilers, for example).

The default value is OFF, and this causes DOS to check for the **Ctrl + Break** combination entered at the keyboard *only* when DOS is performing operations involving the screen, keyboard, printer, or asynchronous communications adapter.

BREAK
(cont)

COMMAND FORMAT

BREAK = [{ON}{OFF}]

The value for BREAK for the current session can be changed after DOS has been started by entering the BREAK command in the following format:

BREAK = ON *or* **BREAK = OFF**

To permanently change the value for BREAK, you must add it to the CONFIG.SYS file.

If the BREAK command is entered without any parameters while the system is running, the current value of BREAK is displayed.

BUFFERS

Set the Memory Buffer Size

ENTERING THE COMMAND

STEP

1) Type the word **BUFFERS**.

2) Enter the parameters, edit the command line if necessary, then press the **Return** key.

COMMAND DESCRIPTION

The BUFFERS command is used to set the number of disk buffers that DOS will allocate in memory when starting up. A disk buffer is a block of memory DOS uses to hold data being written from or to a disk. The more buffers DOS has, the more data will be in memory. This data can be accessed faster than data that must be read from or written to sectors on the disk.

For applications that access records in a random fashion, having more buffers is a significant advantage. Examples of this are many BASIC programs or data base applications.

However, for applications doing sequential accesses (read an entire file, write an entire file, for example), there is little advantage in having a large number of buffers allocated.

BUFFERS
(cont)

The default number of buffers is 2, and this value remains in effect until DOS is started with a different BUFFERS value in the configuration file. If you are not doing a large amount of random accesses from files, this default value should be sufficient.

If you are running a lot of data base applications or other programs requiring a large number of random accesses, you may want to increase the number of DOS buffers. For most data base applications, a buffer size between 10 and 20 provides good results.

Each extra buffer decreases the amount of random-access memory (RAM) available in your computer by 528 bytes. Having too many disk buffers will reduce the amount of memory available for the applications you run and may in some cases actually slow down the speed at which the application runs.

If your Hyperion has a hard disk attached, it is recommended that you set the BUFFERS value to 3 for general use (i.e. without a large number of random accesses).

COMMAND FORMAT

BUFFERS = xx

PARAMETERS

xx The maximum number for **xx** is 99.

DEVICE

Specify Name of Device Driver File

ENTERING THE COMMAND

STEP

1) Type the word **DEVICE**.

2) Enter the parameters, edit the command line if necessary, then press the **Return** key.

COMMAND DESCRIPTION

The DEVICE command is used to specify the name of a file containing the device driver. When starting DOS, this file is loaded into memory as an extension of itself and gives it control.

The standard device drivers loaded by DOS support the standard screen, keyboard, printer, diskette and fixed disk devices and an auxiliary device. You do not need to use the DEVICE command to use any of these devices.

If you are writing programs for applications that need to have device drivers loaded by DOS when starting, include a DEVICE command in the CONFIG.SYS file to load each driver.

DEVICE
(cont)

COMMAND FORMAT

DEVICE [d:][path][file]

PARAMETERS

– If no parameters are entered, DOS searches the names specified on the previous PATH command (i.e. the search path currently defined to DOS).

d: The *drivespec* of the drive on which the device being listed is located. If no *drivespec* is given, DOS assumes the default drive.

path The *path* defining the device. The *path* consists of a list of directory names separated by back-slashes (\), needed to tell DOS the location of the device.

file The *filename* and *extension*, if any, of the device driver.

FILES

Specify Maximum Number of Open Files

ENTERING THE COMMAND

STEP

1) Type the word **FILES**.

2) Enter the parameters, edit the command line if necessary, then press the **Return** key.

COMMAND DESCRIPTION

The FILES command is used to set the number of files you can have open simultaneously. DOS creates a 16-bit handle for each file it opens or creates.

This command has no effect on files opened using file control blocks (e.g. OPEN FCB).

The maximum number of files a process can have open at the same time is 15, in addition to the 5 system-defined handles used for standard input, standard output, error, auxiliary and standard printer.

FILES
(cont)

The default value for the number of open files (and handles) is 8. This is sufficient for most cases. However, if your applications are sending error messages indicating an insufficient number of handles, the FILES command can be used to increase the number of handles for DOS.

