1500 DISK OPERATING SYSTEM DOS.H

User's Guide

Version 2 (Upgraded to 2.5.1)

May, 1980

Document No. 50308

DATAPOINT

1500 DISK OPERATING SYSTEM
DOS.H

User's Guide

Version 2 (Upgraded to 2.5.1)

May, 1980

Document No. 50308

PREFACE

The purpose of this User's Guide is to provide the user of the Datapoint 1500 DOS that information required to generate a system, make effective use of the available commands, and to make user-written programs compatible with the DOS.

The Datapoint 1500 processor has a different architecture from previous Datapoint processors (1100, 2200, 5500, etc). DOS.H is designed to minimize these differences from the user's point of view while providing capabilities consistent with the features of the new processor. Wherever possible, DOS.H commands are identical to corresponding commands in other Datapoint "Dot-series" operating systems (DOS.A, DOS.B, etc.).

The 9320 disk system referred to in this User's Guide is Datapoint's cartridge disk system with a Four-Port Communications Adapter. Not all cartridge disk systems include the Communications Adapter, and those where it is not included are designated as 9310 cartridge disk systems. All references herein to the 9320 disk are valid for the 9310 disk as well.

TABLE OF CONTENTS

CHAPTER 1. INTRODUCTION

Datapoint Corporation`s disk operating systems (DOS) are comprehensive systems of facilities for sophisticated data management.

DOS provides the operator with a powerful set of system commands by which the operator can control data movement and data processing from the system console. These commands allow the system operator to accomplish tasks that often are substantially more difficult on other computing systems. Sorting a large file, for instance, can be accomplished in one single command line. In spite of the simplicity of operation, a wide range of features is provided in the DOS facilities package.

To the user, DOS offers a set of facilities to simplify and generalize his task and file management . Concepts like dynamic disk space allocation allow programs to operate efficiently without regard to the amount of space required for the data files being used. In addition, the disk file structure used by DOS allows for direct random access. DOS also makes use of fully space-compressed text files.

These features, combined with the ability to support up to one million bytes of diskette storage and up to 40 million bytes of cartridge disk storage, provide a wide range of data processing and intelligent data entry capabilities.

1.1 Planning for DOS.H

There are three basic configurations for the DOS.H 1500 Disk Operating System:

--32K with two or four diskette drives
--64K with two or four diskette drives
--64K with two or four diskette drives and one to four 9320 cartridge disk drives.

A serial printer may be connected to any of these system configurations, and is of great value in a business data processing system that requires hardcopy reporting. The four-diskette-drive system is a must if larger file storage and greater disk I/O is desired, and the 9320 cartridge disks, with their larger storage capacity and greater speed will greatly enhance system performance.

Greater memory is a consideration when running SORT and other programs concurrently, editing large files, and for overall system efficiency. Note that 9320 disks are supported only on 64K processors.

If the system is configured with only diskettes, then the number of drives needed should be considered. Use of a single diskette drive is possible, but the user should at least have access to a double-drive system for backup purposes. Some consideration must be given to the question of copying files from one diskette to another. Since DOS.H software is distributed on diskettes from Datapoint Corporation, the user will need to have at least one two-drive system to copy the software from the diskette received from Datapoint to his working diskette(s). Single-drive systems should be considered only by those intending to use them as satellite systems--that is, as data-entry stations only--and, in this case, there should be at least one other system with two or more drives on which to develop programs, create diskette backup, and perform file processing.

1.2 Performance of DOS.H

Users who are currently using Datapoint computers in cassette-based systems will find substantial improvements in performance when they upgrade to DOS.H. The diskette drives are several times faster than the cassettes for ordinary sequential data transfers; random-access type operations (such as sorting and indexed sequential access method--ISAM-- file access) can easily be two orders of magnitude faster than is attained using tape cassettes.

Even further upgrade is obtained by adding the 9320 cartridge disk system. The ten-megabyte capacity of this disk offers a capability approaching that of the large capacity, high performance, disk operating systems while still being contained in a small-system package.

Users who are currently working with competitive disk-based equipment will generally find that total system performance of Datapoint systems will exceed that to which they are accustomed. This improvement is due to the generally superior data handling techniques and file structuring that are incorporated in Datapoint`s DOS.H. These characteristics stem from the fact that instead of employing a lower performance cassette-style file structure as a base for the operating system, Datapoint chose instead to adapt the same advanced and dynamic disk file access techniques that are used in its other disk operating systems to the new small-disk media. The result is a degree of software sophistication previously unavailable in business-oriented systems of this size.

1.3 Functional Differences Between Versions of DOS.H

The following is a list of the functional differences between DOS.H 2.4.2 and DOS.H 2.5.1.

1. Support for up to four 9320 disk controllers with four logical drives on each controller has been added to the 64K system. This system will support 16 logical drives (numbered 0-15), of which up to four may be diskettes.

2. Support for the 9320 disks has been added to the standard DOS utilities.

3. A new utility has been added to support the 9320 cartridge disks. REMAP will allow the logical remapping of drives under a 64K system with 9320 disks on-line. This permits the user to have any mix of 9320 disks and diskettes that is desired, (up to four diskettes), and to change this mix at any time.

4. The concurrent region in the 64K system has been decreased by 2K, to 10K, to allow for the 9320 disk handlers. On a 64K system with no 9320 disks spinning, the disk handlers are not present, so the concurrent region is a full 12K.

1.3.1 Compatibility of Utility Programs

The set of facilities described in this manual come as a complete package and are compatible within the 1500 DOS.H system. Additionally, programs written in DATABUS will generally run unmodified on DOS.H.

1.4 Hardware Support Required

The standard configuration required for DOS.H is a Datapoint 1500 with 36K of memory (32K of RAM, and 4K of ROM), and two or four diskette drives. The 1500 is also available with 64K of memory (60K RAM, 4K ROM). This system supports both diskettes and the 9320 cartridge disks.

CHAPTER 2. FUNDAMENTALS OF DOS.H

Basic information on the hardware, the disks themselves, and the organization of data on the disk files is requisite to setting up the user`s system. A general overview of physical elements and disk organization is presented here.

A 1500 DOS.H system may be configured with diskettes only, or with a 64K processor, diskettes, and 9320 disks. The appearance of the system to the user is different between these two configurations. If 9320 disks are being used, reference to this chapter, the chapter on System Generation, and the appendix on Disk Comparison should be made to understand these differences. Especially important is an understanding of logical drive and controller numbering with 9320 disks.

2.1 Diskette Systems

A diskette system supports up to four diskette drives through two integral controllers units. Each diskette controller contains 256 bytes of high speed, random access memory which buffers one sector between the diskette drives and Datapoint computer.

2.1.1 Diskette Media

The diskette drives use a flexible disk for data storage. This diskette is media- and format-compatible with the IBM 3740-style flexible diskette.

Data is recorded in 77 concentric circles on only one side of the diskette (according to the IBM standards for diskette data interchange). Each such circle is referred to as a track. Although each such track on the diskette actually contains 26 physical records of 128 bytes each, these are paired by the controller so that to the Datapoint computer, each track appears instead to consist of 13 records, called sectors, of 256 bytes each. In Datapoint documentation, the term sector always refers to a 256-byte logical sector unless stated otherwise.

The diskette is permanently enclosed within a durable, opaque plastic cover. This cover aids insertion of the diskette into the drives and provides structural rigidity when the diskette is not in use. The cover also furnishes a degree of environmental protection, guarding the oxide surface from accidents caused by careless handling.

2.1.2 Loading and Unloading Diskettes

The diskette loading slot is protected by a cover/handle that slides up and down to open and to secure the loading aperture. Pushing on the rectangular pushbutton below the cover disengages the latch and permits insertion and removal of the diskette.

The opaque plastic cover on the diskette has three holes on each side through which the oxide-coated surface can be seen. Only the holes on the labelled side are relevant to this discussion.

The large, round hole in the center permits installation on the drive. A hub on the drive clamps to the diskette, causing it to spin within its plastic cover.

The longer, narrower radial slot near the edge of the plastic cover permits the read/write head in the drive to contact the diskette`s oxide coating for data transfer operations.

The small, round hole near the center permits exposure of the disk`s index hole. This is a small opening on the recording surface which is sensed by the controller for purposes of timing and control.

These holes are described in detail in order to provide reference for inserting the diskette into the drive, as follows:

1. First, hold the diskette with the label side facing up.

2. Next, insert the edge of the diskette with the long slot into the drive.

3. When the long slot is inserted first, the small hole near the center is then the last of the three holes to enter the drive.

Always ensure that the labelled surface of the diskette is facing upwards.

On insertion, a spring resistance is felt when the diskette is about three-quarters of the way into the drive. Press the diskette gently into place until the spring catches with a faint click. The diskette will then stay in place of its own accord. When the diskette is in place, pull the door/cover down until it latches closed. The drive`s hub engages as soon as the door is latched, bringing the diskette online almost immediately. Diskette input/output activity is indicated by the LED below the diskette door on the front of the diskette unit.

To remove a diskette, first ensure that all input/output activity has stopped; if the door/cover is opened in the middle of a write operation, improper data deposition will result. When I/O activity is complete, press the pushbutton beneath the door. The diskette will pop out, much like toast from a toaster. As soon as the diskette is removed from the drive, place it in its protective paper envelope to keep the exposed surface safe from abrasive contaminants and mishandling.

2.1.3 Drive Numbering and Switches

--32K OR 64K SYSTEM WITH ONE OR TWO DISKETTE CONTROLLERS

> Diskette drives are normally installed in the cabinet starting from the right. The first controller electrically in-line to the 1500 processor is controller 1. The drives are, from right to left, drive 0 and drive 1. The second diskette controller, if it exists, is controller 2. The drives here are, from right to left, drive 2 and drive 3.

These drive numbers are sometimes referred to as the logical drive numbers, and frequently as simply the drive number. This equating of logical and physical diskette drive numbers is only valid on a system with no 9320 disks spinning.

Power for the disk drives is provided through the cable connecting them to the processor. There is no power switch on the diskette drive cabinet.

The power switch for the system is on the 1500 processor. There are no other user controls on the diskette system.

2.1.4 Care and Handling of Diskettes

Diskettes will give long and trouble-free service if they are handled with reasonable care. Important considerations for handling this media are emphasized below.

1. Diskettes should always be placed in their protective paper envelopes when not in use. The envelopes should

then be stored in the carton in which they were shipped.

2. Do not force too many diskettes into one box. Avoid placing diskettes under pressure; this can warp the diskette, causing read/write errors and/or failure of the recording surface to spin.

3. Diskettes should not be rolled, folded, or otherwise subjected to strain that could cause creasing.

4. Never touch the oxide coating of the diskette surface; skin oils deposited through the holes in the plastic cover attract and retain dust and other contaminants. Never permit the oxide coating to make contact with another hard surface; the smallest abrasion can scrape away the information-carrying oxide, resulting in unrecoverable errors on the disk.

5. Never subject diskettes to strong magnetic fields.

6. When mailing diskettes, ensure they are placed between two sheets of corrugated cardboard to maintain rigidity and provide protection through the mails. Diskettes may be packed in a suitable protective carrier available in singles or multiples from many disk manufacturers. These carriers are specifically designed for sending diskettes through the mail.

7. Diskettes can generally be taken through airport security X-ray and metal-detecting equipment without damaging the information recorded on the surface.

2.1.5 Preparing Diskettes for Use

When a diskette is first received from the manufacturer, it contains formatting information recorded across all boundaries on the surface. This information is provided to allow the controller to identify where a given sector is on the surface of the diskette, and also allows the controller to verify proper head positioning in the drive mechanism. Normal reading and writing on the diskette does not destroy the formatting information.

Only diskettes in IBM 3740 format (128-byte sectors) are usable by DOS.H.,Diskettes that have been reformatted with bad tracks flagged and alternate tracks substituted cannot be used. Diskettes in System 32 format (256-byte sectors) or IBM 3600 format (512-byte sectors) also cannot be used.

Diskettes cannot be used by DOS.H until they have been initialized with the DOSGEN program described in the command section of this manual. Datapoint DOS.H uses its own unique file structure, capable of sophisticated data and file manipulation. This structure requires DOSGEN generation of basic system tables on the diskette before it can be used with DOS.H.

A special note regarding diskettes and cartridge disks which are to be used in the booted drive is appropriate here. ("Boot" is the term used to describe bringing the system into a ready state. It is discussed later in the chapter on initial generation.) All of the DOS.H commands use DOS FUNCTIONS which are described in the DOS.D User`s Guide. These functions are resident on the disk in the file SYSTEM7/SYS. When updated versions of DOS.H and associated utilities are received from Datapoint Corporation, the file SYSTEM7/SYS may also have one or more new DOS FUNCTIONS resident. In this instance, wholesale copying of DOS.H commands from newly received disks to older disks (overwriting the older versions of SYSTEM7/SYS) should be avoided. The indiscriminate copying often results in command programs that do not work when the older version of SYSTEM7/SYS is present on the disk in the booted drive. The user should therefore keep his DOS.H commands disk (UTILITY/SYS, etc.) intact, using the newly-released disk to generate as many new system disks as he needs, and not supplying new commands to previously DOSGENed disks.

2.1.6 Suggested Diskette Organization Techniques

Due to the relatively small capacity of the flexible diskette, compared with the larger disk systems, careful consideration should be given to placement of files. For program development, the following convention is suggested.

1. DOS.H SYSTEM DISKETTES: These contain the system files and whichever DOS.H commands are regularly used. This is the diskette used in drive zero during program generation, debugging, and other DOS system-type functions. It will contain all of the DOS.H itself and DOS.H commands released by Datapoint. This diskette will also frequently contain the editor scratch file, SCRATCH/TXT.

2. SOURCE PROGRAM DISKETTES: These diskettes can be considered as library file diskettes. They contain DATABUS programs,

DATAFORM forms, and other user text files that are used during program generation. Once these programs are finalized, they can be copied to DOS.H System disks or User System disks, as appropriate.

3. USER SYSTEM DISKETTES: These diskettes are similar to DOS System diskettes, but differ in that they are intended for more specific applications. They will generally be used with DATABUS 1500, or DATAFORM and/or have large numbers of user-written applications on them. These diskettes generally will not contain the more specialized DOS.H commands such as DSKCHK15, APP, CHANGE, DUMP, and the like.

4. DATA FILES DISKETTES: These diskettes contain user data files. Typical characteristics of files on this type of diskette: non-executable; user information such as program output data, documentation, and data which is user-entered or user-generated, but not program code.

5. SCRATCH DISKETTES: These diskettes contain no important files. Their suggested use is for transferring files from one diskette to another, and for providing large unallocated areas to be used as scratch files for programs such as the SORT and EDIT commands.

As support for the above five basic types of diskettes, the followng color-coding convention is suggested for disk labels:

Red	-- DOS System diskettes
Green	-- User System diskettes
Blue	-- Text files (source programs and text data)
Yellow	-- Data files
Grey	-- Scratch disks

For best results, users should use only diskettes provided by those manufacturers recommended by Datapoint Corporation, as these diskettes are completely tested and certified. The use of substandard media can cause intermittent problems.

2.2 Cartridge Disk Systems

The 9320 cartridge disk system is a ten-megabyte disk-and-drive unit contained in a desk-top cabinet. Its integral controller contains a 256-byte (one sector) memory which functions as a speed buffer between the processor and the disk. Two switches are present on the front panel of the disk unit: RUN/LOAD and WRITE PROTECT ON/OFF.

When the RUN/LOAD switch is in the LOAD position, the controller stops the disk and unloads the heads to allow loading or unloading of cartridges. When this switch is in the RUN position, the disk is spun up and the heads are loaded.

When the WRITE PROTECT switch is ON, the disk is put in a read-only mode. The 1500 processor will hang beeping if a WRITE operation is attempted while the disk unit is write-protected.

A three position power switch is located on the back panel of the disk unit and functions as follows:

OFF:	No power is supplied to the disk unit.
LOCAL:	Power is supplied to the disk unit.
REMOTE:	Power supply is controlled by power indication signal on the I/O bus connected to the 1500 processor.

A disk is considered to be spinning if the back panel power switch is REMOTE or LOCAL and the RUN/LOAD switch is RUN. Otherwise, the disk is considered not to be spinning.

2.2.1 Cartridge Disk Media

A rigid, plastic cartridge, approximately 11"x11"x1", is the permanent housing for a 10.5" disk with two usable sides. Data is recorded on both surfaces of the disk, single density, 392 tracks per side. Each track is composed of 48 sectors, and each sector contains the Datapoint standard 256 bytes.

On the right front of the 9320 desk-top unit is a bottom-hinged, pull-down door. A large (1"x5") depression on the top of the door serves as a finger-pull to draw it open. As a security feature, the door cannot be opened when the drive is spinning. Nor can the door be opened when power to the 9320 System is disconnected or otherwise off. (These precautions help to insure the integrity of I/O operation--although it is possible that input data will be disturbed if the drive is switched to LOAD while a write procedure is taking place.)

To insert a cartridge, after the drive is in the LOAD mode and the door is open, hold the cartridge so that its label is facing up (and the side with the centered metal circle is facing down), and the hook-shaped corner is at the right rear. No other orientation will be admitted to the drive cage.

Once the cartridge is inserted, drive heads access the disk through the normally closed rear access window. This window opens when an unseen plunger, located in the extreme left rear of the drive cage, penetrates the narrow slot on the left rear corner of the cartridge case. To bring the disk online, close the door and switch the front panel switch to RUN.

To unload the cartridge, first ensure that I/O activity is complete. Then switch the front panel switch to LOAD. The door will open only when power to the unit is on. Lift the cartridge out carefully and place it in its storage rack.

2.2.2 Drive Numbering with 9320 Cartridge Disks

--64K SYSTEM WITH ONE OR TWO DISKETTE CONTROLLERS AND ONE TO FOUR 9320 CONTROLLERS

> The following discussion of drive numbering assumes that the 9320 disks are spinning. If the 9320 disks are not spinning, the system appears to be the same as a 64K system with only diskettes. DOS informs the user that it has recognized 9320 disks on-line (spinning) by the message, "DOS LOGICAL DRIVES 0 THRU 11 ARE ON 9320 DISKS, 12 THRU 15 ARE ON DISKETTES" after the DOS sign-on. This reflects the difference in logical drive mapping with 9320 disks.

The diskette controller numbers are 1 and 2, as with no 9320 disks, but the logical drive numbers are mapped differently by DOS.H. Logical drives on controller 1 are, from right to left, drive 12 and drive 13. Logical drives on controller 2, if present, are, from right to left, drive 14 and drive 15.

The first 9320 disk controller electrically in-line to the processor is controller 8, with logical drives 0-3. The second controller is controller 9, with logical drives 4-7. The third is controller 10, with logical drives 8-11. The fourth controller, which is not recognized by DOS.H at load time since the diskettes are mapped, is controller 11, with logical drives 12-15.

This logical mapping of diskette and 9320 drives can be changed (e.g., to delete diskette controllers 1 and 2, and add 9320 controller 11) via the DOS utility REMAP.

2.2.3 Care and Handling of Cartridge Disks

Size and shape of the cartridge make it convenient to handle and store. Since the disk surface is entirely enclosed, there is maximum protection from abrasion and contaminants, and the rigid case can be positioned book-style on a shelf when not in use. Alternatively, the hook-shaped corner can suspend the cartridge on a horizontal rod for storage.

It is strongly recommended that a cartridge be kept in the drive at all times, even when the drive is off, to protect the heads from airborne contaminants.

2.2.4 Preparing Cartridge Disks for Use

When the cartridge disk is received from the manufacturer, formatting information is already recorded across all boundaries on the surface. This information is provided to allow the controller to identify where a given sector is located on the surface of the disk, and also allows the controller to verify proper head positioning in the drive mechanism. Normal reading and writing on the disk does not destroy the formatting information.

Only disks in the 9320 format are usable in the system. Disks cannot be put into use until they have been initialized using the DOSGEN command program described in a later section of this manual. The 9320 cartridge disk requires the generation of basic system tables before it can be used with DOS.H.

CHAPTER 3. EQUIPMENT CARE

Computers, disk drives, printers, and other data processing equipment are delicate devices that must be operated correctly, and then handled with some degree of care to perform properly.

3.1 Environment

Datapoint systems must be installed in an area with adequate air conditioning. The Datapoint 1500 and its associated peripherals require a temperature range of 50 to 105 degrees Fahrenheit (10 to 40.5 degrees Centigrade). Humidity must be kept low enough to avoid moisture condensation (below 80%), but high enough to avoid excessive static electricity problems (above 10%).

The machine area must be reasonably clean and dust-free. While excessive cleanliness is not necessary, accumulations of dust and cigarette ashes can seriously affect machine operation. Spilled liquids are also hazardous.

Processors and peripherals require "clean" power to avoid erratic operation. Machine room power should be supplied from a completely separate transformer when possible. Ensure that devices such as adding machines and power tools are not connected to the same power leads that are used for computer equipment. Electric motors in these devices cause severe power line noise that may interfere with machine operation. Isolation transformers to insure a supply of clean power for Datapoint equipment should be used when necessary.

3.2 Processor

The only user-maintenance on the processor is an occasional dusting and cleaning of the cabinet, CRT screen, and keyboard.

Be sure the ventilation slots on the rear of the processor are never blocked; impeded air flow will cause overheating.

3.3 Disks and Disk Drives

Ensure that all operators know how to insert and remove diskettes and disk cartridges in the drives. Diskettes and disk cartridges must be stored properly in an environment similar to that provided for the equipment. Refer to the chapter, "Fundamentals of DOS.H," for details on disk handling. Other publications carrying this information are the Guide to Operating Datapoint Equipment and the Datapoint Product Specifications (green sheets).

The drives must not be subjected to bumps or jolts; this can cause head misalignment. Physical location of the drives must allow adequate air circulation for cooling purposes.

3.4 System Printer

The printer should be dusted occasionally in keeping with the necessary environmental cleanliness practices. Printer ribbons must be changed periodically to maintain print quality. Cloth ribbons left in use for too long can disintegrate, making print quality marginal at best. To avoid paper jams, ensure that the paper is aligned properly when loaded, and that there are no obstructions in the paper path.

CHAPTER 4. SYSTEM GENERATION

Before a disk can be used with DOS it must first be prepared by writing onto it basic system tables. Also, a surface verification must be performed so any bad areas of the disk surface will not be used. On a new installation, the system utility programs must be placed onto the disk for use. All these operations constitute system generation. With DOS.H there are two methods of system generation. The method used depends on the condition of the disk being prepared. If the disk is a new one, the method to be used is termed initial generation. This procedure writes the basic system tables and the directory onto the disk for the first time. If the disk has previously been in use, its tables and directory may require updating in order to accomodate a new revision of the Disk Operating System. The method is then called an upgrade.

4.1 Initial Generation

Datapoint distributes the Disk Operating System (DOS.H) and utilities on a single diskette. This diskette contains all programs necessary for generation of further system disks. In the chapter titled "Fundamentals of DOS.H," under the section, 'Preparing the Disks for Use,' there is general information about system generation. Specific procedures are found in the chapter describing the DOSGEN command.

4.1.1 Initial Loading

Initial loading of DOS.H, as described here, is known as \`hard booting.\`

To load DOS.H into the 1500 processor, the diskette should be inserted into the right hand drive of diskette controller 1. The RESTART and INT keys are then depressed simultaneously to initiate the loading of the DOS. To load the DOS from any of the other diskette drives, the DSP key is depressed to increment the diskette drive number that is being searched for a diskette. Note that the first on-line diskette found (starting with the right-hand drive of controller 1) will be the diskette that is loaded. Whichever of these drives loads the system is then recognized by DOS.H as the booted drive.

Loading DOS.H with the 9320 disk spinning follows the same procedure outlined above; however, the DOS handles the booted drive designation differently in a normal system load. As above, the system is loaded from the diskette, but the booted drive will be the first on-line 9320 drive found. (Remember, with 9320 disks spinning, the diskettes are mapped as logical drives 12-15.) To inhibit the 9320 drive from becoming the booted drive, the F5 and F4 keys are held down during system loading until a click is heard. In this way, the diskette drive that the DOS was loaded from will be recognized as the booted drive. This is useful for DOSGENing 9320 disks, or using UPGRADE/H.

The booted drive designation is valid until the next hard boot, or until the DOS utility BOOT is invoked to change the booted drive. The utility REMAP can also be used to change the booted drive.

In most cases, the booted drive must be kept physically on-line at all times, since DOS.H searches it first when encountering a non-specific drive command. DOS.H also sometimes places work files on the booted drive.

4.2 UPGRADE/H

An upgrade facility is available to accomodate upgrading existing diskette systems and diskette/cartridge disk systems. The program is a standard text file called UPGRADE/H (where H is the letter specification of the DOS in use). It is used as a chain procedure by the command:

```
CHAIN UPGRADE/H;IN=<input drive>,OUT=<output drive>
```

This command upgrades the tables and directory in drive OUT from the diskette in drive IN.

Disk files are then copied with the COPY utility. Since SYSTEM7/SYS contains the subdirectory information, the new disk will have the same subdirectory information as the old one. After the system files are copied, the new utility files are copied to replace any old utilities on the output diskette. The program PUTIPL (described in a later chapter) is then run to place the necessary initial program loader (IPL) blocks on drive OUT.

Since UPGRADE is a text file, it can be edited to modify the chain procedure that is used. This allows adjustment to special needs. Care should be exercised, however; any modification performed should be very carefully considered to assure a good upgrade. System conversions are a complex process and errors made at this juncture can result in an unusable diskette or lost data.

4.3 Scratch Disk Generation

Any diskette or disk cartridge that is to be used in a DOS system must first be prepared by writing onto it the necessary system tables and basic system files. Scratch disks or new system disks must be produced using the initial generation method and the DOSGEN command that is discussed above.

CHAPTER 5. DISK FILES

On all DOS-supported diskettes and cartridge disks, information is stored in sectors, each of which contains 256 bytes of information. Sectors containing related information are organized in a single structured group called a file. All information on a diskette and cartridge disk will generally be organized in files, except for certain system tables.

5.1 File Names

When calling up or creating a file from the console keyboard, the file is usually identified by a three-part designation consisting of a name, an extension, and a logical drive number. The first part of the name is the unique identifier. It consists of up to eight alphanumeric characters--none of the special characters are, however, allowed in a file name. Typical file names might be:

EDIT	PAYROLL
EMPLOYEE	5500
23NOV79	X1

The second part of the name, the extension, identifies the type of file. This may be from one to three alphanumeric characters. If an extension is used in a file name, it is separated from the name by a slash (/). The extension further identifies the file and usually indicates the type of information contained in the file. A "TXT" extension means text, and usually implies data. "CMD" implies an object code file to be used as a command program from the system console. Other common extensions are ISI, DBC, OVn, SYS, PRT, and REL.

The logical drive number specifies the logical drive on which the file is (or will be) located. Drive specification is identified by a leading colon (:) and has the form ":DRn" and ":Dn" where "n" is the logical drive number that was assigned at installation. It may also be specified as ":<volid>." The ":<volid>" form allows logical volume identification, regardless of the physical drive on which the disk is located. Valid drive numbers for diskettes are 0-3 on a system with no 9320 disk spinning. Note that on a 64K system with 9320 disks, the DOS utility REMAP may be used to change the logical drive numbers. Valid drive numbers of a 9320 cartridge disk system are 0-15; they may also be changed via REMAP. <Volid> is an eight-character identifier placed on the disk by the PUTVOLID program. This command is discussed in a later chapter.

The complete form of the name is thus:

```
NAME/EXTENSION:DRIVE
```

All three parts of the file name usually are not needed when a file name is entered as part of a command, though they can be specified. The presence or absence of any part of a file name is determined by the special separators "/" and ":". Syntactically correct file names are:

```
NAME/ABS:DR0        /ABS:DR1
NAME/REL            /TXT
NAME:D0             :D0
NAME                NAME:DOSH1
```

If a portion of the file name is not used, DOS and various of the utility programs apply default values. The default value depends on the location of the name on the command line, and on the command in use.

The first field on any command line is the command program to be run. For this field, a name must be given. If no extension is given, the default is then CMD, and the default drive is any drive. (An "any drive" default usually means a search of all drives, starting with drive zero.) If the command name is preceded by an asterisk (*), or a colon (:), the default extension and all-drive search do not apply; the leading character indicates that the given name is to be located as a member of the UTILITY/SYS (an "absolute library") before it is searched for as a file. Thus, if the program is not found in UTILITY/SYS, the Command Interpreter attempts to load a command file by that name.

The default values for file names given as parameters to a command are described separately for each command in the chapters following.

5.2 File Creation

Files are always created implicitly. That is, the operator never specifically instructs the system to create a given file. Any command that writes to an output file will write into an existing file or will automatically create a new file if necessary.

A file to be created will be placed on the drive specified in its file name field or specified in default values applied to its name. When a file is being created on a specific drive, files with the same name and extension on other drives are unaffected. If no drive is specified in the name or by default, the file is created on any drive which has free space. The search for available space starts on drive zero. "Available space" means one free space in the drive's directory, in which to place the name of the new file, and at least one cluster of free space on the disk, in which to place the data the file will contain. A "cluster" is the smallest unit of disk space that can be assigned to a file; clusters are defined in the chapter on System Structure.

5.3 File Deletion

Deletion of a file is always performed explicitly by operator command. The deletion command, KILL, is described in a later chapter. No other standard DOS command programs delete an existing file, although overwriting procedures such as system generation and BACKUP naturally destroy all files on the output disk.

5.4 File Protection

DOS files can be given three types of protection: write protection, delete protection, and no protection. If a file is write-protected, it cannot be written into, deleted, or renamed. If a file is delete-protected, it cannot be deleted, although it may be overwritten, thereby effectively destroying any data perviously written into it. If a file has no protection, it may, of course, be modified in any manner. The CHANGE command, described later, is used to set the protection of a file.

CHAPTER 6. OPERATOR COMMANDS

All Datapoint computers include, as a standard feature, an integral CRT display through which the internal computer communicates with the operator. The system console also includes a typewriter-style keyboard which the operator uses to communicate with the computer. The Disk Operating System is normally controlled by commands entered at this system console.

When DOS first becomes ready for commands, it displays a signon message on the CRT and says "READY". Upon completion of any job the DOS generally again displays "READY". Whenever the ready message is shown, the operator may key in a command, and this will be displayed on the bottom line of the CRT as it is keyed in. While typing a command, the BACKSPACE key will erase one character to make corrections, and the CANCEL key will erase the entire line.

6.1 Command Line

The command line is a directive that generally consists of from one to three parts, specifying first the job that is to be performed; second, the name of the disk file to be operated one, or special system directives; and third, any options for the job. The command programs provided with DOS are described in the following chapters. All information that must be entered for each command is specified in detail. A command line is always terminated by the ENTER key.

In general, a command line is entered as:

```
<field> <field>,<field>,<field>;[options]
```

The first <field> on the line always specifies the program to be run. Fields following the program name indicate a DOS file name and possibly a special file such as a subdirectory name. Separators between the fields, usually the space and the comma, must be given special attention. For readability, the first two fields are most often separated by a space; subsequent fields are separated by a comma. A command usually appears as:

```
SORT ACCTFILE,SRTFILE,:DR1;2-11
```

In this example, the first field, SORT, is the program to be executed.

It is separated from the second field by a space. Subsequent fields, "ACCTFILE", "SRTFILE", and "DR1", are separated by commas. The fields following the program name, SORT, provide additional information to the SORT program. [Options] following the <fields> are always separated from the preceding field by a semicolon. In the above exmple, "2-11" is the option. The colon (:) within the fourth field ":DR1" is a special separator used within a file name. The slash (/) is also used in this way, as shown in the preceding chapter, "Disk Files."

Aside from the separators noted above, most special characters (#, $, %, &, @, and other characters which are neither alpha nor numeric) act as separators in the same way as the space and the comma. In general, any character that is not a syntactically valid part of a file name is interpreted as a field separator. The command example above could have been entered as:

```
        SORT@ACCTFILE=SRTFILE$:DR1;2-11
```

The use of special characters is not recommended, however, since the resulting command line is confusing and difficult to interpret.

As already noted, the first field on the command line specifies the program to be executed. For any command, this first field must be given. Any other fields may or may not be needed for a particular command. The command program must be a loadable object file, loading above the resident operating system, or the program load will fail and the DOS will return to "READY" condition. If program specified to be run cannot be found, the the DOS displays the message "WHAT?" and waits for another command. When the program name specification is preceded by an asterisk (*), the UTILITY/SYS file will be searched for the command. If it is not found there, the Command Interpreter then looks for a separate command file.

6.1.1 Order-Dependency in the Command Line

Fields on the command line are often order-dependent. If a command is being used which accepts several fields and one of the fields is not wanted, skip that field by entering two separators with nothing between them. Such a command would look like:

```
        SORT ACCTFILE,,:DR1;2-11
```

By using two commas, ":DR1" is recognized as the fourth field on the line, with the third field being null.

6.2 General Command Characteristics

Some features of the commands suplied with the Disk Operating System apply to most of the DOS commands. These characteristics and the messages that can appear following them are discussed briefly.

6.2.1 General Command Format

When the command line is discussed, the first field is called the "command;" subsequent fields before the semicolon are called <filespec>; characters following the semicolon are called "options" or "parameters." The general format of the command line is:

```
Command [<filespec>][,<filespec>][,<filespec>]...[;options]
```

The item referred to as "Command" is required on a command line. This first field defines the command being issued to the system.

The items referred to as "<filespec>" represent one or more specifications for files. These files generally are input, output, scratch, or other files to be used by the command program. Usually, the first such specification represents input files(s); the following specifications represent output or scratch file(s).

A square bracket is used here to indicate fields whose presence is optional. The pointed bracket convention (as in <filespec>) represents replacement fields where the replacement field name is contained within the pointed brckets. The brackets are used here as text cues only and are not keyed in on the command line.

6.3 Signon Messages

When a system command is entered, the command program being invoked will always display a message identifying itself. The message includes the version number of the command program. The primary purpose of this signon message is to allow the operator to determine, in the event of some difficulty, which version of the command is in use.

6.4 Common Error Messages

There are basically three types of error messages that may be displayed. The error messages common to many of the DOS utilities are listed here, the DOS error messages are listed in Chapter 43, and hardware error messages are listed in Chapter 44.

INVALID DRIVE

This message appears when one of the drive specifications given by the operator is invalid. Either the drive specification was not of the correct format, or the drive number specified exceeds the range available under the resident Disk Operating System. The other possibility is that a <volid> was specified, and the proper disk was not on any online drive.

NAME IN USE

This message generally occurs when the command`s continued execution would necessarily result in a conflict of the specified file name with an already existing file.

NAME REQUIRED

This message indicates that a file specified on the command line could not be found. Generally the name as specified is misspelled or otherwise incorrectly entered. However, the message will sometimes occur because the file desired is not in the current subdirectly. (A subdirectory is described later in the SUR command.)

NO! THAT FILE IS PROTECTED

This message indicates that a request was made to modify a file having write or delete protection.

WHAT?

This message means that the command name (the first item on the command line) is illegal. Most often, the command specified is not valid, or is not in the current subdirectory or the SYSTEM directory.

WRONG DOS!!

This message indicates that the command being attempted is not compatible with the DOS being used.

CHAPTER 7. ABTONOFF COMMAND

7.1 Purpose

Within the DOS, there is a flag which is set by several of the utilities under certain conditions. This is the ABTIF bit, which may be tested with the //ABTIF operator in a CHAIN (see the CHAIN command). If the flag is on, the chain will abort when tested by //ABTIF; otherwise it will continue. The ABTONOFF command is a means of insuring the setting of this flag. The format of the command line is as follows:

```
ABTONOFF [<parameter>]
```

The only valid values of the parameter are ON and OFF, and these determine the settings of the flag. If ABTONOFF is executed with no parameters, the current setting of the flag is displayed, but it is not changed.

CHAPTER 8. APP COMMAND

8.1 Purpose

The APP command appends two object files together creating a third. Object files are files containing absolute object code in a format that can be loaded by the DOS loader.

8.2 Use

```
APP <file spec>,[<file spec>],<file spec>
```

The APP command appends the second object file after the first and puts the result into the third file. Note that neither of the input files are disturbed. If extensions are not supplied, ABS is assumed. The first two files (if a second is specified) must exist. If the third file does not already exist, it will be created. The first file's transfer address, if present, is discarded and the new file is terminated by the transfer address of the second file, if one is present. The transfer address of an object file is defined as the entry point of the program contained in the file.

Omitting the second file specification causes the first file to be copied into the third file. For example:

```
APP DOG,,CAT
```

will copy the file DOG/ABS into the file CAT/ABS.

The first and third file specifications are required. If either is omitted, the message

```
NAME REQUIRED
```

will be displayed. The second and third file specifications must not be the same. If they are, the message

```
FILES TWO AND THREE MUST BE DIFFERENT
```

will be displayed. If a non-object record is encountered, the message

```
FILE X CONTAINS A NON-OBJECT RECORD
```

will be displayed with the value of X being 1 or 2 to correspond with the position of the file specification on the command line.

Because the APP command recognizes the actual end of the object module contained in a file, APPing an object file, similiar to the example above, is one technique for releasing excessive unused space at the end of an object file. However, it should be noted that the APP command cannot be used to append overlay library files. If this is attempted, the message

```
YOU CANNOT APPEND OVERLAY LIBRARIES
```

will be displayed. If the third file specified is write protected, the message

```
FILE THREE WRITE PROTECTED.
```

will be displayed

Another use of the APP command is to append patches to object files, since the object file being APPed may load at the same address as object code in the original program. However, since the new code is at the end of the module, it is loaded over the old code.

CHAPTER 9. AUTO COMMAND

AUTO - Set Auto Execution

```
AUTO <file spec>
```

The AUTO command establishes the indicated program to be automatically executed upon the loading of DOS. If no extension is supplied, ABS is assumed. The message

```
AUTO NOW SET TO NAME/EXTENSION (PFN).
```

will be displayed when the named file is set for auto execution. A check is made to see if the file is an object file and if the file is on the booted drive. If the specified file does not exist, the message

```
NO SUCH NAME
```

will be displayed. Note that if a program has been set to auto-execute, its execution can be inhibited by depression of the keyboard (KBD) key when DOS is loaded. Any attempt to auto a file not on the booted drive will cause the message

```
YOU CAN`T AUTO THAT FILE!
```

to be displayed. If there was already a file set for auto-execution, the message:

```
AUTO WAS SET TO NAME/EXTENSION (PFN).
```

will be displayed before the new auto-execution setting has been made.

If no <file spec> is entered on the command line, no change in the current auto-execute program will be made. If no program was set for auto-execution the message:

```
NAME REQUIRED
```

will be displayed.

Note that the AUTO command does not make provision for file specifications to be given to the program which is to be automatically executed. This makes it impossible to use AUTO for programs requiring or accepting such parameters. AUTO also does not place anything in the DOS communications region (MCR$). Therefore, programs which use overlays with the same name (but different extension) as the program will not run. For more information, refer to the chapter describing the AUTOKEY command.

Auto-execution mode is cleared with the MANUAL command, described in a later chapter.

Programs contained in absolute libraries (UTILITY/SYS for example) cannot generally be "AUTO'd" directly. Use the AUTOKEY command described below, then "AUTO AUTOKEY/CMD". If an absolute library is specified for auto-execution, the first member of that library will be the program executed. System files cannot be "AUTO'd" under any circumstances.

CHAPTER 10. AUTOKEY COMMAND

10.1 Introduction to AUTOKEY

Many users allow their Datapoint computers to run in an unattended mode. This allows large data processing tasks, perhaps running via the DOS command chaining facility (see CHAIN), to be run during the evening hours when no operator is present. (An example might be the creation of several new index files for one or more large, ISAM-accessed data bases). However, the momentary power failures which data processing users are being forced to contend with during times of shortage, thunderstorms and the like can bring down any computer not having special, uninterruptible power supplies. When this happens to a computer running in unattended mode, the office staff will generally return the next morning to find their computer sitting idle and its work unfinished.

The Datapoint computers are all equipped with an automatic-restart facility which can be used to cause them to automatically resume their processing tasks following such an interruption. The purpose of the AUTOKEY (and AUTO) commands is to provide a software mechanism for users who wish to handle such unusual circumstances and provide for the restarting of a processing task.

10.2 The Auto-Restart Facility

The 1500 is provided with firmware auto-restart. Thus if the machine ever halts, for example due to a power failure, on being started again, it will attempt to load the operating system.

10.3 Automatic Program Execution Using AUTO

In order to provide a mechanism for programs to resume automatically following an interruption (such as a DATABUS system, for instance, which might be running unattended) DOS has a comparable facility to enable a program to be automatically executed whenever DOS comes up. (Note that any loading and running of the DOS, whether by an auto-restart, manually restarting the machine, or under program control, will activate this facility).

The AUTO command is used to establish a program to receive control when DOS comes up. This setting can be cleared with the MANUAL command. For some applications, the AUTO and MANUAL commands are adequate to allow a programmed restart of a lengthy data processing task. However, some programs require parameters be specified on the command line, and these are obviously not present if no command line has been provided.

10.4 Auto-Restart Facilities Using AUTOKEY

AUTOKEY is simply a command program which can be AUTO'd. The way it works is very simple. If it is run via the DOS auto-restart facility, AUTOKEY supplies a command line just as if the same one line were entered at the system console. If AUTOKEY is run from the system console (or likewise from an active CHAIN), it simply displays the command line it is currently configured to supply and offers the user the option of changing that stored command line.

The command line supplied to AUTOKEY could do anything specifiable in one command line to the DOS; DATABUS could be brought up, a SORT invoked, a user's own special restart program started or even a CHAIN begun. AUTOKEY, when used with AUTO, MANUAL, and CHAIN can therefore provide a very powerful facility.

10.5 A Simple Example

To specify a command line to be used during automatic system restart, simply enter:

```
AUTOKEY
```

at the system console. AUTOKEY will display a signon message and display the current autokey line if there is one. It then asks if this line is to be changed. If "N" is answered, AUTOKEY simply returns to the DOS and the DOS "READY" message is displayed. If "Y" is answered, AUTOKEY requests the new command line to be configured and then returns to the DOS and "READY".

Alternatively, if the user wishes to simply specify a new command line to be configured regardless of the current setting of the AUTOKEY command line, he can merely place the new command line after the "AUTOKEY" that invokes the AUTOKEY command.

As a simple example, suppose that the XYZ company has several 1500s running DATABUS, which run completely unattended at night to produce reports. An electrical surge from a lightning bolt causes the power to be cut off for 15 seconds. As soon as power is restored, their Datapoint 1500 computers re-boot their DOS and warm-start the DATABUS system. (OVERNITE/DBC is the Databus program that controls the report preparation.) One command sequence to accomplish this would look like the following:

```
AUTOKEY
DOS.H AUTOMATIC KEYIN COMMAND
NO AUTOKEY LINE CONFIGURED.
CHANGE THE AUTOKEY LINE? Y
ENTER NEW AUTOKEY LINE:
DB15 OVERNITE
READY
AUTO AUTOKEY/CMD
AUTO NOW SET TO AUTOKEY/CMD (nnn)
READY
```

An alternate form of the above would be the following:

```
AUTOKEY DB15 OVERNITE
DOS.H AUTOMATIC KEYIN COMMAND
NO AUTOKEY LINE CONFIGURED.
ENTER NEW AUTOKEY LINE:
DB15 OVERNITE (supplied automatically)
READY
AUTO AUTOKEY/CMD
AUTO NOW SET TO AUTOKEY/CMD (nnn)
READY
```

Once a program has been set for auto-execution, the only way one can bypass it is to hold down the keyboard (KBD) key while the DOS is coming up. This action bypasses the auto-executed program and enters the normal command interpreter. The user then can use the MANUAL command to clear the auto-execution option.

10.6 Special Considerations

When building long chain files that allow for automatic restart, several considerations must be made. Among these are that a file must not be changed in such a way that the change cannot be repeated if the previous checkpoint is actually used. To accomplish this, frequently the file being updated must be copied out to a scratch file, and the scratch file then updated. Following the completion of the update is when another checkpoint would be taken. The next phase would copy the updated file back over the original. Note that a checkpoint (i.e. resetting the AUTOKEY command line) would have to be before the creation of the dummy copy to be updated; putting a checkpoint between the creation of the copy to update and the actual updating process could result in the updating of a partially updated copy. A little thought when choosing places to update the AUTOKEY command line is called for to ensure that the chain may be resumed from any of them without incorrect results.

10.7 AUTOKEY and DATABUS

Some users who make frequent use of the DATABUS ROLLOUT feature will notice that AUTO-ing AUTOKEY with the AUTOKEY command line set to DB15;R will mean that whenever the system rolls out to any program or chain of programs, DATABUS is automatically brought back up when that program or chain of programs finishes, regardless of whether or not DB15;R was included at the end of the port's chain file.

CHAPTER 11. BACKUP COMMAND

11.1 Purpose

The BACKUP command provides for making disk-to-disk copies of DOS. The user can make either an exact image copy of the input disk or can select reorganization, which will group files by extension and file name, remove unnecessary segmentation and allow deletion of unnecessary files. Reorganization also allows copying of DOS disks onto disks with locked out cylinders that differ from those on the input disk. The user also may optionally create a data disk containing null DOS system files with the use of the "N" option.

Disks may have a symbolic volume identification (see the PUTVOLID command). If the user chooses to backup without file reorganization, the VOLID on the output disk will be replaced. If the the file reorganization option is chosen, the output disk VOLID will be unaffected.

Note that a BACKUP may only take place between like media; a disk to diskette backup or vice-versa will not be allowed, due to the differences in size betweeen the two media.

11.2 Use

A disk backup is initiated by the operator entering the following command:

```
BACKUP <input drive>,<output drive>[;N]
```

Input drive and output drive are specified as :DRn, or :Dn, or :<volid>. The optional "N" parameter will allow the user to create a data disk containing null DOS system files on the output drive. The purpose of doing so is to make more space available on the output disk. The remaining user files will be copied as if the "N" option had not been specified. The program will respond by displaying the message:

```
DRIVE n SCRATCH?
```

If the disk on drive n is scratch, enter a "Y". Any other reply will cause the program to return to DOS. If you do reply "Y", the program will display the message:

ARE YOU SURE?

If you are absolutely sure that you want to write over the output disk, type "Y" again and press the ENTER key. Any other reply will cause the program to return to DOS. If the output (logical) disk has not been DOSGENed or the DOS file structure on it has been damaged, the message:

DOSGEN YOUR OUTPUT DISK FIRST!

will appear and control returns to DOS. If the output disk has been DOSGENed and seems in reasonable condition, the following message is displayed:

FILE REORGANIZATION?

If different cylinders are locked out on the input and output disks (if the disks' lockout CATs do not match), an exact image BACKUP is not possible so the "FILE REORGANIZATION?" question is bypassed. Instead, a message appears specifying that reorganization is required and BACKUP with reorganization proceeds as described below.

If you wish to reorganize the files being transferred to the output disk, enter a "Y" in response to the reorganization question. In this case, see the section on reorganizing files for further instructions.

If you do not wish to reorganize your files and desire an exact image copy of your input disk, enter an "N" in response to the reorganization question.

11.3 Exact Image Copy

If you have typed "N" in response to the file reorganization question, the program will ask the question:

DO YOU WANT THE OUTPUT COPY VERIFIED?

This question should always be answered "Yes". It takes only a little longer to verify the output copy, but verification can catch a significant number of errors.

The program then asks:

DO YOU WANT TO COPY UNALLOCATED CLUSTERS?

Type "Y" and press the ENTER key if you want all data on the disk copied regardless of whether or not it is in an area allocated by DOS. This option is preferred in cases where you suspect that your DOS files may be partially destroyed or the output disk has never been fully initialized with data.

Type "N" and press the ENTER key if you wish to copy your disk as quickly as possible without copying unused areas of the input disk. "Y" and "N" are the only replies allowed.

11.4 Reorganizing Files

If you have typed "Y" in response to the file reorganization question, the program will copy the System files (unless the "N" option was specified), sort the Directory names, and allow the operator to delete files before copying the files to the disk copy.

Backup with reoganization to the booted drive is not possible.

11.4.1 Copying DOS to Output Disk

Various program status messages will appear during the copying of DOS. System tables are initialized and then the SYSTEMn/SYS files are copied to the output disk unless the "N" option was given on the command line. If the "N" option was given, the message:

```
SYSTEM FILES WON'T BE COPIED
```

will be output and null DOS system files will be created on the output drive.

11.4.2 Deleting Named Files

When all directory names have been sorted into file extension followed by file name sequence the following question will be displayed:

```
EXCLUDE ANY FILES DURING REORGANIZATION?
```

Type "N" and press the ENTER key if all files are to be copied. Type "Y" and press the ENTER key if you wish to delete any files. If you reply "Y" a message asking which files are NOT to be copied will appear. The screen will be filled by a numbered list of files for you to choose from. Type the number or range of numbers (nn or nn-nn) found next to names of individual files you wish deleted. Type "ALL" and press the ENTER key if you wish to delete all of the files in the list. The files selected for deletion will be erased from the list. When all desired deletions have been made from a list, type "." and press the ENTER key to advance to the next list of file names.

When all file name lists have been examined, the program will advance to the "Copying Named Files" phase.

11.4.3 Copying Named Files

Before copying, the files named in the system directory are sorted into ascending order, first by extension, then by name. The name of each file is displayed as it is copied. All files are written as close together as possible with a minimum of segmentation.

11.5 Use of KEYBOARD and DISPLAY Keys

The keyboard (KBD) and display (DSP) keys may be pressed any time messages are being displayed. Depressing the display (DSP) key will hold the current display until the key is released. Depressing the keyboard (KBD) key will cause the program to terminate and return to DOS.

11.6 Error Messages

During the execution of BACKUP the following error messages may appear:

```
INVALID DRIVE SPECIFICATION!
```

Action: Retype the BACKUP command with correct <input-drive> and <output-drive> specification.

```
INPUT AND OUTPUT MUST NOT BE ON THE SAME PHYSICAL DRIVE.
```

Action: <input-drive> and <output-drive> have been specified as the same drive. Retype BACKUP command with correct specification.

```
BAD CLUSTER ALLOCATION TABLE!
```

Action: A bad Cluster Allocation Table has been detected on the input disk. The Cluster Allocation Table may be able to be fixed using the DSKCHK15 command.

INPUT DISK LOCKOUT CAT UNREADABLE

Action: A read error has been detected on the input lockout CAT. Reorganization will be done automatically.

OUTPUT DISK LOCKOUT CAT UNREADABLE

Action: A read error has been detected on the output lockout CAT. Reorganization will be done automatically.

THE DISK'S LOCKED-OUT CYLINDERS DO NOT MATCH

Action: The input and output disk CAT's do not match. Reorganization will be done automatically.

BOOTED DRIVE IS ILLEGAL OUTPUT DRIVE DURING REORGANIZATION!

Action: You cannot backup to the booted drive if reorganization has been specified. Retype BACKUP command with correct drive specifications.

BACKUP PROCESSING ABORTED!

Action: None! The keyboard (KBD) key was hit causing BACKUP to abort it's current processing.

UNABLE TO OPEN INPUT FILE!

Action: BACKUP was unable to open the input file currently being processed. The file will be bypassed and processing will continue.

DUPLICATE NAME. NOT COPIED.

Action: BACKUP has encountered a file name that has already been copied to the output disk. The file will be bypassed and processing will continue.

SYSTEM TABLE SECTORS OF BACKUP DISK ARE UNUSABLE!

Action: Your scratch disk cannot be used for a system disk due to surface defects in the sectors used for system tables. Use another output disk and start over.

SYSTEMn/SYS IS MISSING!

Action: Your input DOS disk cannot be reorganized due to a missing system file. Catalog the missing system file on your input disk and start over.

PARITY- :DRn address

Action: An irrecoverable parity error has been detected on drive n during the BACKUP operation. The address is shown for each error. If drive n is your output disk, DOSGEN must be rerun to lockout the bad addresses or use a different scratch disk for exact image copy. If drive n is your input disk, new parity will be computed and the record will be copied. Note the error address and check for errors when copy is complete.

CANNOT BACKUP BETWEEN DIFFERENT MEDIA

Action: Only disk-to-disk or diskette-to-diskette BACKUPs are allowed.

BACKUP CANNOT EXECUTE WITH CONCURRENCY ACTIVE

Action: Allow the concurrent job to terminate and re-enter the BACKUP command.

11.7 Reorganizing Files for Faster Processing

After a DOS disk has been used for awhile, the file structure becomes fragmented and related files become scattered. The more the disk is used the more total system performance is degraded due to increased disk access time. System degradation is especially noticeable when DATABUS is being used. File reorganization using the BACKUP program is one way to clean up DOS disks and improve their efficiency.

BACKUP reorganization improves system efficiency by making the following changes:

. File segments are consolidated

. Files are packed more closely together

. Related files are clustered together

. Unused trash files are removed (optionally)

. Files are rewritten reducing marginal parity errors

11.8 BACKUP with CHAIN

Because BACKUP does not abort if parity errors occur, and due to the complexity of the questions (which may depend upon the condition of the input and output disks) BACKUP should not generally be invoked from CHAIN. Since the BACKUP operation is so critical to the protection of important files, an operator should monitor the entire backup operation.

11.9 Clicks during Copying

A click occurs each time an unused sector is copied (reorganization mode only). A file which, when copied, results in many clicks (more than three, perhaps) can probably be reduced in size, without any data loss, by using APP or SAPP as appropriate.

CHAPTER 12. BLOKEDIT COMMAND

12.1 Purpose

The BLOKEDIT command is a program for DOS text file manipulation. It copies lines of text from any DOS text file(s) to create a new text file.

The BLOKEDIT command is useful for such things as:

New program source file generation by copying routines from existing program source files.

Existing program source file re-arranging by copying the lines of source-code into a new sequence (into a new source file).

Re-arranging lines or paragraphs of a SCRIBE file into a new file.

In this Chapter, the following applies:

Text file means a DOS text file as defined in the REFORMAT chapter.

Line means one line of a text file as displayed by the DOS LIST program.

12.2 File Descriptions

BLOKEDIT deals only with text files. For any given application, there will be one text file called the COMMAND FILE which will hold the controlling commands for BLOKEDIT. Optionally the controlling commands may be entered directly to BLOKEDIT via the keyboard by defaulting the command file parameter. There will be one or more text files called SOURCE FILES from which lines of text will be copied. And there will be one text file called the NEW FILE which will be the desired end result for the application.

12.2.1 Command Statement Lines

The command statements are the controlling factor for a BLOKEDIT execution. The command statements specify which source files will be used and which lines of text will be copied from them. If the command statements are to be read from a command file it must be generated by the DOS EDIT command, or DOS BUILD command, etc., before BLOKEDIT can be used.

There are three kinds of statement lines that are meaningful to BLOKEDIT: COMMENT lines, COMMAND lines, and QUOTED lines.

A COMMENT line is a line which has a first character of period.

This is an example of COMMENT LINES:

```
.
. THESE THREE LINES ARE COMMENT LINES.
.
```

As in program source files, a comment line may have explanatory notes or nothing at all following the period.

A COMMAND LINE is a line which has a SOURCE FILE NAME and/or source file LINE NUMBERS, or begins with a double quote symbol (").

The following are some example command lines:

```
FILENAME/EXT:DRO           NAME THE SOURCE FILE
1-100                      COPY LINES 1 THRU 100
350-377                    COPY LINES 350 THRU 377
150/TXT                    NAME THE SOURCE FILE
```

A command line must have a first character of an upper-case alphabetic character, or a digit, or a double quote symbol.

A command line that begins with an upper-case alphabetic character indicates that a new SOURCE FILE is being named. A new source file can be named only by putting the name of the file at the very beginning of the command line. Optionally, the extension and/or drive number for the file may be included with the source file name. If the source file name begins with a digit the file extension must be given.

A command line that begins with a digit indicates that the command line will have one or more numbers, which are the numbers of the lines to be copied from the source file previously specified into the new file.

A command line that begins with a double quote symbol indicates the beginning/ending of QUOTED LINES. The only information used by BLOKEDIT in a command line that begins with a (") is the (") itself, therefore the rest of the line can be used for comments.

A QUOTED LINE is a line between a pair of command lines which begin with a double quote symbol.

This is an example of QUOTED LINES:

```
" THIS IS THE BEGINNING OF QUOTED LINES COMMAND LINE.
CHAIN REORG
DB15;R
" THIS IS THE END OF QUOTED LINES COMMAND LINE
```

There may be more than one quoted line between the command lines that begin with ("). A quoted line will be copied directly from the command file or keyboard to the new file. Quoted lines enable a BLOKEDIT user to include original lines of text in a new file along with lines copied from source files.

12.2.2 Source File

The SOURCE FILE is a text file from which lines will be copied. Source files are named in the command lines for a BLOKEDIT application, and the lines to be copied from the source file will also be specified in the command lines. It will be useful to have a listing of a source file with line numbers, as produced by the LIST command, when creating the command statement lines for a BLOKEDIT application.

12.2.3 New File

The NEW FILE is a text file produced by the BLOKEDIT command. The new file is named at BLOKEDIT execution time by the second file specification entered on the command line.

12.3 Using BLOKEDIT

Before the BLOKEDIT command can be used, one must create a command file, unless the command statements are to be entered via the keyboard. When the BLOKEDIT command is to be executed, the operator must enter the following command line:

```
BLOKEDIT [<file spec>],<file spec>[;O]
```

The first file specification refers to the command file. If not specified the commands will be entered via the keyboard. The second file specification names the new (output) file. If no extension is supplied with the first file specification, TXT is assumed. If no extension is supplied with the second file specification, the extension given or assumed for the first file is used. If no drive is given for the first file, all drives are searched. If no drive is given for the second file, the drive given or assumed for the first file is used. The output file must not already exist unless the "O" option is used. Only if the Overwrite (O) option is specified will BLOKEDIT overwrite an existing file. If commands are being entered through the keyboard and the output file already exists, the user will be asked if it is to be overwritten.

12.4 Messages

This section describes the operator messages that BLOKEDIT may display on the CRT screen during execution. Some of the messages are monitor messages to keep the operator informed of the progress of the program, while other messages are error messages. If the keyboard was selected as input to BLOKEDIT, the user will be prompted by the "Please enter a BLOKEDIT command Enter * to exit. " message when input is required. The character * will terminate BLOKEDIT and return to DOS.

The general format of the CRT display screen varies depending on the source of the BLOKEDIT command statements.

If the command statements are being read from a command file the format of the display is:

```
 _______________________________________________________________
/  DOS.VER. TEXT FILE BLOKEDIT   DATE     OUTPUT FILE IS XXXXX/XXX \
|  PROCESSING COMMAND LINE nnn  CURRENT SOURCE IS XXXXXXXX/XXX:DRn |
|                                                                  |
|                                                                  |
|                                                                  |
\                                                                  \
/                                                                  /
|                                                                  |
|                                                                  |
|                                                                  |
|   Error Message Displayed Here If Necessary                      |
|                                                                  |
\__________________________________________________________________/
```

If the command statements are being entered via the CRT keyboard, the format is:

```
 _______________________________________________________________
/  DOS.VER. TEXT FILE BLOKEDIT   DATE     OUTPUT FILE IS XXXXX/XXX \
|  PROCESSING COMMAND LINE nnn  CURRENT SOURCE FILE IS -NONE-/ :DR |
|                                                                  |
|                                                                  |
|                                                                  |
|                                                                  |
\                                                                  \
/                                                                  /
|                                                                  |
|                                                                  |
|                                                                  |
|  PLEASE ENTER A BLOKEDIT COMMAND  ENTER * TO EXIT                |
|                                                                  |
|                                                                  |
\__________________________________________________________________/
```

As BLOKEDIT commands are entered on line 12, they are rolled through lines 3 through 17.

12.4.1 Informative Messages

PROCESSING COMMAND LINE .. CURRENT SOURCE FILE IS ../..:DR.

This message is the BLOKEDIT monitor message. This message is displayed while BLOKEDIT is writing lines of text to the new file. The monitor message displays the command file line number currently being processed and the name, extension, and drive number of the last named source file.

SOURCE FILE WENT TO E.O.F.

This messge is displayed if the source file from which lines were being copied ended before the specified lines were finished.

BLOKEDIT TRANSFER COMPLETE
OUTPUT FILE WAS name LINE COUNT WAS nnn

This message is displayed when all of the command file lines have been executed. If the command statements are being entered via the CRT keyboard, this condition will be determined when an asterisk (*) is entered. If the command statements are being read from a command file, either an asterisk (*) or an end of file (EOF) will determine the condition. The number of lines in the new file is displayed following the second line.

12.4.2 Fatal Errors

If BLOKEDIT detects a fatal error in the command statement line the monitor message is rolled up the screen, an appropriate error message is displayed, and the program returns an error condition to the DOS.

NEW FILE NAME IS REQUIRED

This message is displayed if the operator did not name a new file when the BLOKEDIT command was called.

COMMAND FILE DRIVE IS INVALID

This message is displayed if the operator specified a drive number for the command file that is invalid.

NEW FILE DRIVE IS INVALID

This message is displayed if the operator specified for the new file a drive number that is invalid.

<u>COMMAND AND NEW FILE NAMES CAN NOT BE IDENTICAL</u>

This message is displayed if the operator specified command file and new file names the same and the extension and the drives for the files were specified or assumed to be the same. Defaulting of extensions and drives is described in an earlier paragraph.

<u>NEW FILE AND SOURCE FILES CAN NOT BE IDENTICAL</u>

This message is displayed if the operator specified source file and new file names are the same and the extension and the drive for the files were specified or assumed to be the same. Defaulting of extensions and drives is described in an earlier paragraph.

<u>COMMAND FILE NOT FOUND</u>

This message is displayed if the command file name was not found on the drive(s) specified or assumed.

<u>NEW FILE NAME IN USE</u>

This message is displayed if the specified output file <u>was</u> found on the drive(s) specified or assumed. BLOKEDIT will not write into an existing file if commands are being read from a command file unless the "O" option was specified. If commands are being entered to BLOKEDIT via the keyboard, the operator is given the option to overwrite the existing file.

<u>NEW FILE NAME IN USE, OVERWRITE IT?</u>

If the operator answers Yes (Y) the file is overwritten.
If the reply is No (N) BLOKEDIT returns control to DOS.

<u>BAD FILE SPECIFICATION</u>

This message is displayed if the first character of a command file line, other than a quoted line, is an upper-case alpha character but the DOS file specification was not recognizeable.

12.4.3 Selective Fatal Errors

These errors are fatal when BLOKEDIT is reading a command file, and informative when commands are being entered via the keyboard.

<u>SOURCE FILE NOT FOUND</u>

This message is displayed if the source file specified could not be found. It is probably either misspelled or in a different subdirectory.

<u>BAD LINE NUMBER SPECIFICATION</u>

This message is displayed if a command file line other than a quoted line began with a digit but contained an unrecognizable line number specification.

Here are some examples of valid line numbers:

4	A single digit is acceptable.
999999	A line number may have up to six digits.
100-364	First and last line to be selected are separated by a dash.
34,55-78,100-147	Commas separate line specifications.

Here are some examples of invalid line numbers:

1A	Only "-", ",", or space after a digit, unless the line is a source file name beginning with a digit. If it is, an extension must be given.
1234567	Number has more than six digits.
17-34-77	Only two numbers separated by "-".

<u>LINE NUMBER ZERO IS NOT VALID</u>

This message is displayed if a line number of zero is specifed in a command line. It is ignored if entered via the keyboard.

<u>START LINE NO</u>. > <u>END LINE NO</u>.

This message is displayed if the first number of a line number pair is larger than the second number of the pair, as in: 235-176. It is ignored if entered via the keyboard.

<u>BAD DATA IN SOURCE FILE LINE nnn</u>

This message is displayed if BLOKEDIT discovers non-ASCII characters in a source file. The line number will be displayed following the message. If commands are being entered via the keyboard, the source file is reselected and the next command is requested.

<u>NO VALID SOURCE FILE FOR TRANSFER</u>

This message is displayed if BLOKEDIT discovers line numbers to be transfered from a source file when there is no open source file.

<u>FORMAT OR RANGE ERROR ON SOURCE FILE</u>

This message is displayed if DOS discovers a file which can not be read. If commands are being entered via the keyboard the source file will be de-selected, and the next command requested.

CHAPTER 13. BOOT COMMAND

The BOOT command allows the user to change the disk drive from which operating system overlays and utilities are loaded without physically restarting the system. This causes DOS to be reloaded from the selected drive after execution.

The BOOT command line is:

```
    BOOT <:Dnn> <;<DOS command line>>
```

For example:

```
    BOOT :DR1
```

Would cause drive 1 to be treated as the system drive for loading programs and utilities until the system is physically rebooted or BOOT is run to change it. This is an example of soft booting.

To check which drive is the currently booted drive, just enter BOOT on the command line, and the current booted drive number will be displayed. Then, BOOT will ask for a new boot drive number. Entering an asterisk here will cause the program to return to DOS.

CHAPTER 14. BUILD COMMAND

14.1 Purpose

BUILD provides an alternative means to create a text file without having to use the standard DOS editor. BUILD is useful for rapid generation of very short text files, such as two and three line CHAIN files. Also, BUILD is usable from within a CHAIN.

14.2 Use

The BUILD command is invoked by entering the command line:

```
BUILD <file spec>[;<end character>]
```

The _file specification_ defines the output file. This output file specification is always required. If the named file does not exist, it is created. The default extension is TXT.

The _end character_ is optional. If no end character is specified on the command line, BUILD terminates upon receiving a null input line (a null input line is a line consisting of only an ENTER; a blank line is not a null line).

BUILD accepts input lines from the keyboard and writes each one to the output file. When BUILD is ready to accept an input line it displays a colon (:) as a prompting character. Each input line BUILD receives is tested for the presence of the specified end character, if any, as the first character entered. If the end character is present as the only character of the entered line, the end line is discarded (it is not written to the output file), and an end of file mark is written to the output file and the output file closed by returning to DOS.

Entering an end character followed by a string will pass the string to the output line without the end character and will not terminate BUILD. This action allows entering CHAIN commands into a chain file being written by BUILD from within an active CHAIN.

14.3 A Simple Example

Suppose that the operator wishes to construct a simple CHAIN file to establish a program to be auto-executed, so that the auto-execute request can be accomplished later with a single command line entered at the keyboard. All that is required is to enter at the system console:

```
BUILD <chain file spec>;*
AUTOKEY <program name>
AUTO AUTOKEY/CMD
*
```

Upon receiving the "*" input line, BUILD closes the output file and terminates. Note that in the two places where the "*" appears, any enterable character could have been used. (This allows nesting calls to BUILD, which can be very useful in the BUILDing of chain files). After the BUILD command is finished, the output file named on the BUILD command line contains the following two lines:

```
AUTOKEY <program name>
AUTO AUTOKEY/CMD
```

It is also possible, through BUILD nesting, to create chain files which during execution of the chain construct other chain files and execute them automatically upon completion of the first chain (since any statement of a chain file _is_ allowed to be a CHAIN command).

The references to CHAIN made here may be premature, since CHAIN is discussed in a later chapter, but are included because BUILD and CHAIN can be a powerful facility when used together in this manner.

The keyboard (KBD) and display (DSP) keys may be pressed any time messages are being displayed. The keys will be effective just prior to the display of the prompting ":". Depressing the display (DSP) key will hold the current display until the key is released. Depressing the keyboard (KBD) key will cause the program to terminate and return to DOS.

14.4 Error Messages

The following error messages can occur during the execution of BUILD.

INVALID DRIVE SPECIFICATION

This message is displayed if the drive specification entered for the output file is not within the range of valid drives.

INVALID FILE SPECIFICATION

This message is displayed if the file specification entered for the output file is blank.

CHAPTER 15. CAT COMMAND

15.1 Purpose

The CAT command selectively displays filenames in the DOS directory or members in a library directory. One may choose to display all cataloged filenames on all drives online or specific filenames on specific drives.

15.2 Use

The CAT command is invoked by entering the command line:

```
CAT [<name>][</ext>][:DR<n>][,L]
```

where: <name> specifies the filename or a portion of the filename, <ext> specifies the extension or a portion of the extension, <n> specifies the logical disk drive number, and L specifies list only those files in the current subdirectory.

To display members of a library directory enter:

```
CAT <library name>*
```

To display members of the UTILITY/SYS directory enter:

```
CAT *
```

DOS Directory entries are displayed in the form:

```
NAME/EXTENSION (PFN)P
```

where PFN is the physical file number in octal (0-0377) and P is the protection on the file; D for deletion, W for write, and blank for none. If the file displayed is in a subdirectory other than system, the directory entry is displayed in the form

```
NAME/EXTENSION-(PFN)P
```

with the dash indicating a subdirectory entry. All drives are searched, unless a specific drive is requested, and as each drive is scanned, the line:

```
     ---- DRIVE n VOLUME ID (volid)   SUBDIRECTORY (subdirectory name)
```

is displayed. This line is not displayed if the drive is not on line, or if no files from it are to be displayed.

Depressing the display (DSP) key causes the catalog display to pause as long as the key is held. Depressing the keyboard (KBD) key causes the catalog display to terminate. If the CAT command is parameterized by only an extension, only files of that extension will be displayed. If the CAT command is parameterized by only a name, only files of that name (all extensions) will be displayed. If the CAT command is parameterized by a name and an extension, only files of that root name and extension (all drives) will be displayed. If the CAT command is parameterized by only the drive number, only files on that drive will be displayed. If only a portion of the filename is entered, all files beginning with the letters specified will be displayed. For example, entering:

```
     CAT /T
```

would cause the display of all files on all on-line drives whose extensions start with "T".

Entering:

```
     CAT MA:WORK2
```

would cause the display of all files on symbolic drive "WORK2" whose file names start with "MA".

CHAPTER 16. CHAIN COMMAND

16.1 Introduction

The CHAIN command enables the sequential execution of more than one command program without operator intervention. To accomplish this, the user creates a chain input file that contains commands to invoke the programs to be used, plus all the inputs that are required by these programs. This input file is then compiled by CHAIN into a procedure file, which then replaces the keyboard entry routines--key-ins which would normally be performed by the operator--with appropriate instruction lines that cause the routines to execute automatically. Each time a program would normally request a line or character to be entered from the keyboard, it is read instead from the procedure file.

The chain input file may be created by EDIT or BUILD, and is a listing of all the commands, comments, tags, and directives needed to call up and execute the several jobs or programs.

CHAIN command functions in two phases, compilation of the input file and execution of the procedure file. In the compilation phase, statements are read from the user-created input file and compiled into the procedure file named CHAIN/SYS. The CHAIN/SYS file then consists only of statements necessary for the execution phase. In the execution phase, CHAIN interfaces with the Operating System and handles control and retrieval of information from the procedure file. CHAIN/OV1, a DOS utility, fetches each line from the procedure file. If the line read is longer than the maximum specified by the calling program the program is aborted and the chain abandoned.

When a program invoked by the CHAIN procedure file terminates normally (rather than in error) , the CHAIN/OV1 routine reads the next statement, if present, from the procedure file. If the end of file is reached when DOS is requesting another command, the CHAIN is determined to be finished. At this time, normal termination of CHAIN, the procedure file is deleted and commands can be entered via the keyboard. If the end of file is reached at any other time, the CHAIN will abort abnormally, and the procedure file will not be deleted. In any case, CHAIN will return to DOS.