For each extra file opened above the 8 default files, the size of the resident portion of DOS increases by 39 bytes.

COMMAND FORMAT

FILES = xx

PARAMETERS

xx The maximum number for *xx* is 99.

SHELL

Specify Top-Level Command Processor

ENTERING THE COMMAND

STEP

1) Type the word **SHELL**.

2) Enter the parameters, edit the command line if necessary, then press the **Return** key.

COMMAND DESCRIPTION

The SHELL command is used to specify the name and location of a top-level command processor that DOS initialization will load in place of COMMAND.COM when you start your system.

You must provide your own command processor file to use with the SHELL command.

COMMAND FORMAT

SHELL[d:][path][file]

SHELL
(cont)

PARAMETERS

d: The *drivespec* of the drive on which the command processor will be located.

path The *path* defining the directory in which the command processor will be located. The *path* consists of a list of directory names, separated by backslashes (\), needed to tell DOS the location of the directory.

file The *filename,* and *extension*, if any, of the command processor.

Part V

Section 4

SAMPLE PROCEDURES

Section 4

SAMPLE PROCEDURES

4.1 SPLITTING FILES USING EDLIN

Files to be edited may exceed the memory capability of drive C, and therefore cannot be edited using the Hyperion Text Editor. In order to use the Hyperion Text Editor, such files must be split into smaller portions. EDLIN can be used to split the files.

Assume you have a file you want to edit, a master DOS diskette in drive A, and the Hyperion powered on.

STEP

1) Insert the diskette containing the file you want to edit into drive B, if your Hyperion has two drives. If your Hyperion has a single drive, remove the master DOS diskette from drive A and replace it with the diskette containing the file you want to edit.

2) Enter the command:

   EDLIN *d: file*

   where *d:* is the *drivespec*, and *file* is the *filename* and *extension*, for example, EDLIN B:BIGFILE.TXT.

3) Enter # to set the current line to the end of the file.

... continued

Splitting files using EDLIN (cont)

STEP *(cont)*

4) Then enter:

L

This displays the last 11 lines in your file. The system prompt, an asterisk (*), appears at the end of the file.

5) Assume your file is 700 lines long. List the lines 325-375 to find a logical break in the text:

325,375 L

Assume a logical break occurs on line 349. The file can be split into two sections. The first section lines 1 to 349. The second section lines 350-700.

6) To create your first section, enter:

350,700 D

This reduces your file to lines 1 to 349, by deleting (D) lines 350 to 700.

7) To save this file to diskette, enter:

E

This automatically saves your smaller file under the filespec you input, creates a backup of your original input file by changing the extension to .BAK, and returns you to DOS.

....continued

Splitting files using EDLIN (cont)

STEP *(cont)*

8) Use the **DIR/P** command to list the files on your diskette. Your directory should show:

```
Volume in drive B has no label
Directory of B:\

      :
BIGFILE  TXT  17500  4-26-84  4:58p
BIGFILE  BAK  35000  3-29-84  2:24p
      :
```

(This is an estimate of the number of bytes you may have. The number of bytes per line depends on the number of characters per line and is, therefore, impossible to gauge exactly.)

You should rename your .BAK file to a different name, to ensure that you have a backup of the complete unsplit file, in case errors occur during the splitting process using EDLIN.

STEP

9) To rename your .BAK file enter:

```
RENAME [d:]file1 file2
```

where *file1* is the *old filename* and *extension* and *file2* is the new *filename* and *extension*. In our example, file1 is BIGFILE.BAK and file2 is 2NDHALF.TXT.

You need to repeat the entire process (Steps 2 to 7) to create the second, smaller file, which consists of lines 350 - 700 and contains the end of your large file. This time you would delete lines 1 - 349 in step 7.

If a file is very large it may be split into more than two sections. The procedures are the same; you simply need to adjust the line deletions accordingly.