The CHAIN command line specifies the name of the text file to be compiled and the parameters, if any, of the file:

```
     CHAIN <input filespec><;parameter><,parameter>...
```

As shown, parameters follow the semicolon after the input file name. If there are many parameters, as there may be in a long, complex CHAIN, the minus sign (-) after the last parameter on the command line, in place of the normal comma, allows the parameters to be continued on the next line.

16.2 Tag (Parameter) Definition

The CHAIN command line can contain both tag names and/or tag names and values for the tags. These parameters follow the semicolon (;) on the command line which invokes CHAIN. The tag names can be from one to eight characters in length and may have values from one to seventy characters in length. A tag must contain only letters or digits. The value of a tag may contain any valid character except comma (,), equals (=) or pound sign (#). The character restriction depends on the syntax being used.

A tag is defined just by its presence on the CHAIN command line. Tags may have a value given to them by one of the following syntaxes:

```
CHAIN DOIT;LIST,DATE=30NOV76,TIME=1500hr     (New Syntax)

CHAIN DOIT;LIST,DATE#30NOV76#,TIME#1500hr#   (Old Syntax)
```

Both syntax structures are supported and the results of the two CHAIN commands is identical. The tag LIST has been defined but has a null value; DATE has the value of 30NOV76 and TIME has the value of 1500hr.

CHAIN allows two uses to be made of tags:

1.) A tag can be tested to determine whether it was defined on the CHAIN command line.

2.) The value of the tag can be substituted on CHAIN input statements before the line is written to the Procedure File.

16.3 Tag Substitution

A tag value is substituted whenever a pair of "#" symbols are found with a syntactically valid tag name between them. The value substituted is the tag value given on the CHAIN command line. For example, with the following contents of an EDITed file called "DOIT":

```
MSTRRCV RUN;XL
ACCOUNTS RECEIVABLE ON #DATE# - #TIME#
MSTRRCV #NAME#;XL
#NAME# LATE/30 DAYS ACCOUNTS ON #DATE# - #TIME#
```

and with the CHAIN program invoked by "CHAIN DOIT;TIME=2400hr,DATE=29NOV79,NAME=LIST1," the procedure file will contain:

```
MSTRRCV RUN;XL
ACCOUNTS RECEIVABLE ON 29NOV79 - 2400hr
MSTRRCV LIST1;XL
LIST1 LATE/30 DAYS ACCOUNTS ON 29NOV79 - 2400hr
```

If a tag is mentioned on the CHAIN command line but given no value, and if the value is to be used for substitution, a null value is substituted for the #tag# within the line. The effect is that the #tag# characters disappear from the line. For example, if CHAIN was invoked by "CHAIN DOIT;TIME,DATE=29NOV79, NAME=LIST1," the procedure file will contain:

```
MSTRRCV RUN;XL
ACCOUNTS RECEIVABLE ON 29NOV79 -
MSTRRCV LIST1;XL
LIST1 LATE/30 DAYS ACCOUNTS ON 29NOV79 -
```

If a tag is not mentioned on the CHAIN command line but is to be used for substitution in the procedure file, no value is substituted, and the value #tag# appears as is in the procedure file.

16.4 Compilation Phase Directives

All CHAIN directives are denoted by the characters // as the first two on a line. Any number of spaces (including zero) are scanned until the CHAIN directive is reached. The first thing after the // must be a valid CHAIN directive or else an error message is issued and CHAIN is aborted. The following is a list of these statements.

//IFS	IF SET (TAG DEFINED)
//IFC	IF CLEAR (TAG NOT DEFINED)
//XIF	END OF IF
//ELSE	REVERSE EFFECT OF IF
//BEGIN	BRACKETS A GROUP OF
//END	IF/ELSE/XIF STATEMENTS
//.	EXECUTION TIME COMMENT
//*	EXECUTION TIME BREAKPOINT
//ABORT	ABORT CHAIN COMPILATION
//ABTIF	CONDITIONALLY ABORT CHAIN EXECUTION
.	COMPILATION TIME COMMENT. (Note that the slashes are not present)

16.4.1 IF Directive

The IF directive has two variations, IFS and IFC, which are IF SET and IF CLEAR. The IFS directive proves positive if the tag named appeared on the CHAIN command line, and negative if the tag was omitted. If an IF directive tests positive, it has an effect; if an IF directive tests negative succeeding lines in the chain file are skipped (compilation is turned off) until a later directive turns the compilation back on.

For example:

```
        //IFS LIST
```

will prove positive if LIST was mentioned in the CHAIN command line, and negative if the tag does not exist. The opposite of this is true for the IFC directive.

For example:

```
        //IFC LIST
```

will prove positive if LIST was omitted and negative if it appeared on the CHAIN command line.

Simple logical operations can be performed on IF directives. The tags to be used are separated by logical operators. The logical OR is indicated by '¦' (vertical bar) or ',' (comma). The logical AND is indicated by '&' (ampersand) or '.' (period). For example the following lines are in the file DOIT:

```
//IFS DATE&TIME¦QUICK  or  //IFS DATE.TIME,QUICK
DBCMP15 TEST               DBCMP15 TEST
```

If DATE <u>AND</u> TIME <u>OR</u> QUICK are defined on the CHAIN command line the DBCMP15 test line will be included in the procedure file. CHAIN DOIT;DATE=30NOV76,TIME=1500hr or CHAIN DOIT;QUICK or CHAIN DOIT;DATE,TIME will all result in a true logical condition and the DBCMP15 line will be included.

IF directives are only evaluated if lines are being included. If one IF directive has proven negative and has inhibited the use of lines, all following IF directives will be ignored until either an ELSE or XIF statement is found.

For example:

```
        //IFS DATE
        //IFS TIME
        DBCMP15 TEST
        //XIF
```

If DATE was not defined, all lines until the //XIF will be ignored. In this example the //IFS TIME statement would not be evaluated and the DBCMP15 TEST would not be included even if TIME was defined.

16.4.2 ELSE/XIF Directives

CHAIN has two directives that will alter the inclusion of lines from an IF directive. The first is the XIF directive. It will unconditionally terminate the range of the last IF directive. The second is the ELSE directive; it will reverse the results of the last IF directive; that is to say, if lines were being skipped because the last IF proved negative, an ELSE would cause lines to be included.

For example, the DOIT file contains the following lines:

```
//IFS LIST
DBCMP15 TEST;L
//ELSE
DBCMP15 TEST
//XIF
//IFS COPY
```

```
COPY TEST/DBC,:D1
//XIF
```

If CHAIN is invoked by 'CHAIN DOIT;LIST' the procedure file will contain

```
          DBCMP15 TEST;L
```

If invoked by 'CHAIN DOIT;COPY', the procedure file will contain

```
          DBCMP15 TEST
          COPY TEST/DBC,:D1
```

16.4.3 BEGIN/END Directives

The BEGIN and END statements allow groups of IF/ELSE/XIF statements to be parenthesized. A counter called the BEGIN/END counter is initialized to zero when compilation of a procedure begins. If the use of procedural lines is turned off and a BEGIN operator is encountered, then the BEGIN/END counter is incremented. If an END operator is encountered, then the BEGIN/END counter is decremented unless it is already zero. The ELSE and XIF operators have no effect if the BEGIN/END counter is not equal to zero. For example:

```
        // IFS FLAG1
        DBCMP15 TEST1;XL
        TEST PROGRAM ONE
        // ELSE
        // BEGIN
        // IFS FLAG2
        DBCMP15 TEST2;XL
        TEST PROGRAM TWO
        // ELSE
        DBCMP15 TESTTEST;XL
        TEST TESTER
        // XIF
        // END
        // XIF
        // IFS FLAG3.FLAG27
        LIST SCRATCH;L
        THE SCRATCH FILE AT FLAG 27
        // XIF
```

The 6th through the 12th lines will not be used if FLAG1 exists, notwithstanding the fact that there is an ELSE and XIF operator within those lines, because the BEGIN/END pair prevented these statements from having any effect.

16.4.4 ABORT Directives

The //ABORT statement will cause CHAIN to return to DOS if it is processed. For example:

```
//IFC TIME¦DATE
.**** TIME AND DATE ARE BOTH REQUIRED
//ABORT
//XIF
.
.
.
```

If the Procedure File is invoked with TIME or DATE missing, the error message comment line would be displayed, and the compilation of the input file would ABORT.

The //ABTIF statement will conditionally cause the execution phase of CHAIN to ABORT. This statement causes the ABTIF bit, internal to DOS.H, to be tested. If the bit is on, the chaining will abort. If errors have been found during the execution of the last program the ABTIF bit should be set. For example, the procedure file contains:

```
KILL TESTFILE/DBC
Y
//ABTIF
KILL OUTPUT/TXT
Y
.
.
```

If the file TESTFILE/DBC is not found by KILL, the ABTIF bit would be set. When the //ABTIF statement is processed the abnormal program completion bit will be checked, and in this case it will be on, so the CHAIN will be aborted. There is a utility, ABTONOFF, whose function it is to turn the ABTIF bit on or off. Thus CHAIN files which make use of the ABTIF facility should use this utiltiy to make sure the bit is off before starting. See the chapter on the ABTONOFF command for further information.

16.5 Comments

CHAIN allows for two types of comment lines within the procedural file. One type is the execution time comment. This type may appear only before a DOS command entry and will not appear until just before that command is to be executed. An execution time comment can appear only just before a command because at any other place in a procedure file, the comment would be presented as keyboard response to an executing program. Comments can be placed at the end of a procedure, since this location is equivalent to immediately prior to a command. For example, the procedure file containing:

```
//. COMPILATION OF THE TEST PROGRAM
DBCMP15 TEST;XL
TEST PROGRAM
```

would cause the first line to be displayed before the compilation was executed. A variation on the execution time comment is the operator break point. An operator break point in a comment line causes the comment line to be displayed, the processor to BEEP, and execution is suspended until the operator depresses the KBD or DSP key. For example, the procedure file containing:

```
//* INSERT SOURCE DISKETTE INTO DRIVE 1
COPY TEST/TXT,:D1
COPY TEST1/TXT,:D1
COPY TEST/INC,:D1
```

The second type of comment line is a compilation time comment. This line is not included in the procedure but is displayed on the screen immediately after it is read from the procedural file. This is useful in communicating to the operator what procedure is about to be followed by CHAIN.

Both types of comment lines will be ignored (not displayed or written) just as other procedure lines if a test has proven negative and an ELSE or XIF operator has not been reached. For example, if the following procedure file MAKETEST was created:

```
.  COMPILATION OF TEST PROGRAM
// IFS LIST
.  YOU ARE GOING TO GET A LISTING
DBCMP15 TEST;XL
TEST PROGRAM
// ELSE
.  YOU AREN'T GOING TO GET A LISTING
DBCMP15 TEST
```

and the CHAIN command:

```
CHAIN MAKELIST;LIST
```

was given, then only the lines:

```
. COMPILATION OF TEST PROGRAM
. YOU ARE GOING TO GET A LISTING
```

will appear on the screen before the procedure is executed. If, however, the CHAIN command:

```
CHAIN MAKETEST
```

was given, then only the lines:

```
. COMPILATION OF TEST PROGRAM
. YOU AREN'T GOING TO GET A LISTING
```

will appear on the screen before the procedure is executed.

16.6 Resuming An Aborted CHAIN

Before the CHAIN overlay fetches the next DOS command, it stores CHAIN/SYS file pointers for the line to be used. If something goes wrong during the DOS command which follows and the procedure is aborted, CHAIN still knows where it was in the CHAIN/SYS file when the problem occurred. Since CHAIN does not delete the CHAIN/SYS file unless the procedure completes successfully, it can pick up where it stopped in the CHAIN/SYS file, if the operator can correct the condition which caused the procedure to abort in the first place. Often, the reason for the abort is something correctable like the disk running out of files. In this case, the operator need only correct the condition and then enter:

```
CHAIN *
```

and the procedure will pick up with the command which failed before. Thus, one can recover from jammed paper in a printer half way through a listing by simply fixing the printer, and then entering the CHAIN * command.

If the failing command cannot ever succeed, it may be bypassed by entering the command:

```
CHAIN/OV1
```

This simply restarts the chain with the next available line in the procedure. If the next line had been intended as a keyin (response) line for the failed program (as opposed to a DOS command line) the chain will generally immediately abort again. However, by restarting the chain in this manner, repeatedly if necessary, the invalid step can usually be bypassed and chaining resumed.

Note that CHAIN/OV1 is not guaranteed to always work. If the pointers to the CHAIN/SYS file are lost, only CHAIN * will work.

16.7 Notes On Usage of CHAIN

CHAIN then replaces the DOS keyboard entry routine. Therefore, only programs that use this routine for input will receive their input from the chain file. Programs which have their own input routines, like the Editor, can be invoked from a chain file but editing must be done manually by the operator. The CHAIN program itself can be called from within a CHAIN file. The chain is aborted when a CHAIN-invoked program makes an exit to DOS that implies that an error of some kind has been made. The error message given by the program will generally remain on the screen after the chain is aborted.

Some programs can go through a rather complex set of requests for input which can make them hard to use with the CHAIN program without making a mistake. For this reason, most DOS programs allow almost all options to be specified on the command line and keep the variation in the number of keyin requests to a minimum. It is good practice for all programs to be written with this in mind to facilitate their use with CHAIN.

An additional item to keep in mind is the fact that some DOS programs use their own keyboard entry routine as well as the one provided by the DOS. This enables the program to avoid the use of the CHAIN procedural lines when special operator intervention is required. Also, if a keyin line from the CHAIN procedure file is longer than that requested by the executing program, the CHAIN will be aborted.

16.8 Error Messages

The following error messages can occur during the compilation phase of CHAIN. All errors displayed will cause CHAIN to terminate its current execution and return to DOS.

BAD DEVICE SPECIFICATION.

This message is displayed if the drive specification for the CHAIN command file is not within the range of valid drives.

NO SUCH NAME.

This message is displayed if the CHAIN command file does not exist.

NAME REQUIRED.

This message is displayed if the CHAIN command file was not entered on the command line.

CHAIN OVERLAY MISSING.

This message is displayed if the CHAIN overlay (CHAIN/OV1) does not exist on the same drive that the CHAIN command is on.

CONDITION SPECIFICATION ERROR.

This message is displayed whenever a logical operator other than a '¦' (vertical bar), ',' (comma), '&' (ampersand), or '.' (period) is encountered in the command file.

LINE OVERFLOW DURING VALUE SUBSTITUTION.

This message is displayed whenever a value substitution causes the generated logical CHAIN/SYS record to be larger than 80 characters.

TAG VALUE NOT TERMINATED.

This message is displayed if the line continuation character (-) is encountered and it is not followed by a carriage return character.

OPTION SPECIFICATION ERROR

This message is displayed if a tag name on the command line is either blank or followed by a terminator other than a '#' (pound sign), '=' (equal sign), '-' (dash), or a ',' (comma).

SYMBOL TABLE OVERFLOW.

This message is displayed if the symbol table overflows during the compilation phase of CHAIN. The symbol table has 2K (2048) bytes in it.

TAG DEFINED MORE THAN ONCE.

This message is displayed if a defined tag appears more than once on the command line.

CHAINING ALREADY ACTIVE.

This message is displayed if a CHAIN * command is encountered within the CHAIN/SYS file during the execution phase of CHAIN. The message would actually appear during the compilation phase of the CHAIN command executed inside the execution phase of CHAIN.

UNDEFINED CHAIN OPERATOR.

This message is displayed if any one of the CHAIN directives is encountered without a carriage return following the directive encountered.

THE CHAIN/SYS FILE HAS BEEN DELETED.

This message is displayed whenever an attempt to restart a CHAIN file is made and the CHAIN/SYS file does not exist on the same drive as the CHAIN command.

CHAINING ABORTED.

This message is displayed if a //ABORT directive is encountered.

The following error message can occur during the execution phase of CHAIN.

CHAINING ABORTED. -ABTIF- ERROR BIT ON.

This message is displayed if a //ABTIF directive is encountered and the ABTIF error bit is on.

CHAPTER 17. CHANGE COMMAND

CHANGE - Change a file's protection

```
CHANGE <file spec>;p
```

The CHANGE command enables one to write-protect, delete protect, or clear the protection of a disk file. If a file is delete -or write-protected, a KILL command (or program generated KILL) cannot affect it. If a file is write protected, it cannot be written into by the standard system routines.

The option parameter "p" is used above to indicate the protection for the file specified. Protection can be specified as:

D	delete protect
W	write protect
X	clear protection.

For example:

```
CHANGE NAME/EXTENSION;D
CHANGE NAME/EXTENSION:DR1;X
```

will delete protect the file in the first case, and remove all protection in the second case. If a first specification is not given, or if the file indicated by the first file specification cannot be found, the message

```
FILE NAME MISSING.
```

will be displayed. If the option parameter does not follow the above syntax rules, the message

```
INVALID PROTECTION SPECIFICATION.
```

and a sample CHANGE line will be displayed. If the semicolon (;) is replaced with any other character, the message

```
PROTECTION UNCHANGED.
```

will be displayed and no protection change will occur. If the drive specification entered is invalid, the message

```
     INVALID DRIVE
```

will be displayed and no protection change will occur.

CHAPTER 18. CLOCK COMMAND

The Datapoint 1500 maintains an internal time-of-day and date clock. When the machine is first powered up, the clock is cleared to zero, and is marked as not having been set. The CLOCK command may be used to read the clock and display the time and date on the screen, as well as to set either the time, the date, or both.

The format of the CLOCK command is as follows:

```
CLOCK[;S]
```

The S option is the only one recognized, and indicates that the clock is to be set, regardless of whether it had been set previously. If no options are specified, the time and date will be displayed if the clock has been set. If it had not been set, the program will request the time and date. The program allows two formats for the date: the familiar "MM/DD/YY" format, and the full display format, for example: AUGUST 23, 1977. Three-letter abbreviations for the month are allowed, as well as specifying only the last two digits of the year. The time is entered in the form "HH:MM:SS", but the seconds field is optional, and will default to zero. When the time and date have been entered, the program asks, "ARE YOU SURE". The clock will not be set until a "Y" answer is given for this question, allowing a certain amount of control over the seconds portion of the time.

When the program asks for the date, which is requested first, a null response (simply pressing the ENTER key) indicates that the current date is to be used. Entering an asterisk ("*") terminates the program immediately. The same applies to the time, except that of course a null entry indicates that the current time is to be used. If the clock had not been set since power-up, null entries will not be accepted.

The CLOCK command gives various messages for error conditions such as invalid dates or times. These are all self-explanatory and all allow the information to be keyed in again.

CHAPTER 19. COPY COMMAND

19.1 Purpose

It is frequently useful to make a copy of a disk file. It may be desired, for example, to make a copy on a separate disk for backup or distribution purposes.

The COPY command allows a user to selectively update (replace) an existing file, or create (add) a new file to receive the copy. These options used in combination with the CHAIN utility provide an easy method of updating and maintaining DOS disks.

The COPY command does not make assumptions about the format of the sectors being copied, but merely copies the file sector-for-sector. It can copy all types of disk files. Some particular types of files are still unmovable, however. The outstanding example is INDEX files, usually with extension ISI. These cannot be moved because index files contain internal pointers indicating their actual physical location on the disk volume. These are made invalid when the file is moved to another place on the disk.

19.2 Use

Note that DOS.H provides two different COPY routines. The selection of which COPY routine is invoked by the COPY command line is made by DOS. A fast COPY routine is invoked if concurrency is inactive, else the normal COPY routine is invoked. The only outward difference between the two is that the fast COPY routine does not click for unallocated sectors or format errors.

The COPY command is invoked by entering at the system console:

```
COPY <file spec>,<file spec>      Unconditional copy
COPY <file spec>,<file spec>;R    Replace only
COPY <file spec>,<file spec>;A    Add only
COPY <file spec>,<file spec>;7    Copy SYSTEM7/SYS
```

UNCONDITIONAL:

This option will cause the first specified file to be copied into the second one. Attributes of the first file, such as its protection, are copied to the second file as well.

REPLACE:
This option will copy a file only to an existing file. If the output file does not exist no copy of data takes place and an informative message is given, "file-name NOT COPIED".

ADD:
This option will copy a file only if the output file does not already exist. If the output file does exist no copy of data takes place and an informative message is given "file-name NOT COPIED".

COPY SYSTEM7/SYS:
This is used only when upgrading disks. Since the subdirectory information is kept in SYSTEM7/SYS, it is sometimes necessary to copy the other information in that file, without overwriting the subdirectories. This option assumes that the output file already exists, and the copy starts with the second sector of each file, after the sector which contains the subdirectory data. Thus a user may upgrade disks to a new version of the operating system without destroying the subdirectories on the output disk.

The only portion of the file specifications that is specifically required is the name of the input file. The extension of the input file, if none is specified, is assumed to be TXT. If a drive specification is entered for the input file, then only that specific drive is searched for the indicated file. If no drive specification for the input file is given, all drives are searched. If the name of the output file is omitted, it is assumed to be the same as that of the input file. If the output file's extension is not given, it is also assumed to be the same as that of the input file. All drives are searched for the output file unless a particular drive is specified.

Example: to copy file PAYROLL/TXT from symbolic drive "WORK2" to symbolic drive "WORK1"

```
COPY PAYROLL:WORK2,:WORK1
```

Example: to make another copy of PROGRAM/ABS on drive zero, but to be named MYPROG.

```
COPY PROGRAM/ABS,MYPROG:DR0
```

Example: to make another copy of PAYROLL/TXT drive 0, on drive 1 only if it does not already exist on drive 1.

```
COPY PAYROLL:D0,:D1;A
```

Example: to update (replace only) TREK/ABS, a file on drive 0 from a newer version on drive 1.

```
COPY TREK/ABS:DR1,:DR0;R
```

A file may not be copied to itself. However it may be copied to a file of the same name on another drive.

The COPY command issues a click each time an unused sector is copied. If more than four or so clicks occur at the end of copying a file, it usually indicates that the file is larger than necessary to contain the data in it. In this case, moving the file using APP or SAPP can sometimes help to reduce its size. Clicks ocurring during the copying (before the end of the file) indicate sectors containing DOS format errors, possibly implying a sector accidentally destroyed by some faulty program.

19.3 Error Messages

The following error messages can occur during the execution of COPY.

```
CAN'T FIND THAT FILE.
```

This message is displayed if the input file specified does not exist.

```
I NEED THE NAME OF THE INPUT FILE.
```

This message is displayed if the input filename is not entered on the command line.

```
INVALID DRIVE
```

This message is displayed if the drive specification entered for input or output file in not in range of the valid drives.

```
INVALID VALUE IN PARAMETER LIST.
```

This message is displayed if a parameter other than 'R', 'A', or '7' is entered on the command line.

'7' OPTION IS ONLY VALID WHEN COPYING SYSTEM7/SYS.

This message is displayed if the '7' option is entered on the command line for any file name other than SYSTEM7/SYS.

CHAPTER 20. DOSGEN COMMAND

20.1 Purpose

Before any disk can be used by DOS, certain tables and other information must be placed onto it to establish the basis that DOS requires for the support of its file structure. These tables include the skeleton of the DOS directory, (where the names of the files contained on the disk are stored), as well as a map showing which places on the disk are bad and should not be used. The purpose of the DOSGEN command is to provide the user with a simple way of accomplishing this preparation.

Note that the use of DOSGEN requires at least a two-drive system, since DOSGEN cannot be made to write on the drive from which it is executed.

20.2 Use of DOSGEN

To DOSGEN a disk enter:

```
DOSGEN <drive spec>
```

The drive spec is a standard DOS drive specification which specifies which drive contains the disk to be prepared for DOS use. Since the directory initialization process will effectively KILL any files that might be on the disk, the command asks several times to make sure that the operator is aware of the potential seriousness of the operation he has invoked.

After the operator has acknowledged that he does not mind the overwriting of the new disk, the command asks if any tracks on the volume are to be locked out. Normally, the answer to this question is NO. However, by answering YES, it is possible to cause the DOS to lock out one or more tracks of the disk from DOS access. This can be useful in some special applications where it is desired to not allow DOS programs access to a file stored in unusual format. If the user does wish to lock out any tracks, he may do so by specifying one or more track numbers, in the format:

```
12,14,16,25-28,40
```

The above example would cause tracks 12, 14, 16, 25, 26, 27, 28, and 40 to be locked out. Note that the track numbers to be locked out are given in decimal as opposed to octal.

After the operator has specified which, if any, tracks are to be locked out, the DOSGEN command checks for bad sectors on the disk and issues a message indicating any tracks it finds which contain bad sectors. Any tracks found bad are automatically locked out and will not be used by DOS. The remainder of the operation is completely automatic and indicates its completion with the familiar DOS message, "READY".

Upon completion of the DOS generation process, the only files on the new disk are the eight system files SYSTEM0/SYS through SYSTEM7/SYS and UTILITY/LNK, if UTILITY/LNK was present on the same drive as DOSGEN.

20.3 Special Considerations

An important thing to remember is that disks must be formatted before DOSGEN can be used on them. Disks and diskettes for the 1500 drives come pre-formatted from the manufacturer. A disk that has been formatted with tracks locked out (error mapped) cannot be DOSGENed. Note that leaving logical drives unDOSGENed on 9320 disk packs will cause system data failures if those unDOSGENed drives are accessed.

20.4 Error Messages

The following error messages can occur during the execution of DOSGEN.

COPY/CMD, PUTIPL/CMD, CHAIN/CMD, AND CHAIN/OV1 MUST ALSO BE PRESENT.

This message is displayed if anyone of the files mentioned in the above error message does not exist on the "booted" drive.

ERROR IN DOS FUNCTION

This message is displayed if any of the calls to the DOS functions to get a specified directory sector, or a physical disk address of a CAT returns with an error condition.

DRIVE OFF LINE.

This message is displayed if the specified drive entered on the command line is off line.

UNABLE TO WRITE CLUSTER ALLOCATION TABLE CORRECTLY ON DISK.

This messge is displayed if a write error is encountered while attempting to write the cluster allocation table (CAT) to the drive entered on the command line.

UNABLE TO READ CLUSTER ALLOCATION TABLE CORRECTLY FROM DISK.

This message is displayed if a read error is encountered while attempting to read the cluster allocation table (CAT) from the drive entered on the command line.

UNABLE TO WRITE DIRECTORY CORRECTLY ON DISK.

This message is displayed if a write error is encountered while attempting to write the directory to the drive entered on the command line.

UNABLE TO READ DIRECTORY CORRECTLY ON DISK.

This message is displayed if a read error is encountered while attempting to read the directory from the drive entered on the command line.

INVALID REQUEST.

This message is displayed if anything other than a drive specification is entered on the command line.

INVALID DRIVE.

This message is displayed if the drive specification entered on the command line is not in the range of valid drives.

YOU CAN NOT DOSGEN THE "BOOTED" DRIVE.

This message is displayed if the drive specification entered is the same as that for the "booted" drive.

CHAPTER 21. DSKCHK15 COMMAND

21.1 Purpose

The purpose of DSKCHK15 is to repair a logically-damaged DOS volume. The performance of the DOS is directly related to the correctness of disk-resident system tables.

DSKCHK15 checks all system tables, such as the CAT, directory, etc. for format and content. DSKCHK15 is able to determine in most cases when an error in system data occurs, what the error is, and if the system data can be reconstructed from the other data on the disk.

21.2 Use

DSKCHK15 is invoked by entering:

```
DSKCHK15 [<drivespec>][;options]
```

The only entry necessary on the Command Line is "DSKCHK15". The drive specification may be entered as a standard drive specification (:D0) or as a VOLID (:PAYROLL). If no drive specification is entered on the command line, the program will ask for one. If no options are selected the default is logging to the screen only and the fix option turned off.

21.3 Options

Option letters that may be used are "L" and "F", and to be activated they must be entered on the command line.

21.3.1 "L" Option

Local Printer On - when specified all messages will be printed on the printer as well as displayed to the screen.

21.3.2 "F" Option

System Fix Option - causes the program to compute the correct data when an error in System Data is detected, if this computation is possible, and allows the user to fix the data in error if he so chooses.

If an error in system data is detected, and the correct data can not be computed from the other pertinent usable data on the disk, the operator is so informed and may be asked if he wishes the file or entry deleted. All error correction messages contain a "no-change" option which will cause the program to continue to the next check without changing the data on the disk being checked. The "F" option is illegal and automatically deactivated if DSKCHK15 is run from CHAIN or while JOB15 is active.

21.4 Program Operation

21.4.1 System Tables and Data

Descriptions of the DOS tables, system data and their uses may be found in the chapter on System Structure.

All of the descriptions below are written for the "F" (fix) option active. If this option is not active the operation is the same except that no request for corrective action is made.

21.5 Execution Phases

There are many execution phases in DSKCHK15, some of which are dependent on the type of disk being checked. For example, if a 9320 disk is being checked, the Hashed Directory Index (HDI) will be tested, but the directory mapping bytes will not be tested, since they do not exist on 9320 drives.

21.5.1 Initialization

During initialization the program displays a signon message, and the option parameters are scanned. If the "F" (fix) option was selected the correction features are enabled. If the drive specification was not specified on the command line, it is asked for during this phase.

21.5.2 HDI Checking

This phase is dependent on the type of disk being checked, and is not executed on diskettes, since diskettes use directory mapping bytes.

This phase checks the HDI (Hashed Directory Index) by comparing the master to the backup. If an error is found, an error message is displayed, but no corrective action is allowed at this time. If no errors are detected, an appropriate message is displayed.

21.5.3 CAT Checking

The first phase of checking reads the CAT on the disk being checked, and compares the master to the backup. If an error occurs in reading either copy, or in comparing the copies, an error message will be displayed, but no corrective action is allowed at this time. If no errors are detected an appropriate message is displayed.

21.5.4 Directory Checking

During this phase each directory page is read and checked. The master is compared to the backup copy, and if the compare fails the entry affected is displayed. The program then asks which entry should be retained, or if both entries should be deleted.

When an unsuccessful read occurs in either the master or backup, an error message is displayed and the compare is skipped. The good page is used to continue the checks. If both pages fail to read successfully an appropriate message is displayed and the next page is checked.

If either or both pages read successfully, and when all compare errors are resolved, each entry is checked for valid format. If a deleted entry is encountered it is checked to see that the delete is complete. If it is not, an error message is displayed and the program asks if the error is to be corrected.

The RIB PDA is checked to see that it points to a valid location on the disk. If an error is detected, an appropriate message is displayed, and the program asks if the entry should be deleted. If this entry is deleted, no space for it will be allocated in the computed CAT.

The program then proceeds by checking the filename/ext for valid characters. Each character should fall into the range of A-Z or 0-9. If the program finds an invalid character, a warning message is displayed and the program continues.

For diskettes, the program generates a mapping byte for each page as the page is checked. If the computed mapping byte does not match the mapping byte on the diskette, a message is displayed and the computed mapping byte is entered into the computed CAT sector being generated.

For 9320 disks, a HDI byte is generated for each file in the directory page being checked. If the computed hash byte does not match the master HDI byte for that file, a warning message is displayed, but no corrective action is allowed at this time.

21.5.5 RIB Checking

After all directory pages have been checked, the RIB for each entry in the directory is checked for format and space allocation validity. The RIB master and backup are first read and compared. If either copy is not read successfully, the program asks if the good copy is to be written to the copy with the read failure. If a read failure occurs on both copies an error message will inform the user and the program will ask if the file is to be deleted. No space will be allocated for that file in the computed CAT.

If a compare error is detected, the filename and the first 16 bytes of both copies is displayed and the program will ask which copy is to be retained on the disk.

After the RIB master and backup have been read and compared, and all errors resolved, the RIB format is checked for the correct PFN, LRN, and an 0377 in the 4th byte. Any format error detected will be displayed, and the corrective action requested.

Each segment descriptor is then checked to see that it points to a valid location on the disk, and that the first sector of the segment has the proper PFN. The space allocation is computed and checked against the CAT and Lockout CAT for conflicts. If a conflict occurs with the Lockout CAT the program will ask if the Lockout CAT is to be rewritten to free this space. If a conflict is detected with another file, the conflicting area is analyzed and displayed with a request for action. If no conflicting areas are found, the next two phases are skipped and the program continues with the Lockout CAT check.

21.5.6 Cluster Allocation Phase 1

If a conflict was detected while checking the RIBs this phase is activated to reread all the good RIBs and find all other conflicts.

21.5.7 Cluster Allocation Phase 2

Upon the determination of all conflicting clusters, they are scanned to see which RIBs conflict and an opportunity to delete one or another or both conflicting files is given. If no change is requested, the CAT is updated with the conflicting clusters so there is protection from another file writing into this area.

21.5.8 Lock-out CAT Checking

If no change in the Lockout CAT has been made previously, it is now read and compared to its backup and checked against the CAT to see that all locked out space is also allocated in the CAT. If changes were made in previous phases of the program the generated Lockout CAT is checked against the CAT and if no errors are detected it is written to the disk.

Any errors detected in this phase cause a descriptive message to be displayed and the program asks if a new Lockout CAT should be written. This new Lockout CAT, if written, will only have the space for the system tables locked out.

21.6 Examples of Command Lines

The following are examples of Command Line entries:

DSKCHK15 :D0 will cause the error messages to go to the screen, and only verification of system tables will be active.

```
DSKCHK15 :D0;L
```

This causes messages to be printed as well as displayed on the screen.

```
DSKCHK15 :D0;LF (or FL)
```

In this example error messages are printed as well as displayed, and the fix option is active. This means that if any errors are detected, the operator will be asked for a response.

21.7 Operational Messages

These are the informational messages. They keep the operator informed of what the program is doing.

CHECKING HDI

> The program is checking the HDI master and backup for read errors in either, and for a match between the two.

CHECKING C.A.T. FOR FORMAT

> The program is reading the CAT master and copy and checking for read errors in either, and for a match between the two.

C.A.T. MASTER AND BACKUP LOOK O.K.

> The CAT master and backup were read without error, and the master compared correctly with the backup.

CHECKING DIRECTORY PAGE nnn

> The directory page nnn master and copy have been read without error. This message also indicates that the directory entries on this page are being checked for format. During this check a directory mapping image is also being constructed and checked against the one on the diskette if necessary.

CHECKING RIBS

> The directory checking is complete, and the RIB' s are now being checked.

CHECKING RIB FOR PFN nnn

The RIB for PFN nnn is now being read and checked for format errors and allocation conflicts.

CLUSTER ALLOCATION PHASE 1

If cluster conflicts occur while checking RIBs and the 'F' option is set, this phase is executed to verify and find all conflicts.

CLUSTER ALLOCATION PHASE 2

This phase follows phase 1 above to resolve any or all conflicts.

CHECKING LOCK-OUT C.A.T.

The lock-out CAT master and backup are being read and compared, and checking that the locked-out space is allocated in the CAT.

ALL LOCKED-OUT CYLINDERS ARE ALLOCATED IN THE C.A.T.

Indicates that no read errors occured, the master matched the copy, and that all cylinders locked-out are allocated in the CAT.

MATCHING COMPUTED C.A.T. TO DISKETTE

or

MATCHING COMPUTED C.A.T. TO DISK

The CAT computed from the R.I.B.s and the computed directory mapping bytes are being compared against that read from the diskette or the disk.

COMPUTED C.A.T. MATCHES DISKETTE

or

COMPUTED C.A.T. MATCHES DISK

The CAT read from the diskette or disk matched the data computed from the R.I.B.s and that the directory mapping bytes (when applicable) match.

COMPUTED HDI MATCHES DISK

The HDI computed during directory checking matches the master HDI read from disk.

DSKCHK15 DONE

All checking functions that were active are complete and control is being returned to the DOS.