4.2 CONCATENATING FILES USING /A AND /B SWITCHES

The **COPY command can be used for concatenating files while copying. An ASCII file is a file of characters ended by a Ctrl + Z** (1A hex), which is interpreted as an end of file character. A binary file contains any characters and may contain an embedded **Ctrl + Z**. The /A and /B switches (parameters to the copy command) are useful when concatenating files. They are used to override, or set, an end of file interpretation (**Ctrl + Z**) in binary or ASCII files.

The /A or /B switch after COPY on the command line defines the default file type (/A for ASCII, /B for binary), of all the source files. If no switch is specified then /A is assumed. In the list of source files, /A or /B is used after the filespec to override the default file type.

EXAMPLE 1:

To concatenate binary files enter:

```
COPY/B A.BIN + B.BIN NEW.BIN
```

This results in concatenating file A.BIN and file B.BIN to produce the file NEW.BIN.

EXAMPLE 2:

To concatenate ASCII files enter:

```
COPY/A A.ASC + B.ASC NEW.ASC
```

This results in concatenating file A.ASC and file B.ASC to produce the file NEW.ASC. (***Note:*** in actual practice it is not necessary to use the /A switch when concatenating ASCII files. If no switch is specified the system assumes /A.)

EXAMPLE 3:

To concatenate binary and ASCII files enter:

```
COPY A.BIN/B + A.ASC/A NEW.ASN
```

This results in concatenating file A.BIN and A.ASC to produce the file NEW.ASN.

4.3 IBM DOS EMULATION

There are two files delivered on the master DOS diskette. These files are called **SETIBM.BAT** and **SETHYP.BAT.** These are batch files. You can use these files to emulate an IBM DOS environment or set the Hyperion DOS environment. Setting an IBM DOS environment is useful when running software created for an IBM PC. The softkeys and other standard features on the Hyperion are not standard for the IBM PC and may interfere with the operation of IBM PC software.

To Emulate an IBM PC

STEP

1) Place your system diskette in drive A, type in the command **SETIBM** and press **Return**.

This calls up the MODE command to removes all the extra Hyperion features. When the command terminates, your Hyperion will respond to commands like an IBM PC, and the screen display will be the same as that for an IBM PC.

The MODE settings are saved to the system diskette in drive A, and the next time you boot the Hyperion with that diskette, the Hyperion will automatically enter the IBM PC environment.

To Reset Your Hyperion to Its Normal Environment

STEP

1) Place your system diskette in drive A, type in the command **SETHYP**, and press **Return**.

This calls up the MODE command to restore all the extra Hyperion features.

The MODE settings are saved to the system diskette in drive A.

Appendix A

ENHANCEMENTS TO DOS VERSION 2.11

Appendix A

ENHANCEMENTS TO DOS VERSION 2.11

A.1 NEW DOS COMMANDS

There are a number of new commands in this version of DOS. They are described in Part II:

– ASSIGN	– BACKUP	– CHDIR
– CLS	– CTTY	– FCOMP
– GRAPHICS	– LOCK	– MKDIR
– PATH	– PRINT	– PROMPT
– RECOVER	– RESTORE	– RMDIR
– SET	– TREE	– UNLOCK
– VER	– VERIFY	– VOL

as well as Batch commands (ECHO, IF, SHIFT, and GOTO) and Configuration commands (BREAK, BUFFERS, DEVICE, FILES, SHELL). The Batch and Configuration commands are described in Part V. Another new command, FDISK, is described in Part I, Section 9.

A.2 ENHANCED DOS COMMANDS

Some of the commands found in earlier versions of DOS have been enhanced as well. They are described in Part II.

– CHKDSK	– DEBUG	– DIR
– DISKCOPY	– DISKCOMP	– EDLIN
– ERASE	– FORMAT	

A.3 HARD DISK SUPPORT

In order to help provide users with a solution to their storage requirements, DOS 2.11 supports systems using hard disks. Up to two hard disks can be installed in an expansion unit attached to your Hyperion. You can create partitions in them allowing you to use other operating systems as well as DOS.

A.4 INCREASED DISKETTE STORAGE

Unlike earlier versions of DOS, version 2.11 normally formats diskettes with 9 sectors per track, instead of 8. This increases the amount of storage space on the diskette by about 40,000 characters. Diskettes formatted with 9 sectors per track must not be used with earlier versions of DOS. If you wish to format diskettes with DOS 2.11 for use with earlier DOS versions, you must use the /8 option to format them with 8 sectors per track.