21.8 Error Messages

The following error messages can occur during the execution of DSKCHK15:

21.8.1 Initialization Error Messages

DRIVE OFF LINE

When accessed the drive being checked was found to be off line.

INVALID DRIVE

The drive selected was not in the range of valid drives.

INVALID OPTION PARAMETER
VALID OPTIONS ARE L=LIST F=FIX

An option other than those listed was detected.

ENTER DRIVE (LIKE :D0, :D1, OR :VOLID)

An invalid drive specification or no drive specification was detected.

PROGRAM NOT LOADABLE

One or more of the library members of DSKCHK15 are missing. The members in DSKCHK15 are:

```
DSKCINIT
DSKCVER1
DSKCVER2
DSKCVER3
DSKCVR3B
DSKCVR3C
DSKCVER4
```

CONCURRENCY ACTIVE - "F" OPTION HAS BEEN DEACTIVATED

The fix (F) option is not allowed with concurrency active.

THIS PROGRAM IS RUNNING FROM A CHAIN - THE "F" OPTION HAS BEEN DEACTIVATED

The fix (F) option is not allowed during the execution of CHAIN.

```
*** BAD PDA *** NO READ OR WRITE TO DISK(ETTE)
DSKCHK15 CANNOT REPAIR (MSB,LSB)
```

An invalid PDA was found in a RIB. DSKCHK15 cannot repair this error.

```
** ERROR ** READ FAILURE AT DOS PDA (MSB,LSB)
```

A hard parity error was found at the PDA indicated. This indicates a physically bad disk, which DSKCHK15 cannot repair.

21.8.2 C.A.T. Errors

```
ERROR ON READ OF C.A.T. MASTER
```

An un-recoverable read error occured during the read of the master C.A.T. sector. No action is taken at this time.

```
ERROR ON READ OF C.A.T. BACKUP
```

An un-recoverable read error occured during the read of the backup C.A.T. sector. No action is taken at this time.

```
C.A.T. MASTER AND BACKUP DO NOT MATCH
C.A.T. WILL BE RECONSTRUCTED FROM THE RIBS
```

The CAT master and backup buffers in memory do not compare correctly. No action is taken at this time. When a computed CAT has been constructed, it will be compared to the master and backup CAT, and at that time corrective action may be taken.

Before the computed CAT is compared to the disk, the program checks internal error flags, and if any of the errors above are found to have occured, one or more of the following messages may be displayed.

ERROR ON READ OF C.A.T. MASTER - WRITE BACKUP TO MASTER ?

An error was detected on the initial read of the CAT master, but the backup copy was read successfully. Answering "Y" causes the backup to be written to the master. The "N" answer causes no change.

ERROR ON READ OF C.A.T. BACKUP - WRITE MASTER TO BACKUP ?

An error was detected on the initial read of the CAT backup, but the master copy was read successfully. Answering "Y" causes the master to be written to the backup. The "N" answer causes no change.

BOTH CAT COPIES BAD - WRITE COMPUTED CAT TO BOTH ?

Neither copy of the CAT could be read successfully. Answering "Y" causes the computed CAT to be written to the CAT master and backup sectors on the disk. The "N" answer results in no change.

CAT MASTER AND BACKUP DID NOT MATCH
WRITE COMPUTED CAT TO DISKETTE ?

or

CAT MASTER AND BACKUP DID NOT MATCH
WRITE COMPUTED CAT TO DISK?

After the initial read the CAT master and backup did not compare. Answering "Y" causes the computed CAT to be written to the CAT master and backup sectors on the diskette or the disk. The "N" answer results in no change.

```
COMPUTED CAT DOES NOT MATCH DISKETTE
WRITE COMPUTED CAT TO DISKETTE ?
```

or

```
COMPUTED CAT DOES NOT MATCH DISK
WRITE COMPUTED CAT TO DISK?
```

The computed CAT constructed from the RIBs does not match the CAT on the diskette or the disk. Answering "Y" causes the computed CAT to be written to the CAT master and backup sectors on the disk or the disk. The "N" answer results in no change.

21.8.3 HDI Errors

```
HDI MASTER AND BACKUP DO NOT MATCH
HDI WILL BE RECONSTRUCTED FROM DIRECTORY
```

The Hash Directory Index master and backup did not match and will be reconstructed during the final phase.

```
ERROR ON READ OF HDI MASTER
```

Parity error found while reading the HDI master.

```
ERROR ON READ OF HDI BACKUP
```

Parity error found while reading the HDI backup.

```
WRITE NEW HDI TO DISK?
```

With the fix option set, the operator will be able to try to write a new HDI.

21.8.4 Directory Errors

The following error messages may appear when the program is checking the directory.

ERROR ON READ OF DIRECTORY MASTER

> The current directory master page could not be read successfully. If the "F" option is active the request "WRITE BACKUP TO MASTER ?" will also appear. If this request is answered "Y" the current backup page will be written to the correct directory master sector on the disk.

ERROR ON READ OF DIRECTORY BACKUP

> The current directory backup page could not be read successfully. If the "F" option is active the request "WRITE MASTER TO BACKUP ?" will also appear. If this request is answered "Y" the current master page will be written to the correct directory backup sector on the disk.

BOTH PAGES READ BAD
NO SPACE WILL BE ALLOCATED FOR THE FILES ON THIS PAGE

> Neither the master nor backup pages could be read successfully. The page can not be fixed and no space will be allocated in the computed CAT for the files on this page, nor will the directory mapping byte for this page be updated.

DIRECTORY ENTRIES DO NOT MATCH - PFN nnn

> The current master and backup page do not match. The master and backup copies of the entry that do not match will be displayed in one of the formats below:

```
DIRECTORY ENTRIES DO NOT MATCH - PFN 000
MASTER -
  001 303 000 000 123 131 123 124 105 115 062 040 123 131 123 377
                   S   Y   S   T   E   M   2       S   Y   S
BACKUP -
  001 003 000 000 123 131 123 124 105 115 060 040 123 131 123 377
                   S   Y   S   T   E   M   0       S   Y   S
ACTION: 1=MASTER>BACKUP 2=BACKUP>MASTER 3=DELETE BOTH 4=NO CHANGE ?
```

```
DELETE INCOMPLETE PFN nnn
WRITE DELETE TO DIRECTORY ?
```

The entry for PFN nnn was found to have one or more bytes other than 0377. The "Y" answer causes a deleted entry to be written to the directory for this entry. The "N" answer causes no change.

```
INVALID PDA IN PFN - nnn
DELETE THE ENTRY ?
```

The entry for PFN nnn points to a physical disk address that is not valid for a RIB location. The "Y" answer causes the entry to be deleted from the directory master and backup. The "N" answer causes no change.

```
***WARNING*** FILENAME/EXT FOR PFN nnn
CONTAINS INVALID CHARACTERS
```

The filename/ext for PFN nnn contains characters that are not in the range of A-Z or 0-9. This is a warning only, and does not allow any corrective action.

```
DIRECTORY MAPPING BYTE DOES NOT MATCH GENERATED BYTE FOR PAGE nnn
```

The directory mapping byte generated for the current page (nnn) does not match the mapping byte on the diskette. The generated byte is entered into the directory mapping area in the computed CAT and will cause a match error when the computed CAT is compared to the CAT on the diskette.

21.8.5 R.I.B. Errors

The error message and display formats that follow may all be displayed in the RIB checking phase of program execution.

```
ERROR ON READ OF RIB MASTER
ACTION: 1=BACKUP>MASTER 4=NO CHANGE ?
```

The RIB master could not be read successfully, but reading the backup was successful. The "1" answer causes the backup RIB to be copied to the master with the correct LRN. The "4" answer causes no change on the disk, and the backup RIB is used for further checks.

```
ERROR ON READ OF RIB BACKUP
ACTION: 2=MASTER>BACKUP 4=NO CHANGE ?
```

The RIB backup could not be read successfully, but reading the master was successful. The "1" answer causes the master RIB to be copied to the backup with the correct LRN. The "4" answer causes no change on the disk.

```
MASTER AND BACKUP READ ERRORS - DELETE THE FILE ?
```

Neither copy could be read successfully. No recovery of data is possible, and the file may be deleted from the disk, or no change may be made. Either action results in no space for this file being allocated in the computed CAT.

RIB MASTER AND BACKUP DO NOT MATCH

The RIB master and backup copies did not compare correctly. Only the area of the RIB used by the system is compared. The responses are self- explanatory. For any action except "3" the resulting master is used for checking allocation. If the file is deleted, no further checks are made, and no space is allocated in the constructed CAT.

```
RIB MASTER AND BACKUP DO NOT MATCH  ***  FILE - filename/ext
RIB MASTER -
  000 000 000 377 001 004 377 377 377 377 377 377 377 377 377 377
RIB BACKUP -
  000 001 000 377 001 104 377 377 377 377 377 377 377 377 377 377
ACTION: 1=MASTER>BACKUP 2=BACKUP>MASTER 3=DELETE BOTH 4=NO CHANGE ?
```

RIB FORMAT ERROR

One or more of the first four bytes of the RIB was incorrect. The total message will display in the format below. Only the error explanations and pointers for which errors were detected will be displayed.

```
RIB FORMAT ERROR *** FILE - name/ext
000 123 125 105 000 377 377 377 377 377 377 377 377 377 377 377
 ^   ^   ^   ^

 1   2   3   4

1 - PFN INCORRECT
2 - LRN LSB NOT 000
3 - LRN MSB NOT 000
4 - 4TH BYTE NOT 0377


ACTION: 1=DELETE THE FILE 2=CORRECT ERROR(S) 3=NO CHANGE ?
```

```
RIB BACKUP LRN ERROR
CORRECT THE ERROR ?
```

The RIB backup LRN was not 001, but the rest of the backup matched the RIB master. The "Y" answer causes the correct LRN to be written to the RIB backup sector.

INVALID SEGMENT DESCRIPTOR # nnn - RIB WILL NOT BE CHECKED

The segment descriptor for segment nnn pointed to a location that can not physically exist on the disk being checked. No further checking of the RIB will occur.

SPACE ALLOCATION CONFLICTS WITH PREVIOUS ALLOCATION

The space allocated for the current segment of this file is in conflict with space allocated to a previously checked file. Additional information is displayed in the format below.

```
                        FILENAME/EXT  *  FILENAME/EXT
                            PFN  000  *  PFN  001
CONFLICTING CLUSTERS             007  *       007
CLUSTERS IN FILE                 005  *       002
```

```
ACTION: 1=DELETE PFN 000 2=DELETE PFN 001 3=NO CHANGE ?
```

This indicates that PFN 000 and PFN 001 have 7 clusters in conflict, and that 5 of them have records from PFN-000 and 2 have records from PFN-001. In this example PFN-001 should probably be deleted.

CURRENT FILE OVERLAYS LOCKED OUT SPACE
FREE THIS SPACE IN LOCKOUT CAT ?

Some of the space allocated to the file being checked occupies space that is locked-out. Answering "Y" causes the cylinders in the LOCKOUT CAT which the file overlays to be removed from the LOCKOUT CAT. The "N" answer causes no change in the LOCKOUT CAT.

INVALID PFN IN FIRST SECTOR OF SEGMENT nnn *** FILE - FILENAME/ABS

The PFN in the first sector of segment nnn does not match the PFN of the file being checked. This message is informational, as this may be a normal condition in some files. The program will take no corrective action.

21.8.6 Lockout CAT Errors

LOCKOUT MASTER AND BACKUP DON'T MATCH
WRITE NEW LOCKOUT CAT TO DISKETTE ?

or

LOCKOUT MASTER AND BACKUP DON`T MATCH
WRITE NEW LOCKOUT CAT TO DISK?

The Lockout CAT master and backup do not compare. Answering "Y" causes a Lockout CAT with only the space for the System Tables locked-out to be written to the diskette or the disk. The "N" answer causes no change.

LOCKED OUT CYLINDER nnn NOT ALLOCATED IN CAT
WRITE NEW LOCKOUT CAT TO DISKETTE ?

or

LOCKED OUT CYLINDER nnn NOT ALLOCATED IN CAT
WRITE NEW LOCKOUT CAT TO DISK?

The locked out cylinder nnn is not allocated in the CAT. Answering "Y" causes a Lockout CAT with only the space for the System Tables locked-out to be written to the diskette or the disk. The "N" answer causes no change.

ERROR ON READ OF LOCKOUT CAT
WRITE NEW LOCKOUT CAT TO DISKETTE?

or

ERROR ON READ OF LOCKOUT CAT
WRITE NEW LOCKOUT CAT TO DISK?

The LOCK-OUT CAT read was in error possibly due to a parity error or a sector not found. The user now has the opportunity to try to write the calculated LOCK-OUT CAT back to the diskette or the disk.

CHAPTER 22. DUMP COMMAND

22.1 Purpose

Occasionally while writing into files on disk (in particular, during the program debugging stage) it is useful to be able to verify that the formatting of the information into the standard text format is being done correctly. Or perhaps a random-access file must be inspected, one that is not in the standard text format.

The DUMP command provides a simplified mechanism for examining the entire contents of physical sectors on a disk. The display includes both the octal and ASCII contents of every byte on the sector. No examination for control bytes of any kind is made, allowing the user to see the precise contents of every physical location in the disk sector.

22.2 Use

The DUMP command is invoked by entering:

```
DUMP
```

or

```
DUMP <file spec>
```

The DUMP command operates with basically five separate levels of control. These levels are:

LEVEL ONE - Logical drive level
LEVEL TWO - File level
LEVEL THREE - Logical record number level
LEVEL FOUR - Physical disk address level
LEVEL FIVE - Disk directory level

The (optional) entry file and/or drive specifications on the command line allow the first one or two input levels in DUMP to be automatically bypassed.

When the DUMP command is used, the top line of the display is the primary control line. Input is accepted on this line. This line is broken into four basic areas, one corresponding to each of the first four control levels. The primary control level at any given time during the operation of the DUMP command can be determined by the position of the flashing cursor on the control line.

For example, if the flashing cursor is positioned after the "DRIVE:" legend on the control line, the DUMP command is operating at level one. If the cursor is positioned after the "FILE:" legend on the control line, the DUMP command is operating at level two, etc.

22.3 Informational Messages Provided

The second line on the display is primarily used for sector informational messages. These serve both to indicate any special significance of the sector just read and to describe any unusual occurrences associated with reading the sector. These messages are generally self-explanatory. Among the messages that can be displayed are the following, along with an explanation of the meaning of each.

RETRIEVAL INFORMATION BLOCK (RIB). This message indicates that the sector being displayed is the primary RIB for the currently opened file.

RETRIEVAL INFORMATION BLOCK BACKUP. Each RIB is maintained in duplicate for backup purposes and to allow recovery in the event of a program erroneously destroying the primary RIB. This message indicates that the sector being displayed is the secondary RIB for the currently opened file.

CLUSTER ALLOCATION TABLE. This message indicates that the sector being displayed is the primary Cluster Allocation Table (normally referred to as the CAT) for the current logical drive.

CLUSTER ALLOCATION TABLE BACKUP. This message indicates that the sector being displayed is the secondary backup CAT for the current logical drive. The CAT is also maintained in duplicate just as is the RIB.

LOCKOUT CLUSTER ALLOCATION TABLE. Associated with each logical drive is a sector that indicates which areas have been locked out, prohibiting their use by DOS. This message indicates that the sector being displayed is the Lockout CAT for the current logical drive.

LOCKOUT CLUSTER ALLOCATION TABLE BACKUP. This message indicates that the sector being displayed is the secondary backup copy of the sector.

SYSTEM DIRECTORY SECTOR. This message indicates that the sector being displayed is one of the DOS directory sectors. The directory sector number (in decimal and in octal) immediately follows the message.

USER DATA SECTOR. This message indicates that the sector is not recognized as one of the above special system sectors.

DISK SECTOR CRCC ERROR. This message indicates that the sector requested for display either was not found on the disk or that a CRCC (Cyclic Reduncancy Check Code or parity) error repeatedly occurred during the read operation. The sector displayed is the data as it was read from the disk, unless the sector was not found.

DISK OFFLINE. This message indicates that the currently specified logical drive is not on line.

DISK SECTOR FORMAT ERROR. This message is displayed when DUMP notices that the sector being displayed does not correspond to standard DOS file conventions (the first byte of each sector is its physical file number, and the two following bytes are the logical record number). The appearance of this message does not necessarily indicate that the sector of the file has been destroyed, since unwritten sectors at the end of a file will fall into this class. It merely means that if the sector were read with the DOS READ$ routine, a format trap would occur.

SECTOR OUT OF RANGE. This message is displayed if the sector requested (by logical record number) is not within the range of the currently opened file.

FILE NOT FOUND. This message indicates that the file requested could not be found. This does not necessarily mean that the file does not exist. For example, the file could be in a non-current subdirectory. If the user has not requested non-specific volume mode (to be described), this message might mean simply that the file desired is on a different logical drive.

INVALID PHYSICAL ADDRESS. This message indicates that the physical disk address specified is invalid.

The remainder of the display contains the contents of the current sector most recently read. The display is arranged as sixteen groups of sixteen bytes each. Each of these groups is preceded by the three octal digit offset of that group within the sector. Each sixteen byte group consists of the octal and ASCII contents of each of the sixteen bytes in that group. Each byte's contents form a column one character wide and four lines high, where the first three lines are the value of the byte, in octal, and the fourth line is the ASCII value of that character. Notice that the character is not examined for special significance before it is displayed, so DUMP may display characters other than the normal ASCII set.

22.4 Level One Commands To DUMP

When the flashing cursor indicates that DUMP is functioning at level one, the following commands are accepted:

<enter> - The CAT on the current drive is displayed and control is transferred to level two. In addition, the non-specific drive mode is enabled.

number - The drive number indicated becomes the currently selected drive. The CAT from that drive is displayed and control is transferred to level two. Non-specific drive mode is disabled.

* - DUMP command returns control to the DOS.

22.5 Level Two Commands To DUMP

When the flashing cursor indicates that the DUMP command is functioning at control level two, the following commands are accepted:

<enter> - If a file is currently opened, the secondary RIB for the file is displayed and control is transferred to level three. If no file is opened, control is transferred to level four.

name/ext - The named file is opened on the current drive, or any drive if non-specific drive mode is enabled. The primary RIB for the file is displayed and control is transferred to level three.

pfn - The file indicated by the octal physical file number given is opened on the current drive. The primary RIB for the file is displayed and control transfers to level three.

I - The current physical file number is incremented and the new file thus indicated is opened. If no file corresponding to that physical file number exists on the current drive, the PFN is incremented repeatedly until a file corresponding to the PFN is found. The primary RIB for the file is displayed and control is transferred to level three.

D - D works just like the I command above except that instead of incrementing the PFN, it is decremented.

#pfn - The directory sector containing the entry corresponding to the file indicated by the specified physical file number is displayed; then control is transferred to level five. Since only the last four bits of the PFN are relevant, the pfn specifier is equivalent to a relative directory sector number. These directory sector numbers are always specified in octal.

* - Return control to level one.

22.6 Level Three Commands To DUMP

When the cursor indicates that DUMP is functioning at level three, the LRN level, the following commands are accepted:

<enter> - The current sector is shown and control is transferred to level four.

number - Access and display the record indicated by the LRN specified. If the number given has a leading zero, it is assumed to be octal; otherwise it is assumed to be decimal. The number specified is the <u>user</u> (as opposed to system) LRN. The <u>system</u> LRN, the value in bytes one and two in the sector, is always two greater than the user LRN. The two numbers displayed at level three in the control line are the user LRN in decimal (the one with leading zeros suppressed) and octal (the one in parentheses, with leading zeros).

I - Increment the current logical record number, access it and display the sector.

D - Decrement the current logical record number, access it and display the sector.

* - Return to the File level of control (level two).

22.7 Level Four Commands To DUMP

Level four of the DUMP command requires more detailed understanding of DOS physical disk addresses, and as such is not usually as useful as the LRN level. However, when access to a specific sector on the disk is desired, it can be achieved using DUMP level four. It is important to realize that the physical disk addresses specified are logical physical disk addresses, i.e. the DOS PDA as defined in chapter 42. They are not necessarily the same as actual physical locations on the disk. The logical disk addresses are remapped onto the disk into different hard physical sector numbers than those indicated by the logical physical disk address. The important thing to understand here is that the disk addresses used in the level four control of DUMP are the two-byte PDA: cylinder, cluster/sector.

The commands accepted at level four of DUMP are as follows:

msb,lsb - Access and display the sector indicated at the given physical disk address on the current logical drive. The first field (most significant byte) is assumed to be in decimal unless a leading zero is supplied. The second field (least significant byte) is always considered to be in octal, regardless of whether a leading zero is supplied or not. The second field is separated from the first by a comma. The physical disk address given by the user is assumed to be valid. If it is not of the proper format, undefined results may occur. Users who are not sure of their understanding of DOS internal physical disk addresses should not use level four of DUMP.

I - Increment to the next logical sector.
D - Decrement to previous logical sector.
* - Return control to level two if no file is opened, or level three otherwise.

22.8 Level Five Commands to DUMP

When the flashing cursor indicates that the DUMP command is operating at control level five (system directory sector level), the following commands are accepted:

number - Show the directory sector indicated by the low order four bits of the number specified. Since only the low order four bits of the number are used, it is not an error to specify simply the physical file number (PFN) of the file whose directory entry is to be examined. A leading zero indicates the number is in octal, otherwise decimal is assumed.

I - The current directory sector number is incremented and the corresponding directory sector is displayed.

D - The current directory sector number is decremented and the corresponding directory sector is displayed.

* - Return control to level two.

22.9 Error Messages

Only one error message is issued by the DUMP command. It is:

```
ERROR IN DOS FUNCTION.  DUMP ABORTED.
```

If this error message occurs, it means that the DOS FUNCTIONs are probably incorrect on the disk, generally indicating that the disk in the booted drive has not been completely (or correctly) DOSGENed. If this is the case, SYSTEM7/SYS should be loaded using the latest copy of DOS as distributed by Datapoint.

CHAPTER 23. EDIT COMMAND

The 1500 Editor is a vastly enhanced editor. All available memory is used as a circular buffer which allows rolling backwards through a file. Users may combine the primitive EDIT commands to create "user-defined" commands for repetitive editing tasks. The use of two scratch files protects the input file from modification until EDIT is completed. Text files now carry their individual configuration (tabs, etc.) in their first sector. Many command enhancements and additions have been made for consistency and usability.

23.1 Introduction

The 1500 Editor is used to create and to update source data files on a disk. The editor enables the creation of files in a variety of formats: text files, DATABUS source code files, or many user-designed data files.

A Glossary of the many terms and phrases used throughout this section is provided in the section 'Glossary of EDIT Terms'. A list of commands and brief definitions is provided in the 'Command List' section. Also included at the end of the chapter is an explanation of error messages, the format of the configuration sector and an example of an EDIT/DEF file.

23.2 General Description of The 1500 Editor

All available memory is used as a circular buffer which allows rolling backwards through a file. Some files may fit completely in memory so there is no time-consuming copying back and forth to a SCRATCH file. This also speeds up FINDs and LOCATEs and reduces disk "thrashing".

The 1500 Editor offers additional file security since the original source file is not modified until the :E command is entered. The use of two scratch files protects the input file from modification until the 1500 Editor is completed. It also allows multiple passes on a file using the 'ONE-PASS' option.

This does, however, slow the end of edit since text in memory must be written out to the scratch file (if in use) and then the scratch file must be completely copied to the original file.

Text files carry their individual configuration (tabs, etc.) in their first sector.

Command enhancements have been made for consistency and usability for the user. For instance, MODIFY has a VERIFY option and FINDs and LOCATEs allow a field parameter and display the full screen.

23.3 Special Features of the 1500 Editor

Some of the enhancements over prior Editors include:

** EDIT allows rolling backward on the screen.

** EDIT does not open a new file until it checks with the user that a new file is to be created.

** Null lines (that is, lines that have been deleted or scratched) do not appear on the screen.

** Default configuration information such as tabs, special characters and modes is preserved with each file.

** Space-compression is an option.

** Tabs may be set by integer column number as well as by spacing across the tab ruler.

** Field designation is part of the :D, :F, and :L commands.

** Keyclicking may be turned on and off during execution (:K, :KI).

** The get command (:G) can be used to roll the screen forward or backward a specified number of lines, or to roll the screen to a specified line number in the file.

** :E\ can be used to truncate the beginning of the file.

** :E- rolls backwards to the beginning of the text contained in memory.

** The modify commands may all be used with the verification option.

```
** Other commands are:
    :A*          Append without moving the pointer.
    :D*          Display the last DELETE string.
    :I [string]  Insert the specified string.
    :O           Abort; go back to DOS without modifying the
                 file.
    :OX [DOS Command]     Abort to DOS, then execute
                     the specified command.
    :V           Split the pointed line into two lines.
    :W           Concatenate the pointed line with the one
                 below it.
```

The EDIT commands may be combined to form 'user-defined' command strings (:0 through :9). Additional commands allow conditional or unconditional skips along the command string. These command strings may be pre-defined in an EDITable file (default EDIT/DEF) which is automatically loaded when EDIT is executed.

EDIT will recognize configuration sectors of most other Datapoint Editors which are similar to EDIT in operation and appearance.

23.4 Software Required

The 1500 Editor requires a DOS. H operating system.

23.5 Operation

23.5.1 Invoking the 1500 Editor

The 1500 Edit program command line syntax is shown below. Items in square brackets are optional;f1, f2, and f3 are file specifications; the parameter list consists of abbreviated notations which cause special functions to occur.

```
EDIT <f1>[,<f2>][,<f3>][;parameter list]
```

23.5.2 Files

<f1> is the source file, [<f2>] is the scratch file and [<f3>] is the definition file. The source file <f1> is assumed to have an extension of TXT if none is provided. If there is no file of the specified name, a new file is created after checking with the operator. If no scratch file [<f2>] is specified, a primary scratch file 'SCRATCH/TXT' and a secondary scratch file 'SCRATCH/XTX' are used. Since the second scratch file will be named <scratch file name>/XTX, the extension XTX for <f2> is NOT allowed. The definition file [<f3>] is assumed to be EDIT/DEF unless otherwise specified.

23.6 Parameter List

A parameter list, indicated by the SEMI-COLON (;) following the file specifications, may be included. That list may include nine parameters which are order independent. The possible parameters are:

```
[;[margin][tab key][mode][shift][line][update]
[keyclick][space-compression][non-verification]]
```

At the start of an edit, the values for these nine parameters are taken from the command line. Values for [margins], [tab key], [mode], [shift], [line], [key click] and [space-compression] not given on the command line are taken from the source file configuration sector, if there is one. The source file is updated unless otherwise specified on the command line and verification mode is assumed. Care should be exercised to be sure that not more than one mode, margin, etc. is specified on the command line or the desired value may not be selected.

When a file has been edited with EDIT, a special sector called the "configuration sector" is written at the beginning of the file. The sector contains the tabs, modes and special characters in effect when the edit was completed. These default values are used in place of any such parameters not specified on the command line.

If no command line parameter list is provided and the source file has no configuration sector, or an unrecognizable one, then tabs are set at 9, 15 and 30 with space-compression, a margin at 75, no keyclick and the space bar for tabbing is assumed.

23.6.1 Margin Bell

A number in the parameter list is taken to be the margin designator; this causes the margin 'bell' to ring at the designated margin. The default margin is 75. Allowable values are 1 through 79.

For example ";30" causes the processor to beep in column 30.

23.6.2 Tab Key Character

A tab key character encountered in the parameter list, i.e., non-alpha, non-numeric, non-colon, non-command special character (\,<,>,etc.) replaces the assumed tab key character (SPACE BAR, in DATABUS mode; SEMI-COLON in Text mode.)

For example, ";^" causes the caret key (^) to replace the assumed character as the tab key. Note: Special characters such as #, &, >, <, \, and blank are not allowed as tab key options.

23.6.3 Mode

A different set of assumptions is used if one of the 'mode' parameters is set. DATABUS / DATAFORM (D) mode sets the tab stops to 10 and 20, tab key to SPACE BAR, no word wraparound, space insertion after a period, plus or asterisk in horizontal column one, and shift inversion on. The Assembler (A) mode is the same except tabs of 9, 15, and 30.

Text mode (T) sets no tabstops, does no shift inversion and enables the word wrap-around feature . The [shift] option 'S' and the [line] option 'L' are recognized only if the text mode 'T' is set (either on the command line or in the configuration file). To activate line truncation instead of word wrap-around in Text mode, enter 'L' in the parameter list. To enable shift key inversion in Text mode, enter the parameter 'S' in the list.

See the Edit Glossary (23.33) for complete definitions of the various modes.

23.6.4 Update

During editing, the source file is transferred into the scratch file as the text is updated. A second scratch file may also be used as the edit proceeds. When the edit is terminated, the physical source file is normally updated.

The 'ONE-PASS' parameter 'O' may be set in the parameter list. Then, at the completion of the edit, the scratch file contains the updated information and the source file is unchanged.

23.6.5 Key-click

If the 'K' parameter is set, a 'click' sounds each time a key is struck.

23.6.6 Space Compression

EDIT normally space-compresses. If an 'E' appears as a parameter on the command line, spaces are "expanded", that is, NOT space-compressed. In expanded mode, EDIT reads in either space-compressed or non-space-compressed data but puts out only non-space-compressed records. To clear expanded mode, enter a 'G' on the command line. The space compression option (either 'E' or 'G') is stored with the file in the configuration sector, but may be changed by putting the desired option on the command line.

23.6.7 Non-verification

Four EDIT commands: :E/, :E\, :B, and :O question the user:

```
SURE?
```

before executing. In certain special cases, for instance, when running under CHAIN, it is inconvenient to provide the 'Y'. The non-verification parameter on the command line allows the user to answer 'YES' in advance by placing a 'Y' on the command line. When 'Y' has been placed on the command line and the commands :E/, :E\, :B, or :O are entered, no verification is requested from the user.

23.7 Examples

To perform standard editing, enter the command:

```
EDIT <source>
```

to edit a text file enter the command:

```
EDIT <source>;T
```

To also change the margin bell to ring at column 35 (e.g. for labels) enter the command:

```
EDIT <source>;35T
```

The parameters set the bell at 35 and select the Text mode. Note that the parameters are not order dependent; therefore, the command:

```
EDIT <source>;T35
```

achieves the same results.

```
EDIT <source>;E
```

produces a file in which all spaces are written out (or non-space-compressed).

To write the edited text into a second file, without updating the original file, enter the command:

```
EDIT <source>,<new file>;OT
```

If the file is DATABUS program code, replace 'T' with 'D'.

A second file, with the same name as <f1> but with a different extension, may be used as the scratch file by entering:

```
EDIT <f1>,/<extension>
```

Note that the extension XTX is not allowed for the scratch file specification.

Once the initial command (and parameter list) has been entered, the Editor sign-on message will appear on the screen followed with the file's configuration information (e.g. tabs, special characters, margin, keyclick, word wrap-around, shift inversion and space-compression) with the cursor left on the 'command line'. From this position data may be entered, lines may be fetched from the source file, or editor commands may be entered.

23.8 Data Entry and Retrieval

23.8.1 Data Entry

To enter text, simply type on the bottom or "command" line. When the ENTER key is presssed the screen rolls up one line. The command line is once again blank and the cursor is at the beginning of the command line, ready to accept more input.

When a SPACE is typed to the right of the margin bell column (except in column 79) and word wrap-around is enabled, the editor will automatically roll up the screen and begin a new line. If a non-space character is typed into the last column of the screen and word wrap-around is enabled, the last word on the line is removed and, after the screen is rolled up, that word is placed on the command line, where data entry may proceed.

The lines that have been rolled above the command lines are referred to as "screen lines".

When typing on a "screen line" (as the result of a command), the ENTER key causes the cursor to return to the command line. To continue data entry at the same screen area, the Pseudo-ENTER key may be used. This key (shift DEL) causes (in all but command mode), a new blank line to be inserted at that point on the screen so that data entry may proceed.

The BACKSPACE key erases the last character and moves the cursor back one position. The CANCEL key erases the line back to the previous tabstop, or back to the beginning of the line if no tabs are set.

Typing the tab key character causes the cursor to move to the next tab stop to the right. If there are no tab stops to the right of the cursor, the tab key character is accepted as a normal data character.

23.8.2 Multi-line Record Entry

A record in a disk file is a string of characters, perhaps including space compression characters, that is terminated with an octal 015. Normally the length of a record is no greater than 80 characters, including the octal 015, and corresponds to a screen line.

Multi-line records are records in a disk file that are greater than 79 characters long and must be continued to the next line when displayed on the screen.

To enter multi-line records, use the continue character. The assumed continue character is '&'. When this character is keyed in the first column, the continue indicator (a solid triangle) is displayed. If the continue character is keyed in any other column, it is accepted as a normal data character.

Note that although multi-line records are indicated on the screen, the editor treats them as if they were separate lines. The copy command, for instance, copies only the pointed line and NOT all of the lines in the multi-line record. The only time EDIT recognizes lines as part of a multi-line record is during output.

Any line beginning with a continue indicator is joined to the previous line on output, forming records greater than 79 characters, if required. Note that records of greater than 253 characters are not supported by EDIT.

When creating or modifying multi-line records, trailing spaces on a line often need to appear to force the line to be greater than 79 characters. In EDIT, trailing spaces are NOT deleted when they have been entered or read in from a file. However, when EDIT modifies, appends or copies a line, trailing spaces are deleted from that line. Note that to update a multi-line record with trailing spaces the record must be re-entered.

23.9 Data Retrieval

To fetch data from the source file, _hold down the display (DSP) key and press the keyboard (KBD) key._ As long as the two keys are depressed, data is fetched, displayed on the command line and rolled up the screen. If the end of file is reached, no more data is fetched and the machine beeps.

To fetch data backwards, _hold down the keyboard (KBD) key and_ press the display (DSP) key. As long as the two keys are depressed, the screen rolls down and prior lines inserted on the top line. If the beginning of the file contained in memory is reached, the machine beeps, and no more data is fetched.

To fetch a single line, the shifted DEL key may be pressed. Using this key insures that only one input line is fetched.

All of available memory is used as a circular buffer. As the operator proceeds through the file, EDIT writes lines from the end of the buffer out to disk, maintaining a pre-screen buffer so the user can roll backwards. Rolling backwards is restricted by the size of the pre-screen buffer. Of course, if the file is small enough to fit completely in memory, there is no restriction. Sometimes the processor appears to hang while it is doing buffer management. If on the command line, and a colon is not in the first column, a pseudo-enter anywhere on the command line will fetch another line.

Note: Since so much is held in memory, when editing a very large file for the first time it is recommended that a :E be done periodically to prevent total loss in event of accidental Boot, or power loss.

23.10 EDITOR Command Format

The text appearing on the screen lines (i.e. the lines above the command line) may be edited using a set of 'commands'. A 'pointer' (>) in the left hand column of the screen indicates the line which the command affects.

To move the pointer up, press the keyboard (KBD) key. To move the pointer down, press the display (DSP) key. The pointer wraps around from the top to the bottom and vice versa.

Commands allow the user to delete a single line (:D) or part of the screen (:SC and :SB), insert (:I) a new line between the current lines on the screen and modify (:M) parts of a line by replacing text or inserting new text. Commands are also available to search the file for specific text (:F and :L), for the end of the file (:EO or :E*), or for a particular line by number (:G).

An editor command is always preceeded by a COLON (:). To enter a command, type a colon in the first column of the command line with the appropriate command character(s) and any necessary parameters following. The command characters may be upper or lower case. To force a line to contain a colon in the first column as text, start the line with two colons (the first one is discarded and the line shifted left).