A.5 ENHANCED FLEXIBILITY IN USING COMMANDS

In addition to offering new commands, DOS 2.11 also provides a number of features to enhance the usefulness and flexibility of DOS commands.

Redirection of Standard Input and Output: This feature allows you to "redirect" the output from a DOS command to some device other than the standard output device (the screen). As well, it lets DOS commands use input from a device other than the standard input device (the keyboard). For more information, see Part I, Section 10.

Piping Standard Input and Output: The redirection of standard input and output feature descibed above can also be used to use the output of one command as the input for another. This feature, called "piping" is also described in Part I, Section 10.

Redefining the Standard Input/Output Console: By using the CTTY command, you can designate an external terminal as the standard input/output console to replace the Hyperion keyboard and screen.

New Devices Recognized by DOS: DOS now recognizes the following device names:

– LPT2 – LPT3 – COM2

A.6 FILE MANAGEMENT FEATURES

This version of DOS features a new filing structure to help users keep track of a large number of files. With DOS 2.11, files are contained in directories. Directories can contain other directories as well, allowing you to create a hierarchy of directories to suit your particular organization. Since this organization of directories branches out from a root directory, the directory hierarchy is described as "tree-structured."

DOS 2.11 also provides a defence against inadvertent erasure or modification of files with the LOCK command. A "locked" file can be copied, accessed for editing or display, but not changed or deleted. To change or delete a locked file it must be "unlocked" with the UNLOCK command.

A.7 MODIFIABLE CONFIGURATION FILE

DOS checks the CONFIG.SYS file every time it starts the system to set certain configuration parameters. You can modify these parameters by using the configuration commands: BREAK, BUFFERS, DEVICE, FILES and SHELL.

Appendix B

TECHNICAL SPECIFICATIONS

Appendix B

TECHNICAL SPECIFICATIONS FOR THE HYPERION

ATTRIBUTE	SPECIFICATION

DISKETTE DRIVES:

* One or two 5-1/4″ double density dual sided diskette drives each with 368,640 bytes of storage capacity;
* Capable of reading and writing IBM PC 5-1/4″ single and dual sided diskettes;

DISPLAY SYSTEM:

* 7″ non-glare CRT with amber phosphor for viewing comfort;
* Separate brightness and contrast controls;
* Alphanumeric screen format: 25 lines of 80 characters (4 separate display pages);
* character set: 256 different characters including Greek alphabet, foreign language special characters, special mathematical and line drawing symbols;
* Typeface: 6 × 7 dot matrix character in an 8 × 10 or 8 × 8 dot box with 2-dot descenders;
* Character attributes: blink, intensify, reverse video, double size (software selectable on a per character basis);
* Bidirectional scrolling;
* Soft key labels on 25th display line for 10 function keys;
* Graphics display format: 640 dots wide × 200 dots high fully addressable array, or 320 dots wide × 200 dots high fully addressable array with 4-level grey scale;
* 200 dot resolution graphics provided for compatibility with IBM PC;
* Refresh rate: 70 Hz (200-dot resolution), 60 Hz (alphanumeric and 250 dot resolution);
* CRT display selectable to turn off when not in use to prolong life.

...continued

Technical Specifications ***(cont)***

ATTRIBUTE	SPECIFICATION

ELECTRICAL:

* 120/240 volts AC + 12%-25%, 45 to 65 Hz operation, selectable via external switch;
* Power consumption 90 watts maximum;
* Front-mounted illuminated power switch;
* Meets applicable CSA, UL and FCC standards.

KEYBOARD:

* Low profile ergonomic design (meets European DIN standard);
* Compatible with IBM PC keyboard layout (84 keys including 10 function keys and numeric keypad);
* Tactile feel with optional audio key click;
* Long life capacitance keyswitches with N-key roll over;
* Detached – may be used up to 4 feet (1.2 metres) from main unit;
* Stows in main unit when not in use or for travelling.