23.11 Basic EDITOR Commands

The following commands are a few basic editor commands (such as FIND, DELETE, and MODIFY) so that the user can get started without worrying about complex command forms. Remember that the 'pointer' on the screen must point to the line affected by the command. Intermediate commands (for experienced users) and advanced commands (for handling repetitive Editing tasks) will be discussed later.

23.12 Setting Tabs

:T - TAB set - this command enables the user to reset the tab stops during execution. The command causes a line of numbers to be displayed across the bottom of the screen.

The operator should space over to each position where a tabstop is desired and type any non-blank character. These tab stops are meaningful during data entry. A maximum of 20 tab stops may be set.

See the section on 'Changing Tabs' for more information on setting tabs.

23.13 Setting TEXT Mode

:X - TEXT - this command enables word wrap-around and disables shift key inversion and space insertion after leading periods, pluses, and asterisks. It automatically enters the tab set command (:T), so that tab stops may be cleared by the operator. The tab key character is not changed; therefore, the ":<tab key>" command must be used to set a new tab key character if one is desired.

See the section on 'Changing Modes and Options' for more information.

23.14 INSERTing a Line

:I - INSERT - Perform a line insert at the pointed line.

This command causes the lines from the bottom of the screen to the pointed line (but NOT including the pointed line) to roll down one line on the screen. A blank line is inserted below the pointed line, the pointer moves down to point to the blank line and the cursor is left at the beginning of the new blank line where data entry may proceed.

If word wrap-around is enabled and the end of the new line is reached during text entry, the new line (and all lines above it on the screen) roll up leaving a blank line containing the overflow from the previously INSERTed line. The cursor is left following the overflow awaiting continued data enty. When the ENTER key is pressed, the cursor returns to the command line and the new text, if any, remains on the screen in its new position.

If the ENTER key is hit in the first column of the blank INSERTed line, the screen rolls up to fill the null line, leaving the screen in its original form.

If the Pseudo-ENTER key (SHIFT/DEL) is used instead of the ENTER key to terminate an INSERTed line, another INSERTion is performed and the cursor remains on the newly INSERTed blank line awaiting data entry of another line.

To make complex changes to a line already on the screen, the operator may INSERT a line immediately below the original and then retype the line with the changes. The original line may then be DELETEd.

23.15 DELETEing a Line

:D - DELETE - delete the entire pointed line.

Blanks are written over the entire pointed line and the cursor is left on the now null line where new text may be entered. When the new line has been entered, it may be terminated with the ENTER key. This leaves a new blank line in the place of the DELETEd line, the pointer at the new line and the cursor returned to the command line.

If no replacement text is needed, pressing the ENTER key again, while the pointer is at the blank line, causes the screen to roll up one line from the bottom to fill the blank line with the line that was originally below it, leaves the pointer in the same vertical position and returns the cursor to the command line. At the end of the file (last screen of data) this action leaves a blank bottom screen line. This line is a "phantom" line with no real existence in the text file. It will not be written to disk.

If a new line is written in the place of the DELETEd line and word wrap-around is enabled, the new line plus the screen lines above the new line roll up, as described in the "INSERTing" a Line section.

If the new line is terminated with a Pseudo-ENTER key, the cursor does not return to the command line. Instead, an INSERTion is made at the pointed line.

For other approaches to deleting lines, see "Deleting Lines" in the "Intermediate Commands" section.

23.16 COPYing a Line

:A - APPEND - copies the pointed line to the bottom of the screen, and rolls the screen up one line. The cursor returns to the command line and the pointer stays with the original pointed line by moving up one position when the screen rolls up. (Use :A* to keep the pointer in the same vertical position on an APPEND).

:C - COPY - deletes the pointed line from the screen, rolls up the screen, copies the pointed line to the bottom of the screen, and deletes the pointed line.

When the COPY command is entered, the pointed line is DELETEd and text may be entered in the now blank pointed line. As with the DELETE and INSERT commands, multiple lines may be inserted in the screen lines when word-wrap is enabled or by terminating lines with the Pseudo-ENTER key.

When the new text entry is terminated with the ENTER key, the cursor returns to the command line. The original pointed line has been written following the line that was on the bottom of the screen when the COPY command was initiated. If one or more new lines has been inserted during the COPY, the user must roll up the screen to view the moved line.

If no new text was entered, i.e., the ENTER key was entered as the first character on the new line, the screen rolls up to fill the null line and the pointer remains in the same vertical position. Since the screen has rolled up, the pointer is now pointing to the line following the first copied line so that a group of lines may be easily copied to another part of the screen.

23.17 MODIFYing a Line

:M [old text][command separator][new text] - MODIFY - a modify command allows the operator to:

1) replace [old text] by [new text] (<),
2) insert [new text] after [old text] (>)
3) or append (i.e., truncate and add) [new text] after [old text] (\).

For instance:

```
:M [old text]<[new text]
```

replaces [old text] on the pointed line with [new text]. The command:

```
:M [old text]>[new text]
```

inserts [new text] immediately following [old text] on the pointed line. The command:

```
:M [old text]\[new text]
```

truncates the pointed line immediately following [old text] and then appends [new text].

If [old text] is not found in the pointed line, the machine beeps and returns to the command line without making any modification to the pointed line. Modifications at the end of the file (on the last screen of data) can create "phantom" lines as described above under "DELETEing a Line".

For the "replace" and "append" forms of the MODIFY command, the [NEW TEXT] may be null, effectively deleting the [OLD TEXT] or truncating the line, respectively. For the "append" form, the [OLD TEXT] may be null, indicating that the line is to be deleted from the beginning. If a field is specified, the field is deleted. See the section on Field Modification for further explanation. For the many various forms of this command see the 'MODIFY Command' section.

23.18 LOCATEing a Line

:L - LOCATE next - typed exactly :L[ENTER], finds the next line of text. If positioned at the end of the file, the 'next' line is the first line of the file.

:L [old text] - LOCATE match - searches for a line containing embedded text matching [old text]. Leading spaces should be supplied if meaningful.

For additional approaches to locating a line, see the 'File Search Commands' section.

23.19 ENDing the EDIT

:E* - EOF without display - searches for the end of the file and, when it is reached, displays the last screen of text. The search may be aborted by pressing the KEYBOARD key.

:E - END - causes the remainder of the logical source file to be copied to the logical scratch file and then, if the logical scratch is not the physical input file, the scratch file is copied back to the source file.

The command line is left on the screen as long as the copy from source to scratch is in progress; it is erased during the final copy from scratch back to source.

The END may be aborted as long as the command line is still displayed, by pressing the keyboard (KBD) and display (DSP) keys simultaneously. When the final copy is completed, control is returned to DOS.

NOTE: If EDIT is exited by any other means than one of the ":E" commands, the format of the scratch file is not guaranteed. Also, if a system error such as "FILE SPACE FULL" is encountered while a ":E" is in progress, the format of the files is not guaranteed.

23.20 Intermediate Commands

Most of the following commands are expansions of the ones in the previous section. One additional concept introduced in this section is that of "fields". A field is a portion of the line between two consecutive tabs. Field one is between the left margin and the first tab, field two is between the first and second tab, etc. Even though up to twenty tabs may be set, only the first nine fields may be referenced.

23.21 Changing Special Characters

:[tab key] - change the tab key character to any non-alpha, non-numeric, non-COLON, non-command special character (/,<,>,etc.), or non-ENTER character typed after a leading colon on the command line.

:[old modify operator] [new modify operator] - change the old modify operator to the new one specified.

:[old continue character] [new continue character] - change the old continue character to be new one specified.

:CH - display the current special characters.

For instance, if a user wants to use "]" for the tab key, "=" for the modify replace operator, "-" for the modify insert operator, and "¦" for the modify append operator, the following commands are typed:

```
:]
:< =
:> -
:\ ¦
```

Then to check that they were properly changed, the user types:

```
:CH
```

which displays:

```
:CH TAB KEY: ] CONTINUE: & MODIFY REPLACE: =  INSERT: -  APPEND:  ¦
```

Note that none of the modify operators nor the continue character may be the same character and the special character must be (like the tab character) non-alpha, non-numeric, non-colon, non-ENTER and not >, <, \, &, or other designated special characters. A beep sounds if a character change command is invalid.

At the end of EDIT, the special characters and tabs are stored in the updated file. The next time that EDIT is used with the file, the same characters are used if not changed by the command line parameters. The tabs and special characters are displayed below the sign-on message.

23.22 Changing Tabs

:T - TAB set - enables the user to reset the tab stops during execution. The command causes a line of numbers to be displayed across the bottom of the screen.

The operator should space over to each position where a tabstop is desired and type any non-blank character. These tab stops are meaningful during data entry and for referencing fields (the portion of the line between consecutive tab stops). A maximum of 20 tab stops may be set.

NOTE: If the cancel key is depressed, it will cause the numbers on the screen to no longer be displayed but the tab set still remains in operation until the enter key is depressed.

:T [nn][,nn]... - TAB set by column number - enables the user to reset the tab stops by column number. For instance, entering ":T 9,15,30" sets the tabs to columns 9, 15, and 30. A maximum of 20 tab stops may be set.

At the end of the EDIT, the tab positions and special characters are stored in the updated file. The next time that the file is EDITed, the same tabs and special characters are used. They are displayed immediately below the sign-on message.

The following command sets tabs to pre-determined default values.

:TD - Set Datashare/Databus tabs at columns 10 and 20.

23.23 Changing Modes and Options

:X - TEXT - enables word wrap-around and disables shift key inversion and space insertion after periods, pluses, and asterisks in horizontal column one. It automatically enters the TAB set command (:T), so that tab stops may be cleared by the operator. The tab key character is not changed; therefore, the ":[tab key]" command must be used to set a new tab key character if one is desired.

:XI - Invert TEXT - enables shift key inversion and disables word wrap-around and enables space insertion after periods, pluses, and asterisks in horizontal column one. It automatically enters the TAB set command so that tab stops may be reset by the operator.

:K - Keyclick - causes the machine to 'click' every time a key is struck.

:KI - Keyclick Invert - turns off the 'click', if set.

23.24 Deleting Lines

The user may delete the leading part of a line, the whole line, or multiple lines.

:D - DELETE - deletes the entire pointed line (See 'DELETing Lines' in the 'BASIC EDITOR COMMANDS' section).

:D [old text] - DELETE through - deletes all characters from the left edge of the pointed line through (and including) the specified [old text]. The remaining characters are left justified and re-displayed. The cursor returns automatically to the command line.

:D[#] [old text] - DELETE through field - deletes all characters from the left edge of the pointed line through (and including) the specified [old text] <u>in the specified field</u>. The remaining characters <u>in that field only</u> are left justified and re-displayed. All characters following the specified field are also deleted. Note: [old text] is required for :D [#]. The cursor returns automatically to the command line.

:SC - SCRATCH above - this command erases the lines from the top of the screen down to the pointed line, inclusive. The cursor and pointer are moved to the top line where data entry may proceed.

:SB - SCRATCH below - this command erases the lines from the pointed line to the bottom of the screen, inclusive. The cursor is left on the pointed line, where data entry may proceed.

As with the DELETE command, additional lines of text may be inserted with word wrap-around, if enabled, or by terminating each line with Pseudo-ENTER instead of ENTER. If no replacement is to be made after :SC or :SB, tap the ENTER key once more to put the cursor on the command line.

23.25 MODIFY Command

The general form of the MODIFY command is:

```
:M[V][#] [old text][modify operator][new text]
```

where [V] is the VERIFY option, [#] is the optional field number, and [modify operator] is the command separator which defines the action of the command. Both [old text] and [new text] fields are optional. If [old text] is omitted, the command takes effect at the left-most edge of the pointed line (or at the left edge of the specified field). If the [new text] field is omitted, a null, or zero-length field is used to execute the modification. Note that the [old text] cannot include any of the modify operator characters; [new text] may contain one of these characters. If necessary, the modify operators can be changed as described above in 'Changing Special Characters' to avoid bad interpretations of modify commands.

The VERIFY option causes the cursor to blink at the first character to be modified. The user then has three responses. If he presses 'Y' for 'YES', the modification takes place. If he presses the ENTER key, control returns to the command line. If he presses any other key, the modify command continues to search the line for another occurrence of [old text] to modify.

CAUTION: In all MODIFY commands, if the pointed line becomes longer than 79 characters, right truncation will occur, unless word wraparound is in effect.

23.25.1 Line Modification

The following descriptions are of the line modification version of the MODIFY command.

:M[V] [old text][replace operator][new text] - MODIFY (replace) - replace the specified [old text] by the specified [new text]. The "less than" character (<) is the default command separator which indicates replacement. If [new text] field is omitted, the [old text] is simply deleted and the line compressed to the left.

For example to modify the text line:

THE QUICK BROWN FOX JUMPED OVER THE LAZY DOG'S BACK.

The command: ":M BROWN<RED" causes the line to be redisplayed like this:

THE QUICK RED FOX JUMPED OVER THE LAZY DOG'S BACK.

The command: ":M .< 1234 TIMES." to the above line generates a line like:

THE QUICK RED FOX JUMPED OVER THE LAZY DOG'S BACK 1234 TIMES.

If the replacement causes the line to become longer than 79 characters and word wrap-around is enabled, the trailing word is wrapped around and a new line is inserted containing the entire last word. If the [new text] is shorter than the [old text] it replaces, the line is shortened.

After the pointed line is redisplayed, the cursor is returned to the command line.

:M[V] [old text][insert operator][new text] - MODIFY (insert) - the command separator "greater than" (>) is the default character causing the [new text] to be inserted in the pointed line immediately after the [old text].

:M[V] [old text][append operator][new text] - MODIFY (append) - the "backslash" (\) is the default command separator causing everything in the pointed line past the [old text] to be replaced by the [new text].

:M or :M[#] - MODIFY repeat - typed exactly :M[ENTER] or :M[#][ENTER], uses the [old text][sep][new text] from the last MODIFY command. This is useful when making the same change repeatedly. Note that the field number is not saved, and must therefore be supplied if necessary.

:MV or :MV[#] - MODIFY VERIFY repeat - typed exactly :MV[ENTER] or :MV[#][ENTER] is the same as the above command except that it invokes verification.

:M* - MODIFY display - display the expression entered for the last MODIFY. After the saved command is displayed, the cursor is turned off and the operator must press ENTER to proceed. No MODIFY is actually performed.

Note: Trailing spaces on the modified line are always deleted. This action may cause a multi-line record to be shortened.

23.25.2 Field Modification

In field modification mode, the MODIFY command acts only on a specific field and does not expand or contract the entire line but maintains the integrity of all fields before and after the affected field.

:M[V][#] [old text][modify operator][new text] - MODIFY field - where the pound sign [#] is a number from 1 to 9 designating the TAB field to be modified (or the starting point to search for matching [old text]). This command may be executed in any of the previous Modify forms.

Modification is performed within the specified field only. Thus, a replacement or append shorter than the original field is blank filled and subsequent fields will maintain their position and content. An insertion longer than the specified field is truncated (with the exception of the last field whenever word wrap-around is in effect).

NOTE: Remember when using the repeat form of the MODIFY command that the modification applies to the entire line if the field number is omitted.

23.26 Line Splitting

:V [old text] - SPLIT - split the pointed line after the text matching [old text] and insert the remainder of the line past the matching text below the pointed line. The pointer remains in its original position. This is useful for INSERTing sentences in the midst of text without doing a group of cumbersome MODIFYs.

23.27 Line Concatenation

:W - CONCATENATE - append the line below the pointed line to the pointed line. The pointed line is assumed to have a trailing space if word wrap-around is in effect. This is useful in text mode where a MODIFY has caused words to wrap-around to the next line and the operator wishes to include them with the following line.

23.28 File Search Commands

Manual operator-controlled searches may be performed by depressing the keyboard (KBD) and display (DSP) keys simultaneously to cause data to be fetched from the file (forward or backward depending which key is pressed first) and displayed on the screen. This continues until either key is released. The :EO command performs the same function automatically, i.e., it causes lines to be fetched and displayed until the end of the file is reached. Pressing the DISPLAY key causes the :EO command to pause until it is released. To terminate an :EO command, press the keyboard (KBD) key. To fetch a single line, use the Pseudo-ENTER key (shift DEL) .

The :E- command works the same way as the :EO except that it fetches lines backwards through the file in memory rolling the screen down.

To find the end of a file without displaying the entire file (since the display is time consuming) use the :E* command. This searches for the end of file and displays the last screen of data.

For more information on the :E commands, see the section on 'Terminating the EDIT' .

FINDs and LOCATEs search the file for a line containing specific text. When a line has been found, it is aligned with the pointer on the screen so that it may be operated on. Lines above and below the line are also displayed. FINDs and LOCATEs allow field specification, that is, if a field is specified, only lines with the specific text in that field are found.

A FIND or LOCATE wraps entirely around the file. If the requested text is not found, the last screen image when the FIND or LOCATE was executed is displayed and the machine beeps. A FIND or LOCATE may be aborted by pressing the keyboard (KBD) key.

The [old text] specified for a FIND command is saved. The saved [old text] may be redisplayed or used again. FIND, LOCATE, DELETE, and QUERY store the [old text] in the same save area.

:F [old text] - FIND match - the input file is searched for a line starting with the specified [old text]. Leading spaces in the file's lines will be ignored and should not be entered as part of [old text]. Note that this command should be typed exactly :F[SPACE][old text]. The :L command should be used if leading spaces are important.

:F[SPACE] - FIND same match - if the FIND command is followed by exactly one space and the ENTER key, the previous FIND (DELETE, QUERY, or LOCATE) [old text] is used for this FIND. Several occurrences of the same text may be searched out in this manner.

:F[#] - FIND in field [#] - search for a field starting with the desired text. Field FINDs may be used in either of the command formats: :F[#] [old text] or :F[#][SPACE].

:F* - FIND display - the asterisk (*) after the FIND command causes the [old text] of the previous FIND, DELETE, QUERY, or LOCATE command to be displayed. The cursor is turned off and the operator must press ENTER to proceed. No FIND is performed.

:L - LOCATE next - typed exactly :L[ENTER], finds the next line of text. If positioned at the end of the file, the screen is cleared and the 'next' line brought up is the first line of the file, ie. file wraparound.

:L [old text] - LOCATE match - similar to FIND match except that the locate command searches for a line containing imbedded text matching [old text]. Leading spaces should be supplied if meaningful.

:L[space] - LOCATE same match - typed exactly :L[SPACE][ENTER], uses the [old text] specified by the previous LOCATE, DELETE, QUERY, or FIND command to perform a search.

:L[#] - LOCATE in field [#] - locate desired string in specified field. Field LOCATES may be used in either of the above command formats: :L[#] [old text] or :L[#][SPACE].

:L* - LOCATE display - display the [old text] entered for the previous LOCATE, DELETE, QUERY, or FIND command. As in the FIND display, the cursor is turned off and the operator must press ENTER to continue. No LOCATE is actually performed.

There are several variations on the GET command that allow the user to quickly jump forward or backward through a file or to a specific line by number.

:G - GET next screen - clears the screen and refills it with the next screen image.

:G- - GET prior screen - clears the screen and refills it with the prior screen image.

:Gnnnnn - GET nnnnn lines - fetches nnnnn (from 1 to 65,535) lines from the file or until the end of file. For example, ":G1" rolls the screen up one line, and displays the next line.

:G-nnnnn - GET nnnnn lines backward - same as the above command except that it fetches backwards through the file in memory, and rolls the screen down.

:GAnnnnn - GET Absolute [nnnnn]th line - position the file so that the [nnnnn]th line of the file (counting from the beginning of the file) is displayed on the bottom line of the screen.

23.29 BYPASS End of File

:B - BYPASS - fetch a line from the file, bypassing the end of file. This may be a true end of file or one caused by RECORD FORMAT errors, PARITY errors, or a RANGE TRAP (see Recovery Procedures). Subsequent lines may then be fetched by the normal mechanisms. This command is intended as a recovery tool for use only if the file has been accidentally shortened or contains badly formatted records.

NOTE: There are certain file conditions which :B cannot handle and/or correct.

23.30 Terminating the EDIT

:E* - EOF without display - searches for the end of the file and, when it is reached, displays the last screen of text. The search may be aborted by pressing the keyboard (KBD) key.

:EO - EOF with display - this command causes the data to be fetched and displayed on the screen continuously until end of file is reached. The search may be temporarily stopped by pressing the display (DSP) key or terminated by pressing the keyboard (KBD) key.

:E- - Display to the beginning of file - works exactly as the :EO command above except that it fetches backward through the file in memory, rolling the screen down.

:E - END - the end command causes the remainder of the logical source file to be copied to the logical scratch file and then, if the logical scratch is not the physical input file, the scratch file is copied back to the source file.

The command line is left on the screen as long as the copy from source to scratch is in progress; it is erased during the final copy from scratch back to source.

The END may be aborted as long as the command line is still displayed, by pressing the keyboard (KBD) and display (DSP) keys simultaneously. When the final copy is completed, control is returned to DOS.

:E/ - END/DEL - same as the END command, except the edited file is truncated below the bottom line on the screen.

More specifically, this command causes the remainder of the source file to be deleted (the lines currently on the screen are written out), and, if the logical scratch file is not the physical source file, the scratch file is copied back to the source file. When the file is completely updated, the system is reloaded.

:E\ - END/DEL - same as the END command, except the edited file is truncated above the top line on the screen.

More specifically, this command causes the prior portion of the source file to be deleted (the lines currently on the screen are written out), and the remainder of the file to be written out to the scratch file. If the logical scratch file is not the physical source file, the scratch file is copied back to the source file. When the file is completely updated, the system is reloaded.

:O - this command causes the EDITOR to return to DOS without updating the source file.

:OX [DOS command string] - this command causes EDIT to return to DOS without updating the source file and then execute the DOS command string.

NOTE: If EDIT is exited using either :O or :OX the format of neither SCRATCH/TXT nor SCRATCH/XTX is guaranteed correct. Also, if a file is created and EDIT is exited using :O or :OX rather than :E etc., then the format of the newly created file is not guaranteed.

23.31 Advanced Commands

:0 [user-defined command string] - Define the user-defined command string zero.

:0 - Execute user-defined command string zero.

The user may define and execute up to ten EDIT command strings. These strings may use themselves, do conditional skips of commands, and request operator response. This gives the EDIT the capability of doing sophisticated file modification in a semi-automatic manner (the user always has complete control).

The user may define commands ":0" through ":9" as one or more EDIT commands. The defined command may be a single command (e.g. a complicated MODIFY command) that is used quite often, but one the user doesn't want to type in every time he needs it. Or it may be a string of commands separated with Pseudo-ENTER (shift DEL) characters. The Pseudo-ENTER character appears as a solid triangle on the screen. The command string should be terminated with a Pseudo-ENTER (shift DEL) if trailing spaces are significant. A carat is used in place of the solid triangle in the examples below. For instance, if the user types:

```
:2 :M abcdefghijklmnop<ponmlkjihgfedcba
```

he has defined command 2. Every time that he types in:

```
:2
```

the pointed line is modified in the specified manner. If the user types:

```
     :5 :M abcde<edcba^:M fghij<jihgf
```

he has defined command 5 as a pair of modify commands. Every time that he types in:

```
     :5
```

the pointed line has both modifications done in sequence just as if the user has typed them in separately. If the first modification fails, the second one is not performed.

If the user needs to replace every occurrence of the string "LABEL" in his file with "LABEL1", he may define a command as:

```
     :2 :L LABEL ^M LABEL <LABEL1^2
```

Note: a new definition discards the old definition. Also, the colon following the Pseudo-ENTER character is optional. The user may then type:

```
     :2
```

which loops changing "LABEL " to "LABEL1" everywhere it is found in the file. The command string terminates when the LOCATE fails. Of course the user can always terminate the command with the keyboard (KBD) key.

For a more complicated example, the user may be EDITing the disk file which may be created by the DOS FILES Command, and define two commands as:

```
     :8 :M2 \:DR0,:DR1^:M <COPY^:9
     :9 :Z^:M  /</^:9^:G1^:Z^:Q /^:8^:Z*
```

Then he may set a tab at column 13 (immediately after the file extension in the FILES-created file). By typing:

```
     :8
```

the user creates a CHAIN file for copying all the files from drive zero to drive one.

Modify commands used on the last screen of lines from the file generate a "phantom" line below the present screen lines. Creation of this line means that a :G1 command will always be successful following a modify. For this reason, repetitive commands should be constructed to use a locate or query command that will fail when the instruction should terminate. The examples above illustrate use of this type of test to terminate a repetive command. The phantom lines that can appear at the end of a file during EDIT are null lines and will not appear in the updated disk file.

The execution of any command string may be temporarily stopped by holding down the display (DSP) key or terminated by pressing the keyboard (KBD) key.

:0* - Display the user-defined command string zero on the bottom of the screen

:00 - Display the user-defined command strings zero through nine on the top of the screen.

The above commands allow the user to examine command strings individually in relation to the text on the screen or to examine them all at once in relation to each other. Press ENTER to terminate the above displays.

:0< - Insert user-defined command string zero into the file text immediately below the pointed line.

:0> - Define user-defined command string zero from the text in the pointed line on the screen.

The above commands allow the user to save the command strings in the text of his file. It also simplifies the modification of command strings as the user can use MODIFY on the string rather than keying in the entire string again. It also assists in defining several similar command strings.

A definition file with a default name of EDIT/DEF may be created by the user to contain a set of user pre-defined command strings. These are loaded automatically every time that EDIT is executed. This is an EDITable file. Remember, the user can force a colon as the first character on a line by starting the line with a double colon. The definition file may contain comment lines (e.g. lines starting with "+", "*", or ".") or null lines. The sequence of the defined command strings has no effect. Command strings may be multiple defined; the last definition will be the one in effect.

:99 [user-defined command string] - Define an initialization command string in the definition file.

If an initialization command is defined in the definition file, it is executed automatically when EDIT is executed. It may be defined to do things such as change the tab key, turn on key click, change the modify operators, set tab stops, or even do file modification without operator intervention. The automatic execution of the initialization command may be overridden by pressing the keyboard (KBD) key when executing EDIT.
NOTE: These changes will not change the screen display of parameters.

:Z* - Terminate execution of the user-defined command string and return control to the bottom keyin line.

:Z - Skip over one command after the following command in the user-defined command string if the following command fails.

:Z[n] - Skip over [n] commands (1 to 9) after the following command in the user-defined command string if the following command fails.

Almost every EDITOR command either "succeeds" or "fails". For instance, LOCATE succeeds if it finds a line containing the specified text and fails if it doesn't. The MODIFY command fails if the string to be modified is not found or if, using the VERIFY option, the user has pressed the ENTER key. The GET fails if the end of file is reached. The use of the above commands allows conditionally skipping over commands in the user-defined command depending on the success of a command.

:U[n] - Unconditionally skip over [n] commands (1 to 9) in the user-defined command string.

This allows skipping over commands that might be jumped to by a conditional skip.

:Q [string] - QUERY, setting a succeed or fail condition, if the specified string is contained in the pointed line.

This command works similar to the FIND or LOCATE command in that it uses the line save area and allows field specification. It does not affect the pointed line at all but sets up a conditional skip. For instance, if command 3 were defined:

```
:3 :Z^Q2 ABC^Z*^G1^3
```

then its execution would GET lines until the pointer pointed to a line containing the string "ABC" in field 2. While the QUERY fails, the first conditional skip command (":Z") causes execution of the command string to skip over the ":Z*", GET a line, and then start over. When the QUERY succeeds, the ":Z*" is executed which terminates the command string. This example effectively does a LOCATE while displaying all the lines examined.

See the Appendix 'Example of an EDIT/DEF File' for more user-defined commands.

23.32 Recovery Procedures

A 'FORMAT TRAP' occurs when a record not belonging to the current file is encountered. This can be caused either by a physical misalignment of the disk read head or because a record has erroneously been written into that file by some other program.

A 'RANGE TRAP' occurs when the physical limit of the file is reached and no end of file is present.

A 'PARITY TRAP' occurs when a record is misread from the disk. This may be caused by physical misalignment of the disk read head or a bad surface on the disk.

These three errors cause the EDITOR to believe that it has reached the end of file. To read past an end of file, use the BYPASS command, ":B", repeatedly if necessary.

File Recovery

If the source file is lost (e.g., erroneously KILLed), one of the scratch files may contain a useful copy. Since the scratch files (SCRATCH/TXT or SCRATCH/XTX) usually contain a copy of the last file edited, they may be used to recover only that file.

Note: If the file fits completely within the memory buffer, scratch files are never used.

Also Note: There are some file conditions that EDIT cannot correct!

23.33 Glossary of EDIT Terms

Assembler Mode - assumed if 'A' appears in the parameter list. Tab stops are set at 9, 15, and 30 (may be changed during execution). The SPACE BAR is assumed to be the tab key character (maybe changed in parameter list or during execution). Shift key inversion and word wraparound are assumed. A leading plus (+), period (.), or asterisk (*), generate a following space for comment lines.

Command - characters typed at the left edge of the command line following a COLON (:) which have special meaning to the editor.

Command line - the bottom line of the screen where most data is entered, lines are fetched and commands are typed.

Configuration sector - the first sector of a file which has been written by EDIT. It is invisible to most other programs. When the configuration sector exists, it contains the tabs, bell margin, special characters, click option, mode and space-compression information and is used for the editor defaults.

Continue Character - a character which when entered in the first column causes a continue indicator to appear (see below). The default continue character is the ampersand (&) which may be changed during execution.

Continue Indicator - A solid triangle appearing in the first column which means that the previous record exceeds 79 characters. When the continue character has been entered in the first column, the continue indicator appears and its presence means the line containing the indicator is joined on output to the previous line, possibly creating records greater than 79 characters.

DATABUS mode - assumed if a 'D' appears in the parameter list. Tab stops are set at 10 and 20 (may be changed during execution). The SPACE BAR is assumed to be the tab key character (this may be changed in the parameter list or during execution). Shift key inversion and no word wrap-around are assumed. Leading periods (.), pluses (+), and asterisks (*) generate a following space (.) or (+) or (*) for comment lines.

Definition file - this is an EDITable file, containing pre-defined user command strings, which is automatically loaded when the Editor is executed. The definition file may also contain an initialization command (":99") which is automatically executed unless the keyboard (KBD) key is pressed. The default name for the file is EDIT/DEF.

Field number - a digit used in commands to designate the portion of the pointed line between two consecutive tab stops. Field '1' is always from column 1 to the first tabstop; thus, in Assembler mode, '1' designates the label field, '2' the opcode field, '3' the expression field and '4' the comment field. During field modification, leading and trailing fields are preserved.

Line insert - results from an INSERT command, data entry or modification when word wrap-around is in effect, or use of the Pseudo-ENTER key. The lines below the pointed line are rolled down and a new, blank line is generated at the pointed line.

Logical scratch file - current output file.

Logical source file - current input file.

Modify operator - the character in a MODIFY command which indicates what is to be done. The default replace operator is the "less than" symbol (<), the default insert operator is the "greater than" symbol (>), and the default append operator is the "backslash" (\).

Multi-line Record - a line on disk that is greater than 79 characters, displaying as more than one line on the screen with each continued line marked with a continue indicator.

New text - a group of characters, typed immediately after a modify operator in a modify command, which will become part of the line being modified.

Old text - a group of characters, including spaces, which are searched for, either in the pointed line (as in the MODIFY command) or in the file (as in the FIND or LOCATE commands).

One-pass option - assumed if an 'O' appears in the parameter list. The one-pass option does not update the physical source file.

Parameter list - initialization information provided when the editor is first executed. Following file specifications, a SEMI-COLON (;) indicates the presence of a parameter list. The mode (A, D, or T), one-pass option (O), tab character, margin bell column, key-click (K), space-compression (E or G), non-verification (Y) and (in text mode - T), 'shift inversion' (S), and 'no word wrap-around' (L) may be set.

Pointed line - a pointer (>) in the left hand margin is used to reference lines for modification by command. The line to the right of the pointer is the pointed line.

Physical scratch file - specified (or implied SCRATCH/TXT) output file.

Physical source file - specified input file.

Pseudo-ENTER - the key marked DEL (always shifted) is referred to as the Pseudo-ENTER key. If pressed in the first column of the command line, one line of text is fetched from the source file.

If pressed while entering a command, a user-defined command string separator is entered.

In all other modes, the Pseudo-ENTER key causes a new line to be inserted so that data entry may proceed in the same area of the screen. If pressed on the last screen line, it causes the processor to beep.

Scratch file - at any point in time, the logical scratch file is the output file.

Screen line - any of the lines on the screen which may be referenced by the command pointer. The command line (bottom line) is not, therefore, included.

Shift key inversion - reverse the function of the shift key for all alpha characters so that, in lower case, alpha characters appear upper case.

Source file - originally this is the input file specified at initial execution. The term source file refers to the current input file; thus, at any point in time, the logical source file may be either the specified input file or one of the scratch files.

Space-compression - an "abbreviation" written to a disk file when two or more blank spaces are consecutive in a string. For example, "HI   THERE" is converted to "HI(011)(006)THERE" where (011) is the space compression indicator, (006) is the number of spaces compressed and integers in parenthesis are octal codes. The default mode is 'G' for space-compressed mode. The 'E' option on the command line causes spaces to be expanded, i.e., written out with no space-compression.

Text mode - assumed by a 'T' in the parameter list. No tab stops are set (tabs may be set during execution). The SEMI-COLON (;) is the assumed tab character (the tab key character may be changed in the parameter list or during execution). No shift key inversion is performed (this may be selected in the parameter list with an 'S'). Word wrap-around is performed (this feature may be turned off by an 'L' in the parameter list).

Word - a word is defined as any group of less than 70 characters preceeded by a space.

Word wrap-around - a feature of text mode. During data entry a space to the right of the margin bell column causes an immediate carriage return. If this occurs on a screen line, a line insert is performed so that data entry may proceed at the same area of the screen. If a character is typed over the last column of the screen, the last word is removed, a line insert performed and the removed word is placed at the beginning of the inserted line where data entry may proceed. If a modify command causes the line to become longer than 79 characters, the trailing characters, including the last word on the line, are moved to a new line which is inserted below the original line. Control then returns to the command line.

23.34 Command List

The full set of EDITOR commands are listed below. All legal combinations of options are included. [SP] represents a space. [ENT] represents the ENTER key. [#] represents the field number (in the range 1-9). Commands may be either upper or lower case.

:[NEW TAB KEY][ENT] - Replace the old tab key character with the new one. The default for the tab key character is the space bar.

:[OLD CONTINUE CHARACTER][SP][NEW CONTINUE CHARACTER][ENT] - Replace the old continue character (for input of long records) with the new one. The default for the continue character is the ampersand (&).

:[OLD REPLACE OPERATOR][SP][NEW REPLACE OPERATOR][ENT] - Replace the old modify-replace character with the new one. The default for the modify-replace character is the "less than" (<) symbol.

:[OLD INSERT OPERATOR][SP][NEW INSERT OPERATOR][ENT] - Replace the old modify-insert character with the new one. The default for the modify-insert character is the "greater than" (>) symbol.

:[OLD APPEND OPERATOR][SP][NEW APPEND OPERATOR][ENT] - Replace the old modify-append character with the new one. The default for the modify-append character is the "backslash" (\).

::[string][ENT] - Force a line beginning with a colon to be entered into the text of the file.

:A[ENT] - Copy the pointed line to the bottom of the screen and roll the screen up one line and move the pointer up one line.

:A*[ENT] - Copy the pointed line to the bottom of the screen and roll the screen up one line without moving the pointer. This allows defining a command that does multiple ":A*"s to copy multiple lines.

:B[ENTER] - BYPASS the end of file which may be caused by RECORD FORMAT, PARITY, or RANGE errors.

:C[ENT] - Copy the pointed line to the bottom of the screen, delete the pointed line, and key in a new line there (if desired).

:CH[ENT] - Display the current tab key character and MODIFY operators.

:D*[ENT] - Display the previously defined string used by DELETE, FIND, LOCATE, or QUERY.

:D[ENT] - Delete the pointed line and key in a new line there (if desired).

:D[SP][ENT] - Delete up through a previously defined string (defined by a previous DELETE, FIND, LOCATE, or QUERY) in the pointed line.

:D[SP][string][ENT] - Delete up through the specified string in the pointed line.

:D[#][SP][ENT] - Delete up through a previously defined string (defined by a previous DELETE, FIND, LOCATE, or QUERY) in the specified field of the pointed line, and all characters in trailing fields.

:D[#][SP][string][ENT] - Delete up through the specified string in the specified field in the pointed line, and all characters in trailing fields.

:E[ENT] - END the EDIT by copying the modified file back to the original source file.

:E/[ENT] - END the EDIT by copying the modified file back to the original source file up to the bottom line displayed on the screen. This deletes the trailing portion of the file.

:E\[ENT] - END the EDIT by copying the modified file back to the original source file starting with the top line of the screen up to the end of the file. This deletes the preceding portion of the file.

:EO[ENT] - Display the file forwards (rolling the screen up) to the end of file. The display (DSP) key stops the display until it is released. The keyboard (KBD) key terminates the display and returns to the command line.

:E-[ENT] - Display the file backwards (rolling the screen down) to the beginning of the file contained in memory. The display (DSP) key stops the display until it is released. The keyboard (KBD) key terminates the display and returns to the command line.

:E*[ENT] - Display the last line of the file at the bottom of the screen (immediately above the COMMAND LINE).

:EX[sp][DOS Command and string][ENT] - End the EDIT by copying the modified file back to the original source file, then execute the DOS command string.

:F*[ENT] - Display the previously defined string used by FIND, DELETE, LOCATE, or QUERY.

:F[SP][ENT] - Find a line starting with the previously defined string.

:F[SP][string][ENT] - Find a line starting with the defined string.

:F[#][SP][ENT] - Find a line starting in the specified field with the previously defined string.

:F[#][SP][string][ENT] - Find a line starting in the specified field with the defined string.

:G[ENT] - Display the next screen-full of lines.

:G-[ENT] - Display the prior screen-full of lines.

:G[nnnnn][ENT] - Roll up the screen and display [nnnnn] lines where the number may range from 1 to 65,535 lines. This stops at the end of file.

:G-[nnnnn][ENT] - Roll down the screen and display [nnnnn] lines. This stops if the beginning of the memory buffer is reached.

:GA[nnnnn] - Display the [nnnnn]th line of the file. This rolls forward or backward through the file depending on the current location in the file.

:I[ENT] - Insert by keying in a new line immediately below the pointed line.

:I[SP][string][ENT] - Insert the specified string immediately below the pointed line.

:K[ENT] - Turn on the key click.

:KI[ENT] - Turn off the key click (Key click Invert).

:L*[ENT] - Display the previously defined string used by LOCATE, DELETE, FIND, or QUERY.

:L[ENT] - Roll up the screen one line. At the end of the file, this causes the file to wrap-around to the beginning.

:L[SP][ENT] - LOCATE a line containing the previously defined string.

:L[SP][string][ENT] - LOCATE a line containing the defined string.

:L[#][SP][ENT] - LOCATE a line containing the previously defined string in the specified field.

:M*[ENT] - Display the previous MODIFY command line.

:M[ENT] - Modify the pointed line using the previous MODIFY command line.

:MV[ENT] - MODIFY with VERIFY the pointed line using the previous MODIFY command line.

:MV[#][ENT] - Repeat the previous MODIFY command line with VERIFY applied to the specified field.

:M[SP][modify string][ENT] - MODIFY the pointed line as specified by the modify string.

:MV[SP][modify string][ENT] - MODIFY with VERIFY the pointed line as specified by the modify string.

:M[#][SP][modify string][ENT] - MODIFY the specified field in the pointed line as specified by the modify string.

:MV[#][SP][modify string][ENT] - MODIFY with VERIFY the specified field in the pointed line as specified by the modify string.

Modify strings are of the following formats. The default replace operator is the "less than" symbol (<). The default insert operator is the "greater than" symbol (>). The default append operator is the "backslash" symbol (\). [string1] and [string2] are optional.

[string1][replace operator][string2] - Replace string1 with string2.

[string1][insert operator][string2] - Insert string2 immediately following string1.

[string1][append operator][string2] - Truncate the line immediately following string1 and append string2.

:O[ENT] - Return to the Operating System without updating the original source file.

:OX[sp][DOS command string] - Return to operating system without updating the original source file and execute the given DOS command string.

:Q*[ENT] - Display the previously defined string used by QUERY, DELETE, FIND, or LOCATE.

:Q[SP][ENT] - QUERY (setting a succeed or fail condition) if the previously defined string is contained within the pointed line.

:Q[SP][string][ENT] - QUERY if the given string is contained within the pointed line.

:Q[#][SP][ENT] - QUERY if the previously defined string is contained within the specified field of the pointed line.

:Q[#][SP][string][ENT] - QUERY if the given string is contained within the specified field of the pointed line.

:SB[ENT] - SCRATCH BELOW deletes all the lines on the screen from the pointed line down through the bottom line and then allows keying in a new line at pointed line.

:SC[ENT] - SCRATCH deletes all the lines on the screen from the top line down through the pointed line and then allows keying in a new line at the top screen line.

:T[ENT] - Set TAB stops by displaying a line of column numbers and allowing the user to space across setting tabs by typing non-blanks. Up to 20 tabs may be set.

:T[SP][nn][,nn]...[ENT] - Set tab stops to the columns specified by [nn][,nn]... where [nn] ranges from 1 to 79. Up to 20 tabs may be set.

:TD[ENT] - Set TAB stops for Databus or Datashare (columns 10 and 20).

:U[ENT] - UNCONDITIONAL skip over the following command in the user-defined command string.

:U[n][ENT] - UNCONDITIONAL skip over [n] (1 to 9) following commands in the user-defined command string.

:V[SP][string][ENT] - Split the pointed line after the text in the line matching [string] and insert the remainder of the line immediately below the pointed line.

:W[ENT] - Concatenate the line below the pointed line to the pointed line. If word wrap-around is in effect, the pointed line is assumed to have a trailing space.

:X[ENT] - Change to TEXT mode with word wrap-around and no shift inversion and then set tab stops (as in :T above).

:Z[ENT] - Skip over 1 command after the following command in the user-defined command string if the following command fails.

:Z[n][ENT] - Skip over [n] (1 to 9) commands after the following command in the user defined command string if the following command fails.

:Z*[ENT] - Terminate execution of the user-defined command string and return control to the bottom keyin line.

NOTE: The following commands, though refering to zero, actually refer to all the user-definable commands zero through nine.

:0[ENT] - Execute user-defined command zero.

:0*[ENT] - Display user-defined command zero.

:0[SP][user-defined command string][ENT] - Define user-defined command zero. The command string consists of one or more EDIT commands separated with a Pseudo-ENTER (shift DEL) character. The colon immediately following the Pseudo-ENTER character is optional.

:0<[ENT] - Insert user-defined command zero into the file text immediately below the pointed line.

:0>[ENT] - Define user-defined command zero as the pointed line on the screen. The combination of this and the above command allow user-defined commands to be saved in the text of the file and to be EDITed.

:00[ENT] - Display the user-defined commands zero through nine.

:99[SP][user-defined command string][ENT] - This is the initialization command which is executed when the EDIT is started. It may be overridden by pressing the keyboard (KBD) key while bringing up EDIT. This command may only be used in the definition file (/DEF) and not during the actual EDIT.

23.35 EDIT ERROR MESSAGES

The following is a list of error messages that may occur during EDIT:

NAME REQUIRED - supply either the name of a new file or the name of the file that is to be edited.

SYSTEM FILES MAY NOT BE EDITED! - Files whose Physical File Numbers are 0 through 7 are designated as system files and cannot be edited without disastrous results.

SCRATCH FILE MAY NOT BE A SYSTEM FILE! - AGAIN This would be disastrous as mentioned above.

POSSIBLE OBJECT FILE! CONTINUE EDIT (Y/N)? - Occurs if a Binary 0 is found as the first byte in Logical Record Number 0 and it is not part of an End of File mark.

BAD DEVICE - the drive specification on one of the three file specifications is either invalid or refers to a drive not on-line.

BOTH SOURCE AND SCRATCH FILES CANNOT BE SAME - either the same file specification has been entered in the first and second positions on the command line or the first file specification is "SCRATCH" and no scratch file specification is given (hence using the default name of SCRATCH) .

BAD EXTENSION (XTX) FOR SCRATCH - since EDIT uses two scratch files where the second scratch file has the name of the first scratch file and the extension XTX, the extension XTX is not allowed in the scratch file specification.

BAD OPTION PARAMETER - an invalid character appears on the command line.

INPUT FILE MUST EXIST IN "ONE-PASS" - the "O" option may not be used during creation of a new file

FAULTY DEFINITION FILE - either the file in the third file specification is not a valid definition file, or EDIT/DEF (the default definition file) does not contain the user-defined commands in the proper form.

TRAPS - FORMAT, RANGE and PARITY traps may occur when there is some problem with the head alignment of the disk drive or when there is something wrong with the file (see the chapter "Recovery Procedures")

Processor "Beeps" - the usual indication of an error in the command just keyed in: rolling off the beginning of the text in memory, rolling off the end of the file, locating text not in the file, or keying in an unrecognizable command (see the "failure" conditions in the chapter "Advanced Commands").

23.36 Configuration Sector

A file that has been written with EDIT contains configuration information in the first sector. This sector begins with an octal 003 so that it is invisible to most programs. As a convenience, the tab settings, special characters and modes are contained in the sector so that these defaults are used the next time the file is edited. If the file has been SAPPed or REFORMATted, for example, the configuration sector is not preserved.

The following describes the contents of the configuration sector. A three-digit integer in parentheses represents an octal byte.

```
(003) * 0 0 0 0 0 0 0 0 (015) 1500 General Editor <version> (015)

<tab numbers> (015) <bell position> (015)
```

```
<tab key> <continue character> <modify replace operator>

<modify insert operator> <modify append operator>

<key click switch> <shift inversion switch> <word wrap-around switch>

<text mode switch> <expand mode switch>

<scratch filename> <definition filename> (015) (003)
```

where:

<tab numbers>

    are decimal integers separated by spaces representing the tab positions and the field lengths. For example, tabs (9,15,30) are represented by "1 8 9 6 15 15 30 50 ".

<bell position>

    is a decimal number of at most two digits.

<tab key>
<continue character>
<modify replace operator>
<modify insert operator>
<modify append operator>

    are the actual ASCII characters.

<key click switch>

    is "N" if off and "K" if the click option is on.

<shift inversion switch>

is "N" if no shift inversion and " " if shift inversion is on.

<word wrap-around switch>

is "W" for word-wrap and "N" if not.

<text mode switch>

is "T" if text mode or "N" if not.

<expand mode switch>

is "E" when space compression is inhibited and "N" if space compression is desired.

<scratch filename>

is the eleven (11) character name and extension of the scratch file.

<definition filename>

is the eleven (11) character name and extension of the definition file.

23.37 Example of a Definition File

```
   . This is an example of a definition file for EDIT.  Note that comments
.  may appear.  Although this contains some useful general-purpose
.  user-defined commands, the most common use of command strings is for a
.  special editing job.  Care should be used in defining user command
.  strings to make sure they perform as intended!  For this illustration,
   the caret (^) is used to represent the SHIFT/DEL triangle.
.
.
.  For example, a file needs "?" following every line beginning with a "*"
.
:0 :t 2^:z1^:q1 *^:I ?^:g1^:0^
```

Many times user-defined commands are required for a one-time application.

Here are some more general examples:

Global search and replace with verification:

```
:1 :z^:mv^:1^:l ^:1^
```

This function assumes that the modification and the locate have been done previously so the repeated form of the command is used. This string can be modified to add field parameters for the locate and modify commands or the verfication option removed from the modify command.

Delete until:

```
:2 :z^:q^:z*^:m \^:2^
```

By using the :Q function, "query" a particular string for the line the cursor is pointing to. This sets up a string for the repeated form of the :Q command. This function deletes all lines beginning with the pointed line up until it finds the particular string entered at first.

Insert line of asterisks:

```
:3 :i *^:m *>**********^:m^:m^:m^:m^:m^:m^:m^:m^
```

Use this function to insert a line of asterisk (or modify the string to insert lines of periods or underscores).

Draw a box of asterisks:

```
:4 :i *^:m *>**********^:m^:m^:m^:m^:m^:m^:m^:m^:t 79^:g-1^:5^

:5 :z^:q ***^:6^:m2 \*^:g-1^:5^

:6 :g1^:z^:q ***^:z*^:m1 <*  ^:6^
```

```
.
.  Use the line function (:3) to draw the top of the box.  Do a :xi to insure
.  that "format" (no word wrap) mode is in effect.  Now go down to where the
.  bottom of the  box is to be drawn and key in :4.  This draws a box of
.  asterisks.
.
.
.
.
.
.
.  Marking updated lines within Databus mode:
.
:7 :t 70^:m2 \ 1.1.D^:ts^
.
.  This is an easy way to mark modified lines.
.
.
.
. Leave :8 empty for "local" user-defined commands.
.
.
```

CHAPTER 24. ENCODE/DECODE COMMANDS

24.1 Purpose

The ENCODE command is used to convert disk files containing data in any format into 79 character records containing only ASCII characters. Data in encoded format can be copied or transmitted by all Datapoint programs.

The DECODE command is used to translate encoded data files back into an exact duplicates of the original disk files.

24.2 Use

```
ENCODE <file spec>,[<file spec>]
```

The ENCODE command converts the first file into encoded format and writes the data into the second file. If extensions are not supplied, ABS is assumed for the first file and ENC is assumed for the second file. If the second file is not specified, the name of the first file with an extension of ENC is assumed. The second file will be created if it does not already exist. Encoded data creates a file 50 percent larger than the original.

```
DECODE <file spec>,[<file spec>]
```

The DECODE command converts the first file from encoded format back into binary and writes the data into the second file. If extensions are not supplied, ENC is assumed for the first file and ABS is assumed for the second file. If the second file is not specified, the name of the first file with an extension of ABS is assumed. The second file will be created if it does not already exist.

ENCODE reads and converts binary data until either a valid text end of file is read or allocated file space is exausted. Data in encoded form is always terminated with a valid text end of file.

24.3 Error Messages

```
INPUT FILE MUST BE SPECIFIED!
```

will be displayed if the first file specification is omitted.

```
INPUT FILE DOES NOT EXIST!
```

will be displayed if the first file specified cannot be found in the DOS directory.

```
OUTPUT WOULD DESTROY INPUT FILE!
```

will be displayed if the first and second file specifications are identical.

```
INPUT FILE CONTAINS BAD DATA!
```

will be displayed if an encoded data file cannot be decoded into its original binary form.

```
INVALID DRIVE
```

will be displayed if any drive specified on the command line is invalid.

CHAPTER 25. FILES COMMAND

FILES is a program which selectively prints or displays DOS file descriptions in file name sequence.

One may select information pertaining to all DOS files or to only those files with names and/or extensions beginning with the characters specified by the operator. Selected directory entries are sorted into ascending file name sequence. If desired, information from associated Retrieval Information Blocks (described in the chapter on System Structure) is also extracted for each directory entry. Extracted data is interpreted and displayed on the screen, listed on a serial printer, or written to a disk file.

25.1 Command Description

To execute the FILES program, type in the name FILES followed by selection criteria and display options (if option codes are to be used):

```
FILES [<name>][/<ext>][:DRn],[<subdir>][,<output-file>][;options]
```

<name>	Select entries for files with names beginning with the 1-8 characters specified.
<ext>	Select entries for files with name extensions starting with the 1-3 characters specified.
:DRn	Specifies the disk drive to be selected. If this field is omitted, the booted drive will be selected.
<subdir>	Specifies the named subdirectory from which to select entries.
<output-file>	Specifies the disk file to which the selected entries will be written, if disk file output is specified.

options: The following option codes are available, and may be entered in any order:

- N - Suppress file allocation map.
- D - Display on CRT.
- L - List on printer.
- F - Write output to disk as DOS text-format file.

If options are keyed and D, L,and F are omitted, then D is assumed. D, L, and F options are mutually exclusive; output can be sent to only one device. If F is keyed and the <output file spec> is not present in the command line, one is requested by the message:

```
DOS OUTPUT FILE SPEC:
```

25.2 Default Messages

If no option codes are entered, the following messages will be displayed on the CRT:

```
SUPPRESS FILE ALLOCATION MAP?
```

If "Y" or "YES" is entered in response to this message, the display of file allocation information from Retrieval Information Blocks (RIB) will be suppressed. If any other response is entered, file allocation information will be displayed for each selected file. Next, the following message will be displayed:

```
LIST ON PRINTER?
```

If the user enters "Y" or "YES" in response to this message, the printer will be selected for output. If a printer has been selected for output, the following message will be displayed:

```
ENTER HEADING:
```

Up to 32 characters can be entered, which will be displayed at the top of each page of printed output.

If no printer is available, or if the operator has rejected printer output, the program will ask for disk output:

WRITE OUTPUT ON DISK?

If the user enters "Y" or "YES", output will be written to a disk file, otherwise output will be displayed on the CRT. If disk output is selected, an output file name will be requested unless one was provided on the command line.

One use for a list of files on the disk is for generating a CHAIN file to perform repetitive operations such as COPYing or KILLing several files. The editor may be used to modify the file written by FILES, as shown in the Editor command string example.

25.3 File Descriptions

File descriptions are sorted into ascending file name sequence for easy reference and displayed or printed in the following format:

```
FILENAME/EXT (PFN) DW
```

DW flags following the Physical File Number (PFN) indicate if the file is delete protected (D), or write protected (W). If the file allocation map was not suppressed, messages describing the file's size and location will be included in the file description. When allocation map information is printed or displayed, the program displays totals lines specifying the total number of files listed and the total number of sectors in those files. Disk output never has totals lines.

Depressing the display (DSP) key during display or printing of file descriptions will cause the program to pause until the key is released. Depressing the keyboard (KBD) key will cause the program to terminate and return control to the operating system.

Allocation map information describes each segment in the file by giving the track and cluster starting address of the segment and its length in sectors. One line is displayed for each segment. See the chapter on System Structure for a description of disk space allocation.

25.4 Error Messages

* PARITY ERROR *

FILES can not continue due to an unrecoverable parity error encountered while trying to read data from the disk.

* DRIVE OFFLINE *

FILES is unable to connect to the disk drive selected by the operator (booted drive if not otherwise specified).

FILE(S) NOT FOUND.

No Directory entries have been found that meet the user's selection criteria.

INVALID DRIVE

An invalid drive specification was entered.

CONFLICTING OPTIONS SPECIFIED

Options specify output on more than one device.

UNRECOGNIZABLE OPTION CODE

An unrecognizable code has been entered in the option field.

PRINTER NOT AVAILABLE

An option code specifies a printer that is in use by the other partition.

PRINTER OFF LINE

The printer responded with a status of off line.

ERROR IN DOS FUNCTION

A DOS function call to read the RIB for a file has returned in error.

SUBDIRECTORY NOT FOUND

The subdirectory name specified on the command line can not be found.

CHAPTER 26. FIX COMMAND

26.1 Purpose

The FIX program can be used to modify bytes of DOS-loadable object code in an absolute code file. This program can be very dangerous and should be used only by qualified assembler language programmers or by someone following specific directions provided by Datapoint.

26.2 Operation

To invoke FIX, enter the command:

```
FIX <file spec>
```

The program will display a sign-on message and will then display an initial line of six zeros, two spaces, and three more zeros on the bottom CRT line. (The zeros represent the current address and its contents.)

```
000000 000
```

The screen is then rolled up. The program then waits for a command from the operator. The <file spec> must specify a DOS-loadable object file. If no extension is provided, ABS is assumed.

Commands are in the form [number][character] where the number is assumed to be octal. If the number is omitted, a value of zero is used. Commands are terminated by the ENTER key. Following a command, the current address and its contents are re-displayed.

26.3 Commands

The following is a list of command characters with their effect:

ENTER - Set current address. There are three cases for ENTER given below:

If no block of object code is currently in memory (as at the beginning of execution or after a block has been rewritten), search the object file forward until a block containing the given location is found, then display the contents of that location. If the address does not exist in the object file, the current address is left at zero.

If a block of code is in memory and the location given is within the limits of the block, the contents of the location will be displayed.

If a block is in memory and the location given is not within the block limits, the current address will be set to the minimum or maximum address of that block, its contents will be displayed and a beep will sound. To access the desired address the current block must first be aborted (A) or transferred (T).

nnnM	- Change the contents of the current address to the number given.
I	- Increment the current address (up to the maximum address in the current block).
nnn.	- Change contents of current address to number given and automatically increment the current address and display the contents of the resulting location.
D	- Decrement the current address (down to the minimum address in the current block).
T	- Transfer the modified block back to disk - rewriting it in place. After the block is written, the current address is set back to zero, so that all searches always start from the beginning of the file. No modification is made to the stored file until a T command is executed.
A	- Abort processing the current block, set the current address back to zero.
O OR *	- Return to the operating system - if there is a block of object code in memory, it is <u>not</u> written back into the file.

If the command character is not one of the above, it is ignored and regarded as if only the ENTER key had been pressed.

26.4 Error Messages

If the <filespec> is not an absolute object code file, the message

```
     RECORD FORMAT ERROR
```

is displayed.

If the file specified on the command line is not found, the message

```
     NO SUCH NAME
```

is displayed.

If the file specified on the command line is an overlay library, the message

```
     YOU CAN'T FIX AN OVERLAY LIBRARY
```

is displayed.

CHAPTER 27. FREE COMMAND

27.1 Purpose

As a disk becomes full, it is useful to know how many 256-byte sectors remain available for allocation. Other useful information is the number of empty slots in the directory remaining for the allocation of file names. The FREE command displays these two values.

27.2 Use

The FREE command accepts a drive specification. It may be entered simply as:

```
FREE
```

which will cause the free space and files for all the on-line drives to be displayed. It may also be entered as:

```
FREE :<drive spec>
```

which will display the FREE space and files for the specified drive only.

The command scans all drives that it finds on-line and displays (1) the number of available file names (representing possible files to be created) and (2) the number of available sectors it finds on each.

Holding down the display (DSP) key will cause FREE to pause. Pressing the keyboard (KBD) key will cause FREE to terminate and return to the operating system.

27.3 Error Messages

If the drive specification entered on the command line is not within the range of valid drives, the message,

```
BAD DRIVE SPECIFICATION
```

is displayed. If the call to the DOS function to get the number of clusters per cylinder returns in error, the message

```
BAD DOS FUNCTION
```

is displayed.

CHAPTER 28. INDEX COMMAND

28.1 Introduction

The DOS INDEX command (with the DOS SORT Command) is used to create the tree structure required by programs using the indexed sequential access method (ISAM), to create a Keytag file from the INDEX file, to create the INDEX file from a Keytag file, or to recreate the INDEX file.

The INDEX command has the capability of creating index files from any DOS text files. The indexed access method can then rapidly access records in this file either in sequential or random order. Records in files to be indexed must contain a record key up to 118 characters long contained in the first 249 bytes of each record.

It is possible to build many independent indices to permit access to records of the same file by many separate, unrelated keys. There are no restrictions on the number of indices that may be built, or on the relationship or lack of relationship among the various keys used.

28.2 System Requirements

INDEX runs under the DOS operating system. In addition, INDEX uses the DOS SORT command, which must be resident on an online disk at the time INDEX is used. If the INDEX command is to pre-process the text file, the REFORMAT command must be available. (See the Section on PREPROCESSING the file). If the INDEX command is to be used to recreate the tree structure file, the NAME command must be available.

28.3 Operation

When the INDEX command is to be executed, the operator must enter:

```
INDEX <filespec>[,<filespec>][,<filespec>][,<drive>];<parameters>
```

where only the first file specification and key field description are mandatory, and specify the file to be indexed. The default extension is TXT. The second file specification is the name of the INDEX file to be created. If no file is specified, the name of the first file is used with default extension of ISI. If no drive is specified, the INDEX file will be placed on the same drive as the file to be indexed. INDEX files may have any names at all - and be located on physically different drives from the file being indexed. However, high-level languages using ISAM files (DATABUS, for example) assume the INDEX file will have the normal ISI extension, and if the input file is drive-directed, the ISI and TXT files must be on the same drive.

The third file specification is for the intermediate tag file. The third file name will also default to the name of the first file with a default extension of TAG. The fourth file specification, which may <u>only</u> specify drive, tells SORT where to put its intermediate work files. Otherwise, SORT will attempt to optimize drive selection.

28.3.1 Parameters

In addition to the parameters that INDEX itself recognizes, the user may specify any parameters acceptable to the REFORMAT utility (if preprocessing is to be done), or a primary record specification to be passed to SORT. Parameters recognized by INDEX are as follows:

```
          K -- Create a Keytag file from the ISI file.
          I -- Create an ISI file from the Keytag file.
          X -- Recreate the ISI file, handling insertions and
               deletions.
          F -- Preprocess the input file with REFORMAT.
          E -- Index in EBCDIC collating sequence.
    mmm-nnn -- Key specification
```

The Keytag file is a standard text file containing the pointer and key of each record to be indexed. The format is explained in the SORT chapter. The file may be LISTed, EDITed or transmitted. This last feature allows the ISI file to be created at a remote site without invoking SORT.

The format of the key is mmm-nnn [,mmm-nnn] [,mmm-nnn]... where mmm is the beginning character position of the key field in each logical record and nnn is the ending position of the key field. Note that each record must have a unique key.

The primary record specification is an option that allows the user to create the ISAM index file from a subset of the data file. The format of the primary record specification is PNNNTC. The P must always appear. The field following P, denoted by NNN, represents the column in each logical record where a one position field exists that differentiates records in the file. The location of this one character field must be 1 to 249. The T can have one of two values. It can be either an equal sign (=) or a pound sign (#). If the former, it means create the ISAM index file from all records that contain the ASCII character C in position NNN. If it is a pound sign, it means that the ISAM file will be created from all records that do not contain the value of C in position NNN.

In general the parameters for INDEX can be specified in any order and may optionally be separated from each other by a blank or a comma. The only exception to this is when a primary record specification exists, it must precede the key field specification and be separated from the key by a blank or a comma.

28.4 Choosing A Record Key

Since the speed of access to an indexed file varies according to how much file space and thus how many levels of index are required for the index tree, the choice of what to use for a record key becomes highly important. Of course, you must choose a key which will uniquely determine the record you wish to access, but you should scrupulously avoid including information in the key which is not absolutely necessary. For example, a file could be keyed according to automobile license plate numbers. Typically, these numbers will include a hyphen or other punctuation, which could easily be excluded from the record's key. The indexed access method will perform more efficiently if all non-significant characters are removed from the record's key.

28.5 Preprocessing the File

In file structures such as an indexed file where records are randomly inserted and deleted, the file tends to become, after much use, non-optimal for searching.

In order to reclaim space vacated by deleted records and padding bytes in inserted records, the file may be processed by the REFORMAT utility prior to indexing.

28.5.1 Invoking Reformat

The INDEX utility will automatically invoke REFORMAT if the "F" option is present when INDEX is invoked. You must have specified the options that REFORMAT will need to process the file. CAUTION: when INDEX invokes REFORMAT, it is done so in place.

Note that if multiple indices are to be created, reformatting need only be specified for the first INDEX step, and MUST not be specified later if it was not specified in the first step. Although REFORMAT will not destroy the file, specifying reformatting may invalidate any previously built indices.

Basically, you must tell REFORMAT what format the records of the file are to have after preprocessing. You may select record compression, space and record compression, or blocking. Since the reformatting is done in-place, the REFORMAT option cannot enlarge the file which is to be indexed. For additional details on the REFORMAT utility, see the REFORMAT section of this guide.

28.5.2 Considerations for Unattended Indexing

Users who use the INDEX command from a CHAIN file (see the section on the CHAIN command for more details) and used AUTOKEY to restart their chain in the event of a failure should generally avoid using REFORMAT directly from INDEX. The reason is that REFORMAT (when invoked by INDEX) uses the REFORMAT-in-place mode of the REFORMAT command. This is because it is faster, and allows the reformatting of a file which is too big for the available scratch space on a single-drive, almost full diskette. Although REFORMAT is very careful not to damage the file being processed, if the file is actually in the process of being reformattedd in place when a power failure occurs, the results can be undesirable.

This potential problem during unattended INDEX chaining can be avoided by setting a checkpoint (see the AUTOKEY command description for details), copying the original file to a scratch file, setting another checkpoint, reformatting the scratch file back into the original (using a separate REFORMAT command) , setting a further checkpoint, and finally INDEXing the file using INDEX. In this way there is always an undamaged file with which execution can resume if necessary.

28.6 INDEX Messages

The INDEX command displays several messages on the operator's console. They are listed below with explanations, in the sequence in which they may appear.

DOS.H VER n.n INDEX COMMAND - date
This is the signon message that gives the user the version of DOS required and the date of the INDEX command.

INFILE NAME MISSING.
This indicates that the user has omitted the first, and required, file specification.

KEYTAG FILE BEING BUILT.
This indicates that INDEX is now creating the ASCII KEYTAG file requested with the "K" option.

SORT COMMAND MISSING.
This indicates that INDEX needs to invoke the SORT command but could not find it on any of the on-line drives.

FILE PREPROCESSING WILL BE DONE BY REFORMAT COMMAND.
This indicates that the user has requested preprocessing of his file by the REFORMAT command.

REFORMAT COMMAND MISSING.
This indicates that INDEX needs to invoke the REFORMAT command but could not find it on any of the on-line drives.

REFORMAT COMMAND LINE:
This is the parameter list passed to the REFORMAT command.

INDEX WILL USE EBCDIC SORT.
The user has requested an index using the EBCDIC collating sequence.

SORT COMMAND LINE:
This is the parameter list passed to the SORT command.

REFORMAT UNLOADABLE!
This indicates that there is something wrong with the REFORMAT command object file. It needs to be

reloaded.

SORT UNLOADABLE!
This indicates that there is something wrong with the SORT command object file. It needs to be reloaded.

BUILDING LOWEST LEVEL INDEX.
This indicates that INDEX is now creating the lowest level of the index file.

NULL INDEX FILE CREATED.
This indicates that an empty tag file was created by SORT. The index file created is usable by programs using ISAM for adding records.

LONG KEY ENCOUNTERED AND TRUNCATED.
This indicates that the tag file contained a key that was longer than 118 characters. It was truncated to 118 characters.

DUPLICATE KEY: <key>
Two keys in the tag file were found to be identical and the first 60 characters of the key are displayed. INDEX will continue so as to display any other duplicate keys that may be found.

INDEX TERMINATED WITH DUPLICATE KEYS.
Duplicate keys have been found and a null ISI file will be created. ABTIF bit set; CHAIN not terminated. The tag file is not deleted and since it is in standard text format, it may be EDITed to remove or modify the duplicate key and tag. Or a program (e.g. in DATABUS) may be written to display the records containing the duplicate keys so the user may resolve the ambiguity. INDEX may then be reinvoked using the "I" option.

BUILDING -NEXT- LEVEL INDEX.
This indicates that the lower level of the index file has been completed and the next level is now being created.

SYSTEM7/SYS MISSING!?
This indicates that an attempt to open SYSTEM7/SYS while looking for a subdirectory name has failed. SYSTEM7/SYS is assumed to be on the same drive as the ISI file.

DONE.
   The creation of the index file is now completed.

Other error messages may be generated by REFORMAT or SORT. See the appropriate chapter for an explanation.

28.7 ISI File Structures

The DOS indexed file structure consists of a multi-level radix tree structure based on the record keys, and contains pointers to the location of the keyed records. Note that since many of these pointers are physical disk addresses, the ISI file cannot be moved without re-invoking INDEX. The text file may be moved so long as it is unchanged in any way. Moving the ISI file will destroy it.

The different levels of indices all have the same content, except for the lowest level index. Index levels are built up until an intermediate level of index will fit in a single disk sector. This becomes the highest level of index. This requirement is the reason for the 118-character limitation on key length.

28.8 Examples of the Use of INDEX

First, a simple example in which only a single ISI file is created, with the same name and on the same device as the text file it indexes. The file is a list of bad checks presented at a local grocery chain, and now each store has a DATABUS terminal to inquire on the current status of each deadbeat. Thus, while the file is accessed often, additions and deletions are fairly infrequent, so the file will not be reformatted. The file is keyed by bank number (8 digits) and account number (7 digits) concatenated and in positions 1 to 15 of each record.

In order to create the index file, the operator must type:

```
INDEX DEADBEAT;1-15
```

The INDEX program will then create a file DEADBEAT/ISI which DATABUS can use to access the DEADBEAT/TXT file.

Now, this same grocery chain has expanded its operations, so it desires to include more information on the location and date of each NSF check presented. Therefore, they have expanded the file to include the old key in positions 1 to 15, a store location number in positions 16 to 18, and a date field in positions 19 to 24. As an afterthought, the manager decides to tack on the name of the person passing the bad check in positions 193 to 216.

In order to create the indices required for access by any of these keys, the operator must type:

```
INDEX DEADBEAT,BANK;1-15
INDEX DEADBEAT,DATE;19-24
INDEX DEADBEAT,STORE;16-18
INDEX DEADBEAT,NAME;193-216
```

The INDEX program will create four files with names BANK/ISI, DATE/ISI, STORE/ISI, and NAME/ISI. Each file is logically separate, yet all are on the same volume as DEADBEAT/TXT.

Now the store owners have uncovered a hitch - first, the number of bad checks is becoming so large, there is no room on one diskette for all the index files and the text file. In addition, access has been slowing way down as the frequency of additions and deletions increases. The store owners have called Datapoint to complain, and their local Systems Engineer has told them they need to REFORMAT the files when they re-INDEX, and has sold them another diskette drive.

The operator now types:

```
INDEX DEADBEAT,BANK/ISI:DR1;FR1-15
INDEX DEADBEAT,DATE/ISI:DR1;19-24
INDEX DEADBEAT,STORE/ISI:DR1;16-18
INDEX DEADBEAT,NAME/ISI:DR1;193-216
```

Note that the reformatting is done only once at the beginning. If reformatting had not been done when the first index was built, it could not be correctly done later without invalidating the previously built indices.

Now, several years later, the grocery chain has expanded and has a large diskette system at their main store. The owners are doing so much processing that there is not the time to run the above INDEX programs as each one invokes SORT. However, they wish to keep access time to the minimum. Also, the DEADBEAT file is so large that numerous additions and deletions hardly affect the size.

Every night the operator now types:

```
INDEX BANK;X
INDEX DATE;X
INDEX STORE;X
INDEX NAME;X
```

which recreates the index files. Then during weekly processing, the operator does the processing above which invokes REFORMAT.

The store owners have wisely dispersed some of their data processing to their branch stores. So each night the operator also types:

```
INDEX BANK;K
INDEX DATE;K
INDEX STORE;K
INDEX NAME;K
```

which creates tag files of the four indices. The operator then transmits DEADBEAT/TXT, BANK/TAG, DATA/TAG, STORE/TAG, and NAME/TAG to each of the branch stores. The operator at the branch store, after receiving these files, types:

```
INDEX DEADBEAT,BANK,BANK;I
INDEX DEADBEAT,DATE,DATE;I
INDEX DEADBEAT,STORE,STORE;I
INDEX DEADBEAT,NAME,NAME;I
```

which creates a local set of indices without invoking SORT.