MECHANICAL:

* Size – 18.3 inches (46.4 cm) wide, 11.3 inches (28.8 cm) deep, 8.8 inches (22.3 cm) high;
* Weight 21 lbs (9.6 kg);
* Portable – built-in carrying handle;
* Attractive soft vinyl travelling case with accessory pockets.

MEMORY:

* 256 K byte user RAM with parity;
* 20 K byte display RAM;
* 8 K byte ROM containing automatic power-up diagnostics, machine initialization and general I/O routines.

...continued

Technical Specifications ***(cont)***

ATTRIBUTE	SPECIFICATION

MICROPROCESSOR:

* Intel 8088 16-bit processor (4.77 MHz clock rate);
* Optional 8087 floating point processor;

MODEM:

* Optional 300 baud modem with auto-answer capability, Bell 103J compatible;
* Auto-dial using either touch tone (DTMF) or dial pulse;
* Direct connection to modular telephone jack in series with telephone set;
* Optional acoustic cups for use when modular telephone jack not available.

PARALLEL PRINTER PORT:

* Compatible with IBM PC printer or Epson/Centronics printers.

SERIAL I/O PORT:

* Meets RS-232C and RS-423 standards;
* Asynchronous 110 to 19.2 K baud with programmable baud rate, parity, stop bits and data bits;

SOFTWARE:

* Microsoft MS-DOS (same as IBM PC);
* Microsoft Advanced Disk BASIC interpreter with graphics support (same as IBM PC);
* Optional Microsoft programming languages: BASIC compiler, COBOL, FORTRAN and PASCAL.

...continued

Technical Specifications ***(cont)***

ATTRIBUTE	SPECIFICATION
OTHER FEATURES:	
*	Programmable sound system;
*	Composite video output jack for connecting external video monitor;
*	I/O expansion connector for attaching an expansion unit (e.g. for winchester hard disk drive).

Appendix C

COMTERM PROGRAM LICENSE AGREEMENT

Appendix C

COMTERM PROGRAM LICENSE AGREEMENT

YOU SHOULD CAREFULLY READ THE FOLLOWING TERMS AND CONDITIONS BEFORE OPENING THIS SOFTWARE PACKAGE. OPENING THIS SOFTWARE PACKAGE INDICATES YOUR ACCEPTANCE OF THESE TERMS AND CONDITIONS. IF YOU DO NOT AGREE WITH THEM, YOU SHOULD PROMPTLY RETURN THE PACKAGE UNOPENED AND YOUR MONEY WILL BE REFUNDED.

Comterm Inc. provides this program and licenses its use in the United States and Canada. You assume responsibility for the selection of the program to achieve your intended results obtained from the program.

LICENSE

You may:

a) use the program on a single Comterm microcomputer;

b) copy the program into any machine readable or printed form for backup or modification purposes in support of your use of the program on the single Comterm microcomputer (Certain programs, however, may include mechanisms to limit or inhibit copying. They are marked "copy protected".);

c) modify the program and/or merge it into another program for your use on the single Comterm microcomputer. (Any portion of this program merged into another program will continue to be subject to the terms and conditions of this Agreement.); and,

d) transfer the program and license to another party if the other party agrees to accept the terms and conditions of this Agreement. If you transfer the program, you must at the same time either transfer all copies whether in printed or machine-readable form to the same party or destroy any copies not transferred; this includes all modifications and portions of the program contained or merged into other programs.

You must reproduce and include the copyright notice on any copy, modification or portion merged into another program.

YOU MAY NOT USE, COPY, MODIFY, OR TRANSFER THE PROGRAM, OR ANY COPY, MODIFICATION OR MERGED PORTION, IN WHOLE OR IN PART, EXCEPT AS EXPRESSLY PROVIDED FOR IN THIS LICENSE.

IF TRANSFER POSSESSION OF ANY COPY, MODIFICATION OR MERGED PORTION OF THE PROGRAM TO ANOTHER PARTY, YOUR LICENSE IS AUTOMATICALLY TERMINATED.

TERM

The license is effective until terminated. You may terminate it at any other time by destroying the program together with all copies, modifications and merged portions in any form. It will also terminate upon conditions set forth elsewhere in the Agreement or if you fail to comply with any term or condition of the Agreement. You agree upon such termination to destroy the program together with all copies, modifications and merged portions in any form.