Note: In the above example that created a BANK tag file, the command line with defaults is:

```
INDEX BANK/TXT,BANK/ISI,BANK/TAG;K
```

As only the ISI and TAG files are needed for creation of the tag file, the same results could have been achieved by typing:

```
INDEX ,BANK,BANK;K
```

CHAPTER 29. KILL COMMAND

29.1 Description

KILL - Delete a file from the directory

```
KILL [<file spec>]
```

The KILL command deletes the specified file from the system if the file is not protected. If the file is protected in any way, the message

```
NO!
```

will be displayed. If the file specification is not given on the command line (file names which contain special characters cannot be given on the command line), the request for the file name:

```
WHAT FILE?  EXAMPLE:  SCRATCH /TXT:DR1  #143     :DR1
         /   :DR
```

will appear. The user must keyin an eight character filename (including trailing spaces), a slash, a three character extension (including trailing spaces), a colon, the letter "D" and the drive number on which the file resides. If the entire filename specification is not entered properly, the message:

```
NO SUCH NAME.
```

will appear. A file can be specified by physical file number by entering "#", followed by the octal PFN, followed by 8 spaces and the drive specification. If the specified file cannot be found (both a name and an extension must always be supplied unless using PFN) , the message:

```
NO SUCH NAME.
```

will be displayed. If the file is found and is not protected, the message:

```
THAT FILE IS <filename> ON DRIVE n
```

will appear. Then the operator must additionally answer the message:

    ARE YOU SURE?

with a 'Y' before the actual deletion of the file is achieved.

NOTE: If Kill is used in a chain file, an answer to the question "ARE YOU SURE?" must be entered into the chain file as the Kill utility is expecting it. After the deletion has occurred the following message is displayed:

    * FILE DELETED *

29.2 Error Messages

If a drive specification is entered on the command line that is not within the range of valid drives, the message

    INVALID DRIVE.

is displayed. If a PFN is entered without a preceeding pound sign (#), the message

    THAT ISN'T THE RIGHT FORMAT FOR YOUR REPLY.

is displayed. If the format for the PFN is not exactly as described, the message

    INVALID FORMAT FOR PFN REPLY.

is displayed. If the PFN entered is the same as the auto execute PFN, the message

    AUTO CLEARED.

is displayed. If a 'N' is entered in response to the "ARE YOU SURE?" question, the message

    I DIDN'T THINK SO.

is displayed. If only the KILL command is entered and the program has responded with "WHAT FILE? EXAMPLE......" and an ENTER key is entered in response to that question, the message

    OKAY, SO YOU CHANGED YOUR MIND.

is displayed. If an OPEN on SYSTEM7/SYS fails, the message

```
     SYSTEM7/SYS MISSING !?
```

is displayed. It is assumed to be on the same drive as the file being killed.

CHAPTER 30. LIST COMMAND

30.1 Purpose

The LIST command will list any DOS standard format text file on the screen or a serial printer.

The command can be used for such things as:

A quick scan of a file by displaying it on the screen (LISTing a file is faster than EDITing it);

Producing a hardcopy listing of a file for permanent records;

Listing a file for use in preparation of a BLOKEDIT COMMAND FILE.

In this chapter, the following terms apply:

<u>Text file</u> means a file with records containing only ASCII characters, except for space-compression bytes and the End-Of-Record and end of file marks. Files created by EDIT and those produced by DATABUS are normally in the class of text files.

<u>Line</u> means one record of a text file. When displayed on the screen, only the first 72 characters of a record will be displayed; when listed on a printer only the first 124 characters will be printed. (The remaining eight characters contain a line number.)

<u>Record</u> means the user logical record number (LRN). The first LRN of a file is zero.

30.2 Parameters

When the LIST program is to be executed, the operator must type:

```
LIST <filespec> [,<spec2>][,<filespec2>][;options]
```

Available options are:

L	-	list on printer
D	-	Display on CRT
X	-	suppress line numbers
F	-	list Formatted print file
P	-	output formatted Print file
Q	-	Queue formatted print file, appending to an existing file
Nn	-	set Number of lines per page to n
I	-	list in Indexed sequence

Options may be entered in any sequence.

30.3 INPUT File Specification

The file specification (<filespec>) must refer to a DOS text file. If no extension is supplied with the file name, an extension is assumed depending on the options given. A default extension of TXT is assumed unless the option "I" or "F" is used. The option "I" (list a file using its index) causes a default extension of ISI and the option "F" (list a file with format control bytes) causes a default extension of PRT. If no drive is supplied with the file specification, all drives will be searched for the filename/ext. If <filespec> is omitted, the message

```
NAME REQUIRED.
```

is displayed. If the file indicated by <filespec> is not found on an online volume, the message

```
NO SUCH NAME.
```

is displayed.

30.4 Starting Point

The operator may specify a line number, or logical record number, in the file at which the list should begin by including an optional second parameter <spec2>. For example:

```
LIST <filespec>,L400
```

would list the specified file beginning with line 400 of the file. If the line number specification exceeds the number of lines in the file, LIST returns to DOS after displaying the message:

```
FILE EXHAUSTED BEFORE LINE FOUND.
```

```
LIST <filespec>,R18
```

would directly access logical record 18 of the specified file and list, starting at line number 1. If RANGE or FORMAT errors occur, the error type is indicated and another record number is requested.

For instance, if the record number specification exceeds the number of records, the message

```
RANGE - NEXT RECORD NUMBER:
```

is displayed.

The <u>DEFAULT</u> value for the second parameter is line 1 and record 0.

30.5 OUTPUT File Specification

If the options "P" (write to a print file on disk) or "Q" ('QUEUED' write to a disk print file starting at the end-of-file mark) are used, then the third parameter (filespec2) may be used to specify the output file. If the filename is not given, it is assumed to be the same as the input file name. If the extension is not given, it is assumed to be PRT.

Output from either the "P" or "Q" option is a text file with print control characters as described in the Format Control section. The file will be paged with headings; line numbers will be included unless suppressed by the "X" option.

30.6 Output Device

The operator may specify an output device other than the CRT display by including an optional parameter of "L" (serial printer), "P", or "Q". For example:

```
LIST <filespec>;L
```

would list the specified file on a Datapoint serial printer beginning at line number one.

For either print or disk output, LIST will request a heading, which will be placed at the top of every page of output.

The DEFAULT output device is the CRT display which may be specified by entering a "D".

30.7 Output Format

A parameter is available to suppress line numbers. If the 'X' is entered, lines of up to 132 characters will be printed. For example:

```
LIST <filespec>;LX
```

would put the output on the printer without line numbers,

```
LIST <filespec>;X
```

would display the listing, showing 80 characters per line on the CRT.

Any paged output (from the "L","P", or "Q" options) is normally listed at 54 lines per page. The "Nn" option can be used to change the number of lines per page, n being the desired lines/page count.

30.8 Format Control

The parameter "F" is available to allow the handling of print files (those with a format character in the first column of each line). If "F" is entered, the file will be listed without line numbers, page numbers, or headings, since all these items should already be in the print file. The following format characters cause the indicated action to be taken before the line is printed.

1	- Skip to top of form
+	- Suppress line feed
(space)	- Single line feed
0	- Double line feed
-	- Triple line feed

Any other character in the first column will be handled as a space (single line feed) and discarded.

30.9 Operator Controls

The listing consists of a continuous stream of the listed file's text. To cause the listing to pause, the operator may hold down the display (DSP) key. To abort the listing, the operator may depress the keyboard (KBD) key.

30.10 Error Conditions

If printer output was specified and the printer is not available, LIST beeps and displays the message:

```
PRINTER NOT AVAILABLE
```

is displayed. If the printer is offline, the message:

```
PRINTER OFF LINE
```

is displayed. If the printer is made ready, listing will proceed. The keyboard (KBD) key may be depressed to abort the LIST at this point if necessary.

LIST checks to be sure the text end of file is exactly six zeroes and a three (see Text File Formats in the REFORMAT chapter). If the EOF is not exactly correct, LIST displays the message:

```
INVALID END OF FILE.
```

is displayed. LIST can be used to test for a bad EOF since most text-handling programs are not so particular about EOF format.

When <spec2> has been entered to start LIST at a particular record number, LIST traps FORMAT or RANGE errors and allows a new starting location to be entered. In any other usage, LIST does not trap FORMAT or RANGE errors and any such errors are fatal. If LIST does trap FORMAT and RANGE errors, either of the following messages may be displayed.

```
RANGE - NEXT RECORD NUMBER:
FORMAT - NEXT RECORD NUMBER:
```

The operator has the option to enter the next record number to be processed by LIST at this time. If the "I" option is used LIST will search for the indexed file on all drives and if not found, the message

INDEXED FILE NOT FOUND!

is displayed. It should be noted at this point, that since the indexed file can be on any drive and possibly on multiple drives, only the indexed file with the same physical file number (PFN) as contained in the index file will be accepted as the correct file.

If any parameter other than the allowable parameters is entered on the command line, the message

PARAMETER ERROR. ALLOWED PARAMETERS ARE D,L,P,Q,X,Nn,F, and I.

is displayed. If the "L", "P", or "Q" options are entered such that more than one of them exists on the command line, the message

PARAMETER ERROR. MULTIPLE PRINT DEVICES REQUESTED.

is displayed. If the number of lines per page is specified on the command line as more than 255 or less than 1, the message

NUMBER OF LINES PER PAGE MUST BE 1 - 255!

is displayed. If the second parameter on the command line entered is not in the form "Ln" or "Rn", the message

BAD SECOND FILE SPEC! SHOULD BE "Ln" OR "Rn"!

is displayed. If the ISI file has been moved, the message

INDEX FILE DOESN'T POINT TO ITSELF!

is displayed. If the "I" option has been specified with either of the "L" or the "R" options for the second parameter on the command line, the message

ISAM AND LINE/RECORD COUNT INCOMPATIBLE.

is displayed. If an invalid end of file is detected anywhere in the specified file, the message

INVALID END-OF-FILE AT LRN XXXXX.

is displayed with the logical record number (LRN) displayed within the message. If the physical end of sector mark is missing in any of the sectors processed, the message

MISSING END-OF-PHYSICAL-RECORD AT LRN XXXXX.

is displayed with the logical record number (LRN) displayed in the message.

CHAPTER 31. MANUAL COMMAND

The MANUAL command is used to Clear Auto Execution. Its syntax is as follows:

```
MANUAL
```

If the auto-execution name has not been set the message

```
AUTO NOT SET.
```

will be displayed. Otherwise, the System Table location reserved for the auto-execution information will be cleared and the message

```
AUTO CLEARED.
```

will be displayed. If the booted drive is off line, the message

```
* SYSTEM DRIVE OFF LINE *
```

will be displayed. If a parity error is encountered when attempting to read in the AUTO command from disk, the message

```
* SYSTEM DATA FAILURE *
```

will be displayed.

CHAPTER 32. NAME COMMAND

The NAME command is used to Change the name, extension or subdirectory of a file. Its syntax is as follows:

```
NAME <file spec1>,[<file spec2>][,<subdirectory name>]
```

NAME will not affect the content of the files. The first file specification refers to the current file name and the second file specification is the new name and/or extension to be assigned. If no extension is supplied in the first file specification, ABS is assumed. If no extension is supplied in the second file specification, the extension of the first file is assumed. If no extensions are supplied, both files will be assumed to have extensions of ABS. The drive number should only be specified in the first file specification.

If the NAME command is used to move a file from one subdirectory to another the second file specification may be omitted (unless the filename and/or extension are to be changed) and the subdirectory name denoting the subdirectory into which the file is to be placed is the third specification:

```
NAME <file spec1>,,<subdirectory name>
```

In both uses of the NAME command, two specifications are required. If either name is not given, the message

```
NAME REQUIRED.
```

will be displayed. If the second name is already defined on the drive that contains the first file, the message

```
NAME IN USE.
```

will be displayed. Note that the drive specification on the second name is ignored. If the first name is not found on an online disk, the message

```
NO SUCH NAME.
```

will be displayed. If the subdirectory name keyed is not found on the disk containing the file to be renamed, the message

NO SUCH SUBDIRECTORY.

will be displayed. If the third parameter is not specified, the file is "brought into" the current subdirectory at the completion of the renaming process. If an attempt is made to change the name of a file which is write protected, the message

NO! THAT FILE IS PROTECTED.

is displayed. If an attempt is made to name a file into a non - existent subdirectory name, the message

NO SUCH SUBDIRECTORY

is displayed. If SYSTEM7/SYS does not exist on the disk on which the file is being renamed, the message

THAT DRIVE HAS NO SUBDIRECTORY

is displayed. If any errors are encountered while attempting to read or write the directory, the message

* SYSTEM DATA FAILURE *

will be displayed.

CHAPTER 33. PUTIPL COMMAND

The PUTIPL command writes an IPL (Initial Program Loader) block and DOS boot blocks to the diskette.

```
PUTIPL <:DRIVE>
```

If the drive number is not specified in the command line, PUTIPL will display the following:

```
LOGICAL DRIVE TO BE WRITTEN (0..n OR "*" TO EXIT TO DOS)
```

Where n = 1,3, or 15 depending on the maximum number of logical drives currently recognized by DOS. Respond with the drive number that you want to write to.

NOTE: IPL blocks cannot be written to 9320 disks.

This program is invoked automatically by the DOSGEN and BACKUP utilities. It may also be used when upgrading diskettes, or as part of a recovery procedure if information on the diskette has been damaged. If the specified drive is off-line, the message

```
OUTPUT DRIVE IS OFF-LINE.
```

will be displayed. If a parity error is encountered while attempting to write to the diskette on the output drive, the message

```
WRITE PARITY ERROR ON OUTPUT DRIVE.
```

will be displayed. If no problems are encountered, the program will display the message

```
THE SECTORS HAVE BEEN SUCCESSFULLY WRITTEN
```

and then exit. If an attempt was made to write the blocks onto a 9320 disk the program will display the message

```
BOOT BLOCKS CAN ONLY BE WRITTEN ON DISKETTES
```

and then exit.

CHAPTER 34. PUTVOLID COMMAND

The PUTVOLID command writes a symbolic volume identification (VOLID) onto a disk.

```
     PUTVOLID <VOLID><:DRIVE>;<OWNER ID>
```

Where VOLID is 1 to 8 characters in length, DRIVE is the logical drive to be written to, and OWNER ID is any information the user wants. The VOLID may contain only alpha-numeric characters; special characters are not allowed.

If only a drive number is entered, the existing VOLID for that drive will be displayed. If the drive number is not entered, the message

```
     OUTPUT DRIVE NUMBER REQUIRED.
```

will be displayed. If the drive number specified is off line, the message

```
     :DRn OFF-LINE.
```

is displayed with the off line drive number in the message. If the DOS identification letter does not already exist in the volume identification for the drive specified, the message

```
     VOLUME NAME MISSING.
```

will be displayed. If a write parity error (CRC) is encountered on the output drive, the message

```
     CRC ERROR ON :DRn.
```

will be displayed with the drive number displayed in the message. If the <OWNER ID> is not entered, the message

```
     MISSING OWNER-ID.
```

will be displayed.

CHAPTER 35. REFORMAT COMMAND

35.1 Introduction

The DOS REFORMAT command is used to change the internal disk format of text (non-object) files. Additionally, it can recover disk space left unused when files are updated by the DATABUS indexed sequential access method. REFORMAT can compress a file in place on disk provided that such compression does not entail the writing of a logical disk sector prior to the time that sector is read. REFORMAT maintains logical consistency in such cases and will not write on a disk file until it has checked to be sure it can complete its job successfully.

35.2 Operation

When the REFORMAT program is to be executed, the operator must type:

```
REFORMAT <file-spec>[,<file-spec>][;<parameters>]
```

where only the first file specification is mandatory, and specifies the file to be reformatted. If the second file specification is given, it must be distinct from the first. Reformatting in place is requested by omitting the second file specification.

The parameter list describes the format the output file is to take, and whether REFORMAT is to free any disk space that might be vacated by the reformatting process. In addition, the user can specify that REFORMAT is to pad short records, and either truncate or segment long records. REFORMAT will produce three different kinds of output files: record compressed, space and record compressed, or blocked records (see the section on TEXT FILE FORMATS). Note that REFORMAT will not produce blocked space compressed records or space compressed non record compressed files although such files can be used as input to the REFORMAT program. If no parameters are given, the output file is blocked one record per sector.

Parameters passed to REFORMAT may be separated by spaces or commas. The valid parameters are as follows:

Parameter	Description
B<n>	The output file will be blocked. This implies no space or record compression, with <n> logical records per physical sector.
C	The output file will be space and record compressed. The number of logical records per physical sector will be indeterminate.
R	The output file will be record compressed, but no space compression will be done. In general, the number of logical records per physical sector will be indeterminate.
L<n>	The length of each logical record will be adjusted to <n> characters. Note that if the logical records are space compressed, this will not make the physical length of the records <n> characters. If the logical record is shorter than <n> characters, it will be padded with blanks to the proper length. If the logical record is longer than <n> characters, the action taken depends on the T and S parameter.
T	(Only valid if L parameter is given) Truncate the logical record if it is longer than <n> characters.
S	(Only valid if L parameter is given) If the length of the logical record is greater than <n> characters, segment it into (q) logical records each of length <n>, padding if necessary. The number (q) is defined as input length divided by <n> rounded upward to the next integer. If neither S nor T is specified, and an input record of length greater than <n> is found, a message is issued and REFORMAT gives up.
D	If reformatting is done in place and this parameter is specified, any disk space vacated by the reformatting process will be returned to the operating system for re-use.

35.3 Output File Formats

The REFORMAT utility permits you to select essentially three different output file formats. It will produce blocked files that are not space compressed, record compressed files that are not space compressed, and files that are both record and space compressed. In addition, it has a subcommand to permit you to specify the logical length of the output records. Use of this subcommand will guarantee that each record has exactly the same logical length. Note that if the output format does not specify space compression, the physical length of each record will be identical. This is especially useful for telecommunications disciplines that require records of fixed length.

If you have set a fixed logical length for output records, there are two subcommands available to tell REFORMAT what to do with records whose logical length exceeds the specified output length. You may select either truncation of the input record, or you may segment it into two (or more) output records, each of the logical length specified.

35.4 Reasons for Reformatting

Several uses of REFORMAT deserve special mention. First, a random disk file is structured to have one logical record per physical sector. Often, however, it is convenient to create a random file through the use of the general purpose editor - which record and space compresses its output. REFORMAT can then reprocess the file into the correct format for DATABUS random access.

Secondly, when a file is accessed with DATABUS indexed sequential access method, any additions may result in an increase in the physical size of the file. Similarly, deleted records are simply overstored with octal 032 (logical delete) characters, and the space they vacate is not reused. REFORMAT recognizes this condition, and will recover such vacated space. Note that ISAM read-only or update-only (no additions or deletions) files do not usually need reformatting.

35.5 Reformat Messages

The REFORMAT utility program produces several messages on the operator's console. The contents and, where necessary, meaning of those messages follow:

DOS.H VER n.n REFORMAT COMMAND - date
Self-explanatory sign on message.

COMMAND LINE ERROR: 015 missing
This is an internal error and should be reported to Datapoint.

PROGRAM ERROR - EXCESS FILE SPACE NOT DEALLOCATED TO PREVENT POSSIBLE LOSS OF DATA
REFORMAT has detected an invalid end of file mark. In order to prevent the possible loss of data which might be after the invalid end of file indicator, space allocated but unused is not freed.

EXCESS FILE SPACE NOT DEALLOCATED; OUTPUT FILE IS DELETE PROTECTED.
Self-explanatory.

OUTPUT FILE IS WRITE PROTECTED AND CANNOT BE WRITTEN INTO OR SHORTENED.
You have requested REFORMAT to output to a write-protected file.

INVALID OPTIONS SPECIFIED
You have given REFORMAT an invalid parameter list. This message is followed by the valid options you may specify.

ILLEGAL, CONFLICTING OR DUPLICATE OPTIONS
You have specified two mutually exclusive options.

YOU SPECIFIED BOTH SEGMENTATION AND TRUNCATION. YOU CANNOT HAVE BOTH
Self-explanatory.

BLOCKING FACTOR CONTAINS INVALID NON-NUMERIC DIGITS
Self-explanatory.

BLOCKING FACTOR REQUIRED BUT MISSING OR ZERO FOUND
You specified blocking but omitted the blocking factor.

LOGICAL RECORD LENGTH REQUIRED BUT MISSING OR ZERO FOUND
You must specify the logical record length of the output file if you wish to have fixed length output records.

YOU HAVE ILLEGALLY ENTERED A SPECIFICATION FOR
A THIRD FILE
REFORMAT recognizes only two file specifications.

HOW DO YOU EXPECT TO FIT THAT MANY RECORDS IN A
256 BYTE SECTOR?
Self-explanatory.

LOGICAL RECORD LENGTH, IF SPECIFIED, MUST
BE <= 250 BYTES.
Self-explanatory.

YOUR BLOCKING FACTOR IS TOO LARGE FOR THE SIZE
OF THE RECORDS YOU HAVE.
Self-explanatory.

YOUR LOGICAL RECORD LENGTH IS TOO SMALL FOR THE
SIZE OF THE RECORDS YOU HAVE
While processing the input file, REFORMAT came across a record that was larger than the specified logical record length. Since you specified neither segmentation nor truncation, this is recognized as an error.

SPECIFIED OUTPUT FILE FORMAT ENLARGES PRESENT
INPUT FILE. FILES CANNOT BE ENLARGED DURING
REFORMAT-IN-PLACE. REFORMAT IN-PLACE REQUEST
REFUSED.
Self-explanatory.

YOU SPECIFIED AN OUTPUT FILE THAT ENDED UP
BEING YOUR INPUT FILE. TO REFORMAT IN-PLACE
DO NOT SPECIFY ANY OUTPUT FILE.
Self-explanatory.

OUTPUT FILE NOT FOUND ON DRIVE X.
OUTPUT FILE FOUND ON DRIVE Y.
OUTPUT FILE WILL BE CREATED ON DRIVE Z.

These messages only occur if no specific drive was indicated for the output file. The first message appears followed by either the second or third. REFORMAT could not find the output file on the same drive as the input file. It either found one on a different drive, or created one on the displayed drive. If the output file is created, it is always created on the same drive as the one the input file is on.

REFORMAT IN-PLACE REQUESTED.
PRESCAN IN PROGRESS.

REFORMAT is checking to make sure it can properly process the file in-place.

FILE ALREADY WAS IN THE SPECIFIED FORMAT

Self-explanatory.

COPYING WITH REFORMATTING IN PROGRESS

Self-explanatory.

REFORMAT-IN-PLACE IS IN PROGRESS.
DO NOT DISTURB!!!

Self-explanatory.

NAME REQUIRED

Either you gave only an extension or drive for the input file, or you specified the output file first, followed by the input file.

INVALID DEVICE

An invalid drive was specified for the input file.

NO SUCH NAME

The input file specified cannot be found.

INVALID DRIVE SPECIFICATION

The drive specification entered for one of the file specifications was not in a valid format.

35.6 Text File Formats

Under Datapoint Corporation's Disk Operating System, text files consist of legal ASCII characters, which make up the text itself, and various control characters with special meanings. It is illegal to have the control characters in the text portion of the file. According to DOS convention, any character between 000 and 037 is considered a control character.

Each physical record of a text file is a logical diskette sector, and contains 256 characters. The first three and last two characters are reserved for control functions; hence, the maximum space available in a single physical record is 251 bytes. The lengths and offsets given below are octal (base 8). The format of a logical sector is as follows:

Offset	Length	Description
000	001	Physical file number of this file. For a detailed description of physical file organization, see the chapter on System Structure.
001	002	Logical record number. This refers to logical physical records, and is not related to text records within the file.
003	373	Text. 251 bytes of text and control characters, depending upon the format of the file.
376	002	Two characters reserved.

The text part of each file is considered a logical stream, crossing sector boundaries without being logically discontinuous. Demarcations of logical record boundaries are made solely by control characters imbedded within the text itself. There are essentially five control characters found in files generated by DOS: 000 <NUL> used for end of file indication, 003 <EM> used to denote the end of medium (a sector boundary) but not the end of a logical record, 011 <CMP> used to denote space compression, 015 <ENT> used to denote the end of a logical record, and 032 <DEL> used to denote deleted data.

Under DOS each file is treated as a single, continuous stream of data. Physical records bear no relation to the logical structure of the data contained in them. In this way, a proliferation of different file structures, and the special routines needed to treat such special cases has been avoided. This does not mean that there cannot be a relation between physical and logical structure, it simply means that such a relationship is incidental to a particular file, and need not be treated as a special case. For example, random access to a data file is defined in the DATABUS language. Files to be accessed in this manner are structured in such a way that one logical record corresponds exactly with one physical record. This structure is not inherent in the makeup of a random file, in fact, such files can be treated exactly as any other text file.

The basis for this treatment of text files is the logical record. A logical record starts at the beginning of a file, or immediately after the end of a previous logical record. It consists of ASCII data and is of no pre-determined length. Instead, the record is terminated with a single ENT character. In this way, complications arising from a multitude of record types are entirely avoided.

If the logical record contains any CMP characters, it is said to be space-compressed. The character immediately following the CMP character is a space count, and the pair represent the number of ASCII blanks removed when the record was compressed. Since the character following CMP is always assumed to be a space count, CMP can never occur as the next-to-last text character in a physical sector, since the EM character following it would be lost.

If the file is organized so that each physical sector contains exactly the same integral number of logical records, with no logical record spanning an EM character, the file is said to be blocked. If the file is not blocked, then it is said to be record compressed. Note that for a blocked file all sectors except possibly the last one in the file contain the same number of logical records while for record compressed files the number of logical records per physical sector is indeterminate.

Under DOS conventions, a valid end of file mark consists of exactly six NUL characters, followed by an EM character:

```
        000 000 000 000 000 000 003
```

This mark must begin at a sector boundary. All information after a valid end of file mark in the sector is indeterminate.

CHAPTER 36. REMAP COMMAND

36.1 Purpose

REMAP allows the logical remapping of disk and diskette controllers, and thus the logical drives of the controllers, on 64K systems with 9320 disks spinning.

DOS.H maintains a logical drive mapping table as a means of locating drives from logical drive numbers. The table maps all logical drive numbers, regardless of whether or not the corresponding mapped drive is on-line, or even exists. The user is urged to be aware of his particular controller and drive configuration.

This table is present only if the DOS was hard-booted with 9320 disks spinning, as this is the only configuration in which logical drives may be remapped. The initial configuration of the map table (after each hard boot) is as follows: Logical drives 0-3 are mapped on 9320 controller 8, drives 4-7 are mapped on controller 9, drives 8-11 are mapped on controller 10, drives 12-13 are mapped on diskette controller 1, and drives 14-15 are mapped on diskette controller 2, if controller 2 is supported by the machine. Note that 9320 controller 11 is not mapped initially.

REMAP deals only with disk and diskette controllers, not with logical drives. The addition and deletion of controllers will, of course, affect the logical drives of the controller.

36.2 Logical Drive Mapping Table Display

The DOS logical drive mapping table is displayed after each change made to it during REMAP. The display shows the logical drive number, disk type, and controller number for logical drives 0-15. An asterisk after the logical drive number indicates that this drive is the booted drive.

36.3 Use

The command line for REMAP is:

```
REMAP[;D]
```

The "D" option simply displays the current mapping table and returns to DOS.

36.3.1 Specific Remapping

The most common use of REMAP will probably be to change the mapping of logical drives 12-15. As noted above, after the initial hard boot, logical drives 12-15 are mapped on diskette controllers 1 and 2. Many users will want to bring the fourth 9320 controller (controller 11) logically on-line to utilize sixteen logical 9320 drives. Conversely, drives 12-15 may need to be remapped onto diskettes if they were previously mapped on 9320 controller 11. Therefore, REMAP first checks the mapping of logical drives 12-15. If these drives are mapped on diskette or on 9320 controller 11, the user is asked if the specific remapping operation, as described above, should take place. If the user responds with a 'Y', the specific remap occurs and REMAP returns to DOS.

36.3.2 General Remapping

If REMAP finds that logical drives 12-15 are not mapped on diskettes or on controller 11, or that one of the drives 12-15 is the booted drive, or the user responded with an 'N' to the specific remapping prompt, REMAP will ask if general remapping is to take place. If the user replies 'N', REMAP returns to DOS. If the reply is 'Y', REMAP displays the mapping table, and the user is free to add and delete controllers at will.

The user is advised to exercise caution in general remapping, as the addition and deletion of controllers will affect the mapping of logical drives. If, for example, an operator assumes that a particular logical drive is on a particular controller, when in fact that logical drive has been remapped onto another controller, confusion may ensue.

In the general remapping phase, REMAP will first display the current logical drive mapping table, then a command may be entered. Valid commands are 'E' or '*' to exit, 'A' to add a controller, and 'D' to delete a controller. If 'A' or 'D' is entered, a controller number is requested. Valid controller numbers are 1-2 for diskette, and 8-11 for 9320 disks. A '*' may be entered here to return to the command entry level. If no error occured in adding or deleting a controller, the logical mapping is displayed again, and another command may be entered. If an error did occur, the error message is displayed, the logical mapping is unchanged, and another command may be entered.

If the execution of a 'delete controller' command would result in the deletion of the booted drive, the user is asked to verify that the booted drive is to be deleted. If the booted drive is deleted, a search to find an on-line drive to become the booted drive is performed before REMAP returns to DOS. If no such drive could be found, the mapping table is restored to its state upon entry to REMAP.

The final display of the mapping table reflects the current mapping, and the current booted drive is flagged with an asterisk.

36.4 Messages

If REMAP is invoked with no 9320 disks spining

```
NO 9320 DISKS ARE RUNNING; LOGICAL DRIVES ARE 0-n
```

will be displayed. The logical mapping in this case cannot be changed, and the diskettes are mapped as logical drives 0 - n, where n is 1 or 3, depending on the number of diskettes supported by the machine.

If REMAP is invoked with a concurrent job active

```
NO REMAPPING WITH CONCURRENCY ACTIVE
```

will be displayed after the logical mapping is displayed. The mapping may not be changed in this case.

If REMAP finds that logical drives 12-15 are on diskette, the mapping of drives 12-15 will be displayed and the prompt

```
MAP LOGICAL DRIVES 12-15 ONTO 9320 CONTROLLER 11 ?
```

will be displayed. This is a specific remap query.

If REMAP finds that logical drives 12-15 are mapped on 9320 controller 11, this fact will be displayed and the other specific remap prompt

```
     MAP LOGICAL DRIVES 12-15 ONTO DISKETTE CONTROLLERS 1 AND 2 ?
```

will be displayed.

If the specific remap phase was bypassed, either by the user responding 'N' to the specific remap prompt, or by REMAP determining that the specific REMAP could not be performed

```
     DO YOU WANT TO PERFORM GENERAL REMAPPING ?
```

will be displayed.

If the general remap phase was taken

```
     ADD CONTROLLER, DELETE CONTROLLER, OR EXIT?
```

will be displayed to request entry of a command.

If a valid command was entered

```
     CONTROLLER NUMBER?
```

will be displayed to request a controller number.

If the booted drive would be deleted by the execution of a delete command

```
     DO YOU WANT TO DELETE THE BOOTED DRIVE ?
```

will be displayed.

36.5 Error Messages

These error messages are displayed when an error was encountered while trying to service a command.

If an attempt was made to add a controller that was already in the table

```
THAT CONTROLLER IS ALREADY MAPPED
```

will be displayed.

If an attempt was made to delete a controller that was not in the table

```
THAT CONTROLLER IS NOT MAPPED
```

will be displayed.

If an attempt was made to add diskette controller 2 on a machine that does not support two diskette controllers

```
THAT CONTROLLER DOES NOT EXIST
```

will be displayed to remind the user that logical drives may not be mapped on a second diskette controller.

If the addition of a controller would result in the mapping of more than 16 logical drives, or if there is not enough contiguous space in the table for all the drives of the controller

```
CAN'T ADD THAT MANY DRIVES
```

will be displayed

If, after the deletion of the booted drive, another drive to become the booted drive cannot be found

```
CAN'T FIND A BOOTED DRIVE. MAP TABLE UNCHANGED
```

will be displayed.

CHAPTER 37. SAPP COMMAND

The SAPP command is used to append two source files, creating a third file.

```
SAPP <file spec>,[<file spec>],<file spec>
```

The SAPP command appends the second source file after the first and puts the results into the third file. If extensions are not supplied, TXT is assumed. The first two files must exist. If the third file does not already exist, a new file will be created. The first file's end of file record is discarded and the copy is terminated by the end of file mark in the second file.

Omitting the second file specification causes the first file to be copied into the third file. One reason for this is to deallocate excess space at the end of the file, since the COPY utility copies by sectors, not records. Neither the first nor second files are changed. Note that the first and third specifications may be the same, in which case the text from the second file is appended to the first file. However, the second and third specifications may not name the same file. If the third file specification is not given, it will be defaulted to the first file specification with the extension /TXT if and only if the second file specification was not given. If the second and third file specifications are the same, the message

```
FILES TWO AND THREE MUST BE DIFFERENT
```

will be displayed.

The first file specification is always required. The third file specification is required if the second file specification is given. If the file specification is omitted in either case, the message

```
NAME REQUIRED
```

will be displayed. If any of the file specifications are not valid file names, the message

```
NAME NOT FOUND
```

will be displayed. If any of the drive specifications are not valid, the message

    INVALID DEVICE

will be displayed.

CHAPTER 38. SORT COMMAND

38.1 Introduction

The Disk Operating System SORT utility enables the user to initiate file sorts directly from the keyboard. SORT will not run while the concurrent job facility is active within a 32K 1500 processor. However, it will run in a 64K 1500 processor at any time.

Using a multi-train radix sort technique, the Datapoint 1500 processor achieves quite reasonable speed. The list of options also compares favorably with much more extensive systems. Since it uses the full dynamic nature of the DOS, it is extremely easy to operate. (Users who have spent several hours figuring out how to set up the myriad of SORT work datasets required by other systems for even the simplest sorts know what we mean.)

For more sophisticated users, SORT may be called from other programs through CHAIN. Using CHAIN also enables complicated sort options to be reduced to a single file name then callable either from the keyboard or another program, such as a DATABUS program. CHAIN also extends the SORT package to operate as a merge, as well.

38.2 General information

SORT attempts to optimize its speed by placing its work files on a 9320 disk separate from the input and output files, or, if no 9320 disks are spinning, on a diskette separate from both the input and output files. Both SORT work files are always placed on the same drive. If SORT selects a drive with insufficient space, it will abort. It may then be necessary to drive-direct its work files.

38.3 Fundamental SORT Concepts

38.3.1 File Formats

All Datapoint systems use a universal text file structure recognized by DATABUS, EDIT, Terminal emulators, etc. Therefore, any text file generated by or for any of the above, may be sorted. The file to be sorted must be on disk, however.

There are two sub-formats a Datapoint file can have: Blocked or Sequential. Blocked files are required to have an equal number of logical records wholly contained within each physical disk record. The maximum record size for blocked records is 249 bytes (plus end-of-record and end-of-sector control bytes for a total of 251 bytes). Sequential records have no fixed relationship to physical disk records and are written as densely as possible in the given file space. Nonetheless, blocked files can be read sequentially in the identical way that sequential files are read. In fact, both types of files, when read sequentially, are indistinguishable. Blocked files are used for achieving random access to records. They generally require more disk space than sequential files for the same amount of data.

Space compression implies that the logical position and the physical position of a character in a record may differ. SORT will always expand the spaces to determine the logical position of a character.

When sorting, consider that the result of the sort is not a restructuring of the original file. It is a NEW file which is a restructured COPY of the original file. The original file is never changed.

Therefore, SORT produces a file which is a sorted version of the original. This gives the user the added opportunity of specifying the type of file to be output regardless of the input file format (with one restriction - see the section on Input/Output File Format Options).