LIMITED WARRANTY

THE PROGRAM IS PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU (AND NOT COMTERM INC. OR AN AUTHORIZED COMTERM COMPUTER DEALER) ASSUME THE ENTIRE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION. SOME STATES OR PROVINCES DO NOT ALLOW THE EXCLUSION OF IMPLIED WARRANTIES, SO THE ABOVE EXCLUSION MAY NOT APPLY TO YOU. THIS WARRANTY GIVES YOU SPECIFIC LEGAL RIGHTS AND YOU MAY ALSO HAVE OTHER RIGHTS WHICH VARY FROM STATE TO STATE OR PROVINCE TO PROVINCE.

Comterm Inc. does not warrant that the functions contained in the program will meet your requirements or that the operation of the program will be uninterrupted or error free.

However, Comtern Inc. warrants the diskette(s) on which the program is furnished, to be free from defects in materials and workmanship under normal use for a period of ninety (90) days from the date of delivery to you as evidenced by a copy of your bill of sale.

LIMITATIONS OF REMEDIES

Comterm Inc.'s entire liability and your exclusive remedy shall be:

1) the replacement of any diskette(s) not meeting Comterm Inc.'s "Limited Warranty" and which is returned to Comterm Inc. or an Authorized Comterm Computer Dealer with a copy of your bill of sale, or

2) if Comterm Inc. or the dealer is unable to deliver a replacement diskette(s) which is free of defects in materials or workmanship, you may terminate this Agreement by returning the program and your money will be refunded.

IN NO EVENT WILL COMTERM INC. BE LIABLE TO YOU FOR ANY DAMAGES, INCLUDING ANY LOST PROFITS, LOST SAVINGS OR OTHER INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE SUCH PROGRAM EVEN IF COMTERM INC. OR AN AUTHORIZED COMTERM COMPUTER DEALER HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES, OR FOR ANY CLAIM BY ANY OTHER PARTY.

SOME STATES OR PROVINCES DO NOT ALLOW THE LIMITATION OR EXCLUSION OF LIABILITY OF INCIDENTAL OR CONSEQUENTIAL DAMAGES SO THE ABOVE LIMITATION OR EXCLUSION MAY NOT APPLY TO YOU.

GENERAL

You may not sublicense, assign or transfer the license or the program except as expressly provided in this Agreement. Any attempt otherwise to sublicense, assign or transfer any of the rights, duties or obligations hereunder is void.

This Agreement will be governed by the laws of the Province of Ontario, Canada.

Should you have any questions concerning this Agreement, you may contact Comterm Inc. by writing to Customer Support Division, Comterm Inc., 8 Colonnade Road, Ottawa, Ontario, Canada, K2E 7M6.

YOU ACKNOWLEDGE THAT YOU HAVE READ THIS AGREEMENT, UNDERSTAND IT AND AGREE TO BE BOUND BY ITS TERMS AND CONDITIONS. YOU FURTHER AGREE THAT IT IS THE COMPLETE AND EXCLUSIVE STATEMENT OF THE AGREEMENT BETWEEN US WHICH SUPERCEDES ANY PROPOSAL, OR PRIOR AGREEMENT, ORAL OR WRITTEN, AND ANY OTHER COMMUNICATIONS BETWEEN US RELATING TO THE SUBJECT MATTER OF THIS AGREEMENT.

SOFT KEY LINES

DOS SOFT KEY LINE

F1	**LASTLN**	Display last command line entered.
F2	**Disks**	Display DISKS soft key line.
F3	**Files**	Display FILES soft key line.
F4	**MODE**	Set system configuration.
F5	**DIR/P**	List files and directories on a disk.
F6	**CHDIR**	Change current directory.
F7	**PATH**	Search directories for DOS commands.
F8	**TREE**	Display all directories.
F9	**EXPLAIN**	Show command features.
F10	**HELP**	Display HELP screens.

FILES SOFT KEY LINE

F1	**Dos**	Display DOS soft key line
F2	**Disks**	Display DISKS soft key line.
F3	**TYPE**	Display file contents.
F4	**DATE**	Display system date.
F5	**DIR/P**	List files and directories on a disk.
F6	**COPY**	Copy a file.
F7	**PRINT**	Print files.
F8	**ERASE**	Erase a file.
F9	**RENAME**	Rename a file.
F10	**HELP**	Display HELP screens.

DISKS SOFT KEY LINE

F1	**Dos**	Display DOS soft key line.
F2	**D-NAME**	Display diskname or rename a disk.
F3	**1Files**	Display FILES soft key line.
F4	**DATE**	Display system date.
F5	**DIR/P**	List files and directories on a disk.
F6	D-COPY	Duplicate a diskette
F7	**D-COMP**	Compare two diskettes.
F8	**FORMAT**	Format a disk.
F9	**CHKDSK**	Check and display disk status.
F10	**HELP**	Display HELP screens.

KEYBOARD IN DOS

KEY	FUNCTION
Esc	Cancel the command line, and redisplay system prompt.
Break	Used in conjunction with other keys to modify their use.
Ctrl	Used in conjunction with other keys to modify their use.
Alt	Used in conjunction with other keys to modify their use.
Tab	Type in a blank space.
Caps Lock	Lock the keyboard into upper or lower case mode.
Shift	Modify other key functions, or enter upper case mode.
Num Lock	Lock the number pad into number mode or function mode.
Rub Out	Backspace over and delete previous character entered
Return	Submit the command line to be processed in DOS.
Print	Used with other keys to print screen contents.
Home	Move the cursor to the beginning of the command line.
End	Move the cursor to the end of the command line.
Ins	Insert a space at the cursor, shifting one space right.
Del	Delete character at cursor, shifting one space left.
←	Move the cursor to the left one position.
→	Move the cursor to the right one position.

PECIAL KEY COMBINATIONS IN DOS

EYS	FUNCTION
trl + ←	Move the cursor to start of previous word.
trl + →	Move the cursor to the start of the next word.
trl + Break	Interrupt operations and return system to prior level.
trl + Alt + Del	Reboot (restart) the system.
hift + Print	Print out contents of screen.
trl + Print	Print out the screen display.

EDLIN COMMANDS

NTRALINE COMMANDS

EY	FUNCTION
1	Copy one character from the template to the edit line.
2	Copy all characters from the template to the edit line up to "x".
3	Copy all characters in template to edit line.
EL	Do not copy (skip over) a character in the template.
4	Do not copy (skip over) the characters in the template up to "x".
SC	Void the current input; leaves the template unchanged.
NS	Insert mode; inserts characters within a line.
5	Make the edit line the new template.

NTRALINE COMMANDS

OMMAND	FUNCTION
line>	Displays the line indicated.
	Append lines from disk file to memory.
	Copy a block of text to a specified point.
	Delete lines from file.
	End editing session and save changes to disk file.
	Insert new text lines into file.
	List text lines.
[	Move a block of text to a specified point.
	Quit editing; changes made in this session are not saved.
	Replace one text string with another.
	Search text for a text string.
	Transfer text from a file to the current file.
'	Writ e lines from file to disk.

ARAMETERS	FUNCTION
nteger	Indicates line number (number less than 65534).
eriod (.)	Indicates current line number.
ctothorpe (#)	Indicates next after last line.
eturn	Use default value appropriate to command.

DOS COMMANDS

COMMAND	PARAMETERS	FUNCTION
ASSIGN	[d1]=[d2]	Assign drives for disk operations.
BACKUP	[d1:][path][file] [d2:] [/S][/M][/A][/D:mm-dd-yy]	Back up hard disk files to diskettes.
CHDIR	[[d:]path]	Change the current directory.
CHKDSK	[/F][/V][d:][file]	Check/display disk status.
CLS		Clear the display screen.
COPY	[{/A}{/B}][/V][d:][path]file1 [d:][path][file2][{/A}{/B}]	Copy contents of one file to another.
CTTY	devicename	Change input/output console
DATE	[mm-dd-yy]	Display/change system date.
DEL	[d:][path]file	Erase a file from a disk.
DIR	[/P][/W][d:][path][file]	List files and directories on a disk.
DISKCOMP	[d1:] [d2:]	Compare two diskettes.
DISKCOPY	[sourced:] [targetd:]	Duplicate a diskette.
DISKNAME	[d:][newname]	Display/change disk name.
ERASE	[d:][path][file	Erase a file from a disk.
EXE2BIN	[d1:][path]file[d2:][file]	Convert .EXE files to .COM files.
EXPLAIN	[command or feature]	Display information about a feature.
FIND	[/V][/C][/N] "string" [d:] [path]file	Search files for specified text.
FORMAT	[/B][/V][/S][/1][/8][d:]	Prepare new disk for use.
GRAPHICS		Print a graphics display.
LOCK	[d:][path]file	Lock files to prevent erasure.
MKDIR	[[d:]path]	Create a new subdirectory.
MODE	(various - see Part II)	Set system configuration.
MORE	<[d:][path][file]	Read a screenful of data.
PATH	[d:]path[;[d:]path]...]	Search directories for commands.
PRINT	[[d:]file] [/T][/C][/P]...]	Print files while doing other tasks.
PROMPT	[prompt-text]	Set a new system prompt.
RECOVER	[d:][path][file]	Recover files from defective disk.
RENAME	[d:][path]file1 file2	Rename a file.
RESTORE	d1: [d2:][path][file][/S][/P]	Restore files from diskettes to hard disk.
RMDIR	[d:]path	Remove a subdirectory.
SET	[name=[parameter]]	Insert text strings in the command processor environment.
SORT	[/R][/+n] [<[d:][path][file1]] [>[d:][path][file2]]	Sort text files.
SYS	d:	Transfer MS-DOS system files to designated drive.
TIME	[hh:mm:ss]	Display/change system time.
TREE	[d:][/F]	Display all directories
TYPE	[d:]file	Display file contents.
UNLOCK	[d:][path]file	Unlock a locked file.
VER		Display DOS version number.
VERIFY	[{ON}{OFF}]	Verify data is properly written.
VOL	[d:]	Display disk (volume) name.

LINK/LIB

COMMAND	PARAMETERS
LINK	<object list>[/switch(es)],[run file][/switch(es)], [list file][/switch(es)],[library list][/switch(es)]
LIB	<library file> <operations> [list file]

Object list = filename and extension of source file.
Switches = /DSALLOCATE, /HIGH, /LINENUMBERS, /MAP, /PAUSE, /STACK
Run file = destination file.
Library list = list of library files.
Operations: plus sign (+) appends module; minus sign (-) deletes module; asterisk (*) extracts module; semicolon (;) begins immediate execution; Ctrl + C aborts execution.

DEBUG

COMMAND	PARAMETERS	FUNCTION
A	[address]	Assemble statements into memory.
C	<range> <address>	Compare portions of internal memory.
D	[range]	Display portions of internal memory.(Dump)
E	<address> [list]	Enter individual byte values into memory.
F	<range> [list]	Fill a range of addresses.
G	[=address] [address]	Execute program (Go).
H	<value> <value>	Perform hexadecimal arithmetic.
I	<value>	Read a port address and display byte found.
L	[address] [drive] [record] [record]	Load file into memory.
M	<range> <address>	Move a block of memory.
N	[filename][filename...]	Name a file and assign parameters.
O	<value> <byte>	Output byte to port.
Q		Exit from DEBUG (Quit).
R	[register-name]	Display contents of registers
S	<range> <list>	Search range for list of bytes.
T	[=address] [value]	Execute an instruction and display CPU register contents (Trace).
U	[range]	Display source statements corresponding to memory contents (Unassemble).
W	[address] [drive] [record] [record]	Write file in memory to disk.

address	Memory address location: segment:offset.
byte	2-digit hex value.
drive	Diskette drive specification.
list	List of bytes each delimited by a blank space.
range	Memory address range: address address; or address L address.
record	Logical record number or number of sector on disk.
string	Character string enclosed in quotes.
value	Hex value of up to 4 digits.

INDEX

HYPERION DOS 2.11 GUIDE

INDEX **Page**

D

INDEX **Page**

E

INDEX **Page**

M

INDEX **Page**

N

O

P

Q

R

INDEX **Page**

S

T

U

V

W