38.3.2 The Key Options

The KEY of a sort is the FIELD or that part of the record which is to ORDER the sequence of records. For instance, it can be a person's name, state, employee number, amount in debt or any aspect of the data base identifiable by a fixed position in the record, based upon the column count from the beginning of the record.

Consider the following record (column count scale below for reference only):

```
     Mule, Francis A.       2422196123 BARN    SAN ANTONIO     TX
     123456789012345678901234567890123456789012345678901234567890
```

The name begins in column 1 and goes to 23. The employee number spans columns 24-30. The street address is 31-42. The city is 43-58. The State is 59-60

If each person had a record in the file exactly in the above format, SORT could order the sequence of records in the file by any of the above fields. For instance, to get an alphabetical list of the records by name, the key would be 1 to 23 (hereafter referred to as 1-23). The key for sequencing the file in order of employee number would be 24-30. The key for ordering the records by state, then city and then employee number would be 59-60,43-58,24-30.

Any portion of the record can be used as a key. Care must be taken when selecting a key to include no more characters than necessary, since each character added to the key slows down the sort.

The key specified for SORT is concatenated to a single string, then sorted character-by-character, with the left-most character being of most significance. It is very important to realize the effect of a right-to-left character sort. To appear in the "right" sequence numeric fields must be right-justified, character fields must be left-justified. If signed numeric fields are sorted, the sign should be moved to the left-most position and the magnitude right-justified; otherwise the resulting sorted sequence will contain positive and negative values in no discernible order, since the "-" and "+" signs are just another character to SORT. A full explanation of character sort concepts is beyond the scope of this manual. Interested users should consult an appropriate information science textbook.

38.3.3 How to Sort a File

Sorting a file is done from the keyboard of the DOS. All the operator must know is the name of the file to be sorted, the name desired for the sorted output file, and the columns containing the key.

For instance, the keyboard-issued command for the above example to sort on the name field (1-23), would be:

```
     SORT EMPLFILE,SORTFILE;1-23
```

This is assuming that the name of that file was EMPLFILE. It is also the operator's decision as to what the resultant sorted file is called, as the command could have easily been:

```
     SORT EMPLFILE,EMPSORT;1-23
```

as well. The second file named is where the resultant sorted output will be placed.

More complicated keys may be stated as well and the command to sort by state and then name would be:

```
     SORT EMPLFILE,SORTFILE;59-60,1-23
```

That is all there is to simplified sorting.

Testing SORT for yourself is simple. Most systems have a source code file for a Databus program on the disk. Such programs can be sorted by op-code and provide an interesting analysis of the usage of each instruction type:

```
     SORT INFILE,OUTFILE;9-12
```

38.4 The Other Options

38.4.1 Generalized Command Statement Format

The following is the generalized statement format for the Datapoint DOS SORT:

```
SORT IN,OUT[,:DRn][,SEQ][;[[F][O][R][H][GNNNTC][N]][K1]...[,On][,Kn]]
```

Information contained within a pair of square brackets

[] is optional; information within brackets is order-dependent. Commas may be used to delimit parameters. (Note that commas MUST be used to delimit sort-key groups.) The first four fields (those ahead of the semi-colon) are considered to be file specification fields. The fields following the semicolon are considered to be SORT key parameters. Default conditions are listed below. Typical statements obeying this format are:

```
     (1)  SORT INFILE,OUTFILE
     (2)  SORT INFILE,OUTFILE;1-3,7-20
     (3)  SORT INFILE,OUTFILE;ID1-3
     (4)  SORT INFILE,OUTFILE;IDL7-20
     (5)  SORT INFILE,OUTFILE;LH11-20
     (6)  SORT INFILE,OUTFILE,,EBCDIC/SEQ
     (7)  SORT INFILE,OUTFILE,:DR0,SEQFILE/SEQ:DR1
```

All the above statements will invoke a sort. Each will provide different results. However, notice that in (1) there are no other parameters than the file specifiers. That is because all the specifiable parameters have a default value in case there is no specification for it.

The following list defines the parameters which can be specified:

IN........This specifies the input file. This file must exist on disk.

OUT.......This specifies the output file. This specification is optional IF AND ONLY IF the 'L' AND 'H' options are used. If an output file is specified AND no drive is specified AND the file exists on a drive on-line to the system then the output file will over-write the existing file. If an output file is specified AND no disk drive is specified AND no file of that name exists on a drive on-line to the system THEN a file of the given name will be created on the same drive as the input file.

:DRn......This specifies the drive for the sort key file. This is only a working scratch file needed during the sort. SORT will attempt to pick the optimum drive on which to put the work file on a multi-drive system. Experience or special considerations may cause the user to want to specify a work drive.

SEQ.......NON-ASCII COLLATING SEQUENCE FILE
This specifies the file which contains the collating sequence to be used. If omitted, ASCII will be assumed.

F.........FORMAT.
This parameter specifies the output file format: blocked or space compressed (standard editor output format). If the user specifies I (and the input file is also blocked), then the output file will be left blocked.

Without typing the 'I', the output file will be record compressed no matter what the input file. If and only if the input file is a blocked file (one logical record per sector), you may include the 'I' parameter and cause the output file to be blocked. If the input file is not blocked, the resultant output file format will be indeterminate.

O.........ORDER.
This parameter specifies the output file collating sequence: Ascending or Descending. The actual character entered is 'A' or 'D'. The default value is 'A'.

Without typing the 'D', the collating sequence order is considered ASCENDING. Including the 'D' parameter will cause the collating sequence to operate in DESCENDING order. Note that if some keys are to be sorted in ascending order and other keys in descending order, the "On" specification described below should precede each key whose order differs from the order of the key preceeding it. However, if all keys are to be ordered in the same sequence, this parameter need only be specified once.

R.........RECORD FORMAT.
This parameter specifies a special output record format: Limited output file format or Tag file or Keytag file output. The actual character entered is 'L' or 'T' or 'K'. The default value is no special output record format; that is, neither 'L' nor 'T' nor 'K', so that the records in the

output file will be exact copies (FULL IMAGE RECORDS) of the records in the input file.

Normally, the sort transfers all of the records of the input file to the output file. It is possible, not only to transfer part of each record, but to select only certain records or to include constant literals in each record as well. Including the 'L' parameter in the list of parameters will cause another question to be asked wherein you may specify the limitations and constants. See the section on Limited Output Format Option.

By entering the 'T' character an output file is generated which consists only of binary record number and buffer byte pointers to the input file records. See the section on Tag File Output Format Option.

By entering the 'K' character a standard text format output file is generated which consists of records containing a 5 byte user logical record number, a 3 byte buffer address, and the key. These records are space-compressed and have trailing spaces truncated. See the section on Keytag File Output Format Option.

H.........HARDCOPY OUTPUT.
This parameter specifies that the output of the SORT will be listed on a printer. The actual character entered is 'H'. The default value is no hardcopy output.

Without typing the 'H' no printing will occur and SORT will require that an output file be named. If the 'H' parameter is given and an output file is named then SORT will list the output to a printer and will generate an output file. If the 'H' parameter is given and no output file is named then SORT will list the output to a printer and no diskette file output will be generated.

If the 'H' and the 'L' parameters are given, the 'L' parameter must precede, the 'H' parameter; 'L' must be used if 'H' is used.

SORT will print to a serial printer. See the section on Hardcopy Output Option.

G.........GROUP INDICATOR
This parameter specifies that the input file consists of Primary and Secondary records and specifies which Group is to be sorted. The actual character entered is 'P' for primary or 'S' for secondary. There is no default value.

If the 'P' or 'S' option is entered then the NNNTC options must follow it.

In a file with Primary and Secondary records, a string of records with a Primary record as the first record and Secondary records following it is considered one block, or group, of records.

When the file is sorted on Primary records the output file has the blocks of records re-ordered so that the Primary records are in the sorted sequence; no change is made in the sequence of the Secondary records following each Primary record. When the file is sorted on Secondary records and the first key specified is in ascending sequence, the output file has the blocks of records in the same order as in the input file, but the Secondary records within each block are in the sorted sequences.

When the file is sorted on Secondary records and the first key specified is in descending sequence, the output file has the blocks of records in reversed order as the input file, but the Secondary records within each block are in the sorted sequence.

SORT has no provision for the sorting of Primary and Secondary records in the same SORT run.
See CHARACTER for example.

NNN.......NUMERIC position of Primary/Secondary flag.
This parameter specifies the character position for the character (the 'C' parameter) indicating whether the record is a Primary or Secondary record . The number must be specified if the option is taken and must fall in the range 1 to 249.
See CHARACTER for example.

T.........TYPE of evaluation.
This parameter specifies equivalence or inequivalence of the group indicator character; that is, whether the character in the record will be equal to or not equal to the character specified. The actual character entered is '=' for equal or '#' for not equal. There is no default character, '=' or '#' must be given if the option is taken.

If '=' is given then if the character in the NNNth position of an input file record is EQUAL to the group indicator character -- indicated by 'C' below -- then the record is a member of the specified sort group -- indicated by 'G' above. Otherwise, it is not a member of the specified group.
See CHARACTER for example.

C.........CHARACTER, group indicator
This parameter specifies the actual test character for determination of a record's membership in the sort group. The actual character entered is any member of the available character set -- this means any combination of eight bits -- except 015. There is no default character. The character immediately following the 'T' parameter is taken to be the 'C' parameter -- except a 015.

This example sums up the above four groups:

```
        ;P240=*
```

N.........This parameter specifies no space compression on output. This applies to Full Image and Limited Output files. It does not apply for blocked or Tag files. If the input file is space-compressed, the 'N' parameter will cause the output file to be non-compressed. If the input file is not space-compressed, the output file will not be compressed, regardless of the N parameter.

K1........SSS-EEE
This is the first sort key specification. If no key is specified, the SORT will assume 1-10,i.e. the first ten characters of the record.
SSS is the starting key position.
EEE is the ending key position. The key is limited to 118 characters and must be contained within the

first 249 characters of the record.

On........This specifies the order for the nth key (ascending and descending are indicated by 'A' or 'D'). If omitted the order used on the previous key is assumed.

Kn........SSS-EEE
The nth sort key specification. The maximum number of keys is that which can be typed without exceeding the input line.

38.4.2 Keys-overlapping and in Backwards Order

The key specification need not be only forward. A specification of 17-12 will cause the 6 delimited characters to be a key but in the order of 17,16,15,14,13,12. This is extremely valuable, clearly, in data which has the most significant digit or character last.

Key specifications may also be overlapping: 1-20,30-15 overlaps 15 to 20. When this occurs, the system will optimize the sort and save time over re-sorting on those columns again.

38.4.3 Collating Sequence File

By specifying a sequence file, the user may substitute any collating sequence for the standard ASCII character set. The sequence file may have any name, but the extension _must_ be /SEQ (SEQ is the default extension).

38.4.4 Ascending and Descending Sequences

Changing the collating sequence from ascending to descending is the same as 'reversing' the file, or placing the last first, etc. Sorting a telephone directory in ascending sequence on name produces the familiar order. Should it be sorted in descending sequence, then Mr. Zyk would be first and Mr. Aardvark would be last. The order of collating (when alphabetic, numeric, and punctuation characters all occur in the same column), follows the character set order. The sequence may be specified for each sort key. However, it need not be specified if it is the same as the key which preceeds it. Therefore, it is possible to sort portions of the key in ascending order and portions in descending order.

38.4.5 Input/output File Format Options

SORT accesses each file sequentially. Due to the techniques used in the Datapoint standard file structure, the sequential reading technique will provide SORT with all of the records in the file whether the file was originally blocked or sequential. Therefore, the file format options only allow specification of the output file's format.

If the input file is blocked, that is one logical record or string per physical disk record, then you have a choice of output formats (F option). If 'I' is chosen, that is blocked, then each output disk record will contain an exact copy of the appropriate input file record. If 'I' is not specified, then the input file, reordered, will be reblocked and appear, generally much more compactly, in the output file in record-compressed sequential format.

If the input file is sequential in its original format, then there is only one choice for the output format; the output file format for a sort on an input file which is sequential must be sequential.

38.4.6 Limited output format option

In many cases, especially when making reports, directories etc. from the data base, it isn't necessary to have the entire record transferred from the input file to the output file during a SORT. For instance, an entire personnel data base can be sorted by name to produce an internal company telephone directory. However, it is obvious that all that is needed is the name and telephone number, NOT all the other payroll information. Therefore, SORT permits transferring only that part of the data base desired.

The following is the generalized statement format for the limited output specification which is entered as a second line of parameters:

```
<(SSS[-EEE]^*^'QQQ')[/(P^NNNTC)]>[,<DUPLICATE OF PRECEEDING>]...
```

For notation's sake, different items within parentheses are separated by ^. Only one item within a pair of parentheses may be specified. Items within square brackets [] are optional and items within corner brackets <> may be repeated and must be separated by commas.

The following list defines the parameters which can be specified:

SSS.......STARTING position within input record.
EEE.......ENDING position within input record.
These parameters specify the character positions within the input record to be copied to the output record. The EEE specification is optional; if it is not specified then only one character, the character at SSS, will be copied from the input record to the output record. The SSS and EEE options must fall in the range 1 to 249.

*.........ASCII TAG output.
This parameter specifies that an ASCII pointer to the input record appear in the output record. The ASCII pointer points to the input file logical record number and the byte in that physical diskette record containing the first byte of the input file logical record. If the 'I' parameter was specified in the SORT options then, since the byte in the physical diskette record containing the first byte of the input file logical record will always be '1', the '1' will not appear. The ASCII pointer is a DATASHARE compatible, leading-zero and space-compressed ASCII number. The number of digits for the logical record number pointer is five; the largest number that can be represented is 65,535. The number of digits for the byte pointer (if it is generated; that is, the 'I' parameter was not specified) is three; the largest number that can be represented is 250.

QQQ.......QUOTED character string.
This parameter specifies an actual string of quoted characters that is to be copied into the output record. The quoting symbol is the single quote ' mark. The string may include any characters except the ' mark itself and 015, and must be less than 90 characters long.

P.........PRIMARY record to be source.
This parameter specifies that the information specified by the prior set of START/END positions is to be extracted from the primary record for the current record block, rather than the present (secondary) record. This parameter has no effect when an output record is being generated from a

primary record.

NNN.......NUMERIC position of evaluation character.
This parameter specifies the character position for the character (the 'C' parameter below) indicating whether the information specified by the prior set of START/END positions is to be copied from the input record to the output record. The number must fall in the range 1 to 249.

T.........TYPE of evaluation.
This parameter specifies the equivalence or inequivalence of the evaluation character; that is, whether the character in the input record should be EQUAL to or NOT EQUAL to the evaluation charater. The actual character entered is '=' for equal or '#' for not equal. If the evaluation is satisfied, then the information specified by the prior set of START/END positions will be copied to the output record.

C.........CHARACTER, record evaluation.
This parameter specifies the actual test character for record evaluation. The actual character entered is any character except 015.

In the same manner that the key of the records is specified by fixed column number, i.e. 1-10 for the first ten characters, the limited output feature specifies that part of the records are to be transferred. Should the response 1-10 be given to the limited output format request, only the first ten characters of each record will be transferred to the output file. The limited output format specifier operates in the same manner as the specification of multiple discontiguous sort key fields. For instance, 1-10,50-70 would transfer thirty-one characters from each record of the input file to the output file. The eleventh character in the output record would be the fiftieth character of the input record, etc.

To invoke the limited output format option, the operator includes the 'L' parameter in the specifier list. If and only if the L is specified during the SORT call, will there be a second question asked of the operator on the next line:

```
LIMITED OUTPUT FILE FORMAT:
```

This question requires at least one field specification or constant (see next paragraph). The number of field and constant specifications is only limited by that which can fit on the keyed in line.

To permit even more utility in report generation, SORT allows inclusion of constants in the output record that didn't occur in the input record. For instance, assume that the personnel data base was a full record of about 240 characters and that the employee's name appears in columns 80 to 110 and his telephone number was in columns 171 to 180. To make a telephone directory in alphabetical order, one could answer the following to the limited file output format request:

```
80-110,' - ',171-180
```

Note that this would put out the name followed by one space, a hyphen, one more space and the number. Any number of input file fields and constants can be placed in the output file up to the limit of the line on which the specification is typed.

Often not every record of the input file is needed in the output file. Limited output allows selection of records from the input file, based on character evaluation on one character position. For example, if a primary/secondary file is being sorted and only the primary records are desired in the output file, the command could appear as:

```
SORT INFILE,OUTFILE;LP1=*,2-10
LIMITED OUTPUT FILE FORMAT:
1-85/1=*
```

Columns 1-85 of the input record will be written to the output file if column one is an *.

Limited output can be used to make more complex selections. If it is desired to output records containing a 0 in column 5 OR a 1 in column 6, the command would be:

```
SORT INFILE,OUTFILE;L5-8,12-15
LIMITED OUTPUT FILE FORMAT:
1-85/5=0,1-85/6=1
```

To output records containing a 0 in column 5 AND a 1 in column 6 would require two SORTs, the first using a limited output to test column 5 and using only a 1-character key, to make the SORT as fast as possible. The second SORT would use a limited output testing column 6 and would be given a sort key to correctly order the output file.

There is no relationship between the primary / secondary specification on the command line and the conditional output specification on the limited output format line.

Also note that the output file requires proportionately less room than the input file when limited. Often this fact can be put to use when the disk file space is nearly exhausted and a SORT is required.

38.4.7 TAG file output format option

For some applications it is useful to have a data file sorted into several different sequences. However, to have several copies of a file on diskette merely to have it in different sequences consumes much of diskette space, and indeed if the file is a very large file many copies of it may not fit onto even two diskettes.

This problem could be avoided if there were a way to index into the one main file in any of several different sequences. The index pointers could exist as a file, and the index entry for each record in the main file would only have to be three bytes long -- two bytes for the LRN (Logical Record Number) and one byte for the BUFPTR (Buffer Pointer -- a pointer to the beginning of the actual desired record within the diskette physical buffer).

SORT provides for the generation of such an indexing file, a TAG file, by the 'T' variation of the 'R' option. A TAG file may be generated for either a sequential or blocked file, and will have the same format for either file. The format of a TAG file is simple:

1. For each record in the input file, the TAG file will have a three byte binary pointer to the first byte of the record.

2. The format of the pointer is:
   Byte 1: MSPLRN (Most Significant Portion of LRN),
   Byte 2: LSPLRN (Least Significant Portion of LRN),
   Byte 3: BUFPTR (Buffer Pointer).

3. The three-byte binary pointers are blocked 83 to a physical diskette record.

4. The Physical-End-Of-Record mark is an 003 and the rest 000's.

5. The End Of File mark is: beginning at the first byte in the physical record, six 000's, one 003, and the rest 000's.

TAG files may be used by other programs and utilities (such as INDEX) as Record Address files).

If the TAG file option is specified then the LIMITED OUTPUT FILE FORMAT or the HARDCOPY OUTPUT can NOT be specified.

If a TAG file is generated when the 'P' (PRIMARY SORT) option is specified then TAG file pointers will be generated only to the PRIMARY records in the input file.

If a TAG file is generated when the 'S' (SECONDARY SORT) option is specified then TAG file pointers will be generated that point to each PRIMARY record of the input file (in their original sequence) each primary tag being followed by pointers to the SECONDARY records in the record block <u>in their sorted sequence</u>.

When a TAG file is generated for 'P' or 'S' sorts, no indication is given in the TAG file pointer as to whether the pointer points to a primary or a secondary record; it is up to the user's program to check the records in the indexed file to determine when a record block begins or ends.

38.4.8 KEYTAG File Output Format Option

Requesting a Keytag file output will cause a file (default extension "TXT") to be created. This EDIT-compatible text file contains the record pointers and the key. The record pointers (first 8 bytes of the record) consist of a 5 byte logical record number (range 0 to 65,535) and a 3 byte buffer address. The record number is the user logical record number, that is, zero points to the first data sector.

If a sequence file (e.g., EBCDIC/SEQ) is used, the key produced by this option will be translated to that sequence. If the untranslated key is desired, a Keytag file may be created by requesting ASCII TAG output from the Limited Output Format Option.

38.4.9 Hardcopy output option

Many times it is desired to have a hardcopy printed output from a SORT instead of or in addition to the creation of a diskette output file. This can be easily accomplished with SORT by specifying the 'H' (HARDCOPY) option along with the 'L' (LIMITED OUTPUT STRING) option. The 'H' option is essentially an expansion of the 'L' option because disk data files are almost never suitable for full image output to a printer; decimal points need to be inserted into dollar and cents amounts, dashes need to be inserted into part numbers, and spaces need to be placed between dollar amounts and part numbers to columnate the data, and so on. If it is desired to list output records in full image format, it is only neccessary to give:

```
1 - n
```

(where n is the maximum printable character on the printer) as the limited output string specification.

Sort will not send a line of over 132 characters to a printer. If the limited output specification designates a longer output record, then the full specified formatting will be applied to the diskette output file (if any), but only the first 132 characters of the record will be printed.

If the following special characters are imbedded in the output record, they will be interpreted as indicated:

```
015 = End-Of-Record and Carriage-Return/Line Feed.
012 = Line Feed.
014 = Form Feed.
```

Note that if the printer goes off-line for any reason while printing the hardcopy output, SORT will discontinue the printing totally. If no output file had been specified along with the hardcopy option, and the printer goes off-line, SORT will abort since no output will be produced.

38.4.10 Primary/Secondary sorting considerations

If the 'P' (PRIMARY) or 'S' (SECONDARY) SORT option is used, then the input file must have a PSPSPS.... format in order for SORT to work as expected, where P is one primary record and S is one or more secondary records. The first record in the file should always be a primary record, and the last record should be a secondary record. There should always be at least one secondary record following each primary record. Tertiary and further level records cannot be accommodated by SORT.

In some cases it may be possible to successfully sort a file using the 'P' or 'S' options even if the file does not faithfully follow the above rules. However, the user must exercise great caution if he is to successfully "fudge" a system as complex as SORT. Pitfalls are many. For example, if a file has the format PPPPSPSPS..., and a sort is done using the 'S' option, the output file will probably not contain the first three primary records at all. This case occurs because when sorting using the 'S' option, pointers are generated for only the secondary records, prefixed by a pointer to the record preceeding the first secondary record of a record block. Since no secondary pointers were ever generated for the first three primary records, they are simply lost. It should be easy to imagine what would happen to a file if a tertiary sort were attempted.

38.4.11 Key File Drive Number

There are three file systems associated with a SORT. The first is, of course, the input file. The second is the output file. The third is the keyfile system. (The user only uses the output file--the keyfile system is a scratch file used by the system during sorting). There are actually two files which get opened during the sort for the keyfile system. They are *SORTKEY/SYS and *SORTMRG/SYS. These two files can grow to considerable sizes during the sorting procedure since they are proportional to the number of records and the size of the key field.

There are two considerations for the location of the keyfile system. The first is the problem of room. The keyfile must be on a drive with sufficient room to hold it. The second is speed. The greatest increase in speed occurs in removing the keyfile system from the same drive as the input file. Greater speeds can occur if it is, as well, not on the same drive as the output file. Normally the SORT does a pretty good job of determining the best location of the two keyfile files and it shouldn't be necessary to specify anything for this. However, under complex circumstances, it may be desirable for the operator to specify the drive number for the keyfile. Should this be the case, the user should type in the <:DRk> specification as indicated in the general command format in the Generalized Command Statement Format section.

38.4.12 Disk space requirements

A formula for determining the room in physical disk records that will be required for the SORT work files is:

$$R = \frac{2N(L+P+3)}{S} + 4T$$

where: R = Room in physical disk records (sectors) required on disk.
N = Number of logical records in input file for which keys will be generated:
= number of records in file if not sorting on 'P' or 'S'.
= number of primary records in file if sorting on 'P'.
= number of secondary records in file if sorting on 'S'.
L = Length of the sort key in bytes.
P = 3 if sorting on secondary records,
0 if not sorting on secondary records.
T = number of sort key trains.
S = bytes per block of physical space available to the user (nominally 253 bytes)

The value of T can be computed approximately, as:

$$T = \frac{N(L+P+3)}{5700}$$

38.5 The use of CHAIN with SORT

The reader should first familiarize himself with CHAIN by thoroughly reading the CHAIN Section.

CHAIN is a system whereby the operator of a Datapoint DOS may pre-define a procedural sequence of his own programs, system commands and utilities (including keyboard answers to questions requested by these programs) and have them automatically executed after being invoked by a single command. This feature is especially powerful when using SORT since there may be a repetitive sequence of routines with complex parameterizations. Little room for operator error remains when only one command is required to kick off many jobs.

A DATABUS program can link to SORT by executing a ROLLOUT instruction to a user-built CHAIN file which includes the SORT command line and, if specified, the Limited Output specification line and a Hardcopy Heading line, followed by the DATABUS command line with 'R' option (DB15;R) to re-load the DATABUS Interpreter.

38.5.1 How to Set up a chain file for SORT

The author of a chain file only needs to remember that ALL questions that the system requests INCLUDING those initiated by the executing programs MUST BE ANSWERED from the chain file just as though they would be typed in from the keyboard.

For instance, the initiation of a sort

```
"SORT INFILE,OUTFILE;I3-42"
```

could be done through chain. To do this, use EDIT or BUILD to type in that exact sequence of characters into a file. Note that the file will, in this case, consist of a single line as typed above. The file can be any name, but for purposes of simplifying the explanation, it shall be referred to as "CHAINFIL". If "CHAINFIL" consists of that single line, and if the operator types the command "CHAIN CHAINFIL" to the DOS, the SORT specified above would be initiated. If the 'L' specification were included in the statement above, then SORT would ask for another line of information. In this case, the file "CHAINFIL" would have to have two lines in it with the first being the SORT command and the second being the limited output file format specification.

38.5.2 Naming a repetitive SORT procedure

Frequently there are sorts and printouts and other procedures which occur together, and for which a single command to invoke the procedure would be a great simplification.

For instance, in the telephone directory example above, the process of sorting the file into a limited output file and then listing it on the printer could be procedurized as follows:

```
SORT EMPFILE,TELFILE;L80-110
80-110,'  - ',171-180
LIST TELFILE;XL
TELEPHONE DIRECTORY FOR XXXXXXXXXX CORPORATION
```

Note that there are four statements. The first is the SORT command. The second is the answer to the limited format initiated by the 'L' in the SORT command. The third is the DOS LIST command with the specifiers of 'X' which says 'without line numbers' and the 'L' which means the printer. Then there is a fourth line which the LIST command requests--the heading. This question must also be answered in the chain file. If the above four statements were placed in a file by the Editor (or by any other means) and then CHAIN were invoked with that file specified, the result would be a printed telephone directory from the personnel files.

38.5.3 Using CHAIN to cause a merge

Consider a situation wherein a system has a master file called 'MASTER' and a file of records to be added, in sequence, to the master file called 'ADDFILE'. To merge these two files in sorted sequence at the end of each day would normally require a sequence of keyed in operations which are somewhat complicated and error prone. CHAIN can cause an effective MERGE and assign it a single name as follows:

```
SAPP MASTER,ADDFILE,MASTER
SORT MASTER,SCRATCH;1-20
KILL MASTER/TXT
Y
NAME SCRATCH/TXT,MASTER/TXT
```

Note that the procedure:
1) appends the ADDFILE to the MASTER file.
2) Sorts the extended MASTER file into a SCRATCH file.
3-5) Renames the SCRATCH file as the new MASTER file. Thus, it is apparent that a merge can be effectively achieved using SORT and by using chain to pre-define the procedure.

38.6 SORT Execution-Time Messages

This section describes the operator messages that SORT may display on the CRT screen during execution. Some of the messages are monitor messages to keep the operator informed of the progress of the program, while other messages are error messages.

DOS. VER. n.n SORT COMMAND - date

This message is the SORT sign-on.

SORT OVERLAY MISSING.

This message is displayed if the SORT/OV1 file is not on the same drive as the SORT/CMD file.

SORT CAN NOT EXECUTE IN 32K WHEN JOB15 IS ACTIVE

This message is displayed if an attempt to execute SORT in a 32K 1500 processor is made when JOB15 is already active.

INPUT FILE REQUIRED.

This message is displayed if no filename was specified for the first file specification. This would happen if a command line such as:

```
        SORT ,OUTFILE      or      SORT /TXT,OUTFILE
```

were entered.

OUTPUT FILE REQUIRED.

This message is displayed if no filename was specified for the second file specification AND if the 'L' and 'H' options were not specified.

BAD DEVICE SPECIFICATION.

This message is displayed if a drive specification in a file specification was not entered in a valid format.

OUTPUT FILE SAME AS INPUT.

This message is displayed if the filename and extention of the INPUT file and the OUTPUT file are the same, and the DRIVE NUMBER for each file is the same or not specified for EACH file.

INPUT FILE NOT FOUND.

This message is displayed if the INPUT file could not be found on any drive on-line to the system if no drive was specified, or on the drive given if a drive was specified. If no extension is supplied in the file specification an extension of TXT will be assumed; in this case if a file FILENAME/TXT is not on-line or on the drive specified then the INPUT file will not be found.

INPUT FILE RIB ERROR.

This message is displayed if a read parity error occurs when the INPUT file's RIB is checked to determine the INPUT file's length.

KEY FILE SPECIFICATION ERROR.

This message is displayed if a filename or extension is given for the KEY DRIVE specification.

KEY FILE DEVICE SPECIFICATION ERROR.

This message is displayed if the drive specification for the KEY file is not a valid drive specification.

SORT KEY FILE PLACED ON DRIVE #

This message is displayed if the KEY DRIVE was not specified on a multi-drive system. The message is to notify the operator of the location of the KEY file. The # stands for a valid drive number.

OPTION FIELD ERROR.

This message is displayed if a semicolon (;) is entered at the end of the SORT command line but is not followed by any option specifications.

OPTION SPECIFICATION DUPLICATION.

This message is displayed if a command line such as:

```
SORT INFILE,OUTFILE;DLA
```

were entered. The 'D' and 'A' options are both variations of the ORDER option, and obviously both cannot occur simultaneously.

HARDCOPY ONLY IF LIMITED OUTPUT SPECIFIED.

This message is displayed if the 'H' option is specified but the 'L' option was not given previously.

PRINTER NOT AVAILABLE!

This message is displayed if the 'L' and 'H' options have been specified but the printer is currently offline or in use by the concurrent partition. The message will flash until the printer is available and will then go away. Depressing the keyboard key will cause an exit from SORT.

ILLEGAL HEADER SPECIFICATION.

This message is displayed if the 'P' or 'S' option is given but is immediately followed by the 015 byte -- the "ENTER" key.

ILLEGAL HEADER KEY EVALUATION.

This message is displayed if the character immediately following the 'PNNN' or 'SNNN' option is not '=' or '#'.

ILLEGAL SORT KEY SPECIFICATION.

This message is displayed if a key position of 0 or greater than 249 was specified, or if a key position was not terminated by , or - or 015, or if a two-position key was not terminated by , or 015.

SORT KEY TOO LONG.

This message is displayed if the total sort key is longer than 118 characters long.

OVERLAPPING SORT KEY SPECIFICATIONS---SORT OPTIMIZED.

This message is displayed if the same record positions were specified for more than one sort key group. SORT does not repeat duplicate positions in sort key generation and thus saves processing and disk read/write time.

OVERLAPPING SORT AND HEADER KEYS---SORT OPTIMIZED.

This message is displayed if the same record position is specified as a sort key position and a header indication position. The position is removed as a sort key position and the key is thus shortened. The effect is as for the previous message.

LIMITED OUTPUT FILE FORMAT:

This message is displayed if SORT has accepted the SORT command line including all option specifications and if the 'L' option has been given. The operator must enter the limited output specification line.

NULL LIMITATION SPECIFICATION.

This message is displayed if the 'L' option was given but the limitation specification was only 015 -- the "ENTER" key. If the 'L' option is given then a non-empty limited output specification string must also be given.

INVALID LIMITATION SPECIFICATION.

This message is displayed if the limited output specification does not fit the syntax given in the section on Limited Output Format Option. Usually the fault is that a comma was not placed between option specification groups, or double quotes (") were used instead of single quotes (') .

ENTER THE HARDCOPY HEADING:

This message is displayed when the limited output specification has been accepted and if the 'H' option was given. The operator must enter from 0 to 79 characters of information which will be printed at the top of each page printed during SORT output generation.

SEQUENCE FILE NAME REQUIRED

This message is displayed when the sequence file field is blank and the file specification fields have not been terminated with a semi-colon or an end of line designator.

SEQUENCE FILE NOT FOUND

This message is displayed when SORT requests the sequence file be OPENed and DOS cannot locate the file on the disk drive indicated. Note that if the drive is not specified, the drive on which the SORT/CMD resides is implied.

SEQUENCE FILE FORMAT ERROR A

This message is displayed when SORT determines that the sequence file specified is not an absolute object file.

SEQUENCE FILE FORMAT ERROR n

This message is displayed when SORT receives an error return when an attempt is made to load the sequence file. The value of n may be 0-6 and is defined as follows:

0	If file does not exist
1	If disk drive is off-line
2	If directory parity error
3	If RIB parity fault
4	If file parity fault
5	If off end of physical file
6	If record of illegal format

LIMITATION SPECIFICATION OVERFLOW

This message indicates that limited output parameters entered require more memory than the 256 bytes allocated by SORT.

INTERNAL ERROR -- GET SYSTEM HELP !!!

This message indicates a probable hardware error occurred during a limited output string sort. SORT cannot continue executing.

PRINTER OFFLINE - PRINTING ABORTED

This message indicates the printer went off-line during the hardcopy output phase. SORT will not continue printing.

NO OUTPUT SPECIFIED - SORT ABORTED

This message will only be displayed after the PRINTER OFFLINE message. Since printing was aborted, and no output file was generated, there will be no output from SORT. Therefore, SORT will not continue.

<u>THE FOLLOWING MESSAGES MAY BE DISPLAYED DURING SORT INITIALIZATION IF SORT WERE LINKED TO BY ANOTHER PROGRAM</u>. These messages result from internal errors if they occur while running standard Datapoint-supplied 1500 software.

INVALID LIMITATION STRING ADDRESS.

INVALID HARDCOPY HEADING STRING ADDRESS.

INVALID USER EXIT ADDRESS.

LFT ENTRIES 1->3 NOT CLOSED WHEN SORT ENTERED.

LIMITATION STRING MISSING.

HARDCOPY HEADING STRING MISSING.

The following messages are displayed afer the SORT initialization is completed:

BUILDING SORT KEY TRAIN n.

This message is displayed when all parameter specifications have been accepted and SORT has started the extraction of the sort keys from records of the INPUT file and is writing them to the *SORTKEY/SYS file.

SORT KEY FILE OVERFLOW.

This message is displayed if there was not adequate room on the KEY DRIVE to hold the *SORTKEY/SYS file. If *SORTKEY/SYS file overflow occurs the file is deleted from the disk before the message is displayed.

NULL OUTPUT FILE.

This message is displayed if no sort key records were generated. A null output file (first record EOF) is prepared before SORT ends.

INTERMEDIATE SORT PASS n.

This message is generated during sorting of the sort key trains on the *SORTKEY/SYS file. The only actual sorting done during a sort is that which can be done on the initial sort key trains, which are made short enough that they will fit in memory. After the sorting of the keys within each initial train, the trains are merged sixteen abreast into larger trains, repeatedly until only one train remains.

INTERMEDIATE MERGE PASS n, TRAIN n.

This message is displayed if more than sixteen sort key trains exist during a merge pass. The intermediate merge pass number is the Nth iteration of the merge process. The train number is the number of the train being output by the merge pass. If more than one train is output by an intermediate merge pass then at least one more intermediate merge pass will be required. If more than sixteen trains are output by an intermediate merge pass then at least two more intermediate merge passes will be required, and so on.

FINAL MERGE: SORT TRAIN n.

This message is displayed during the generation of the output file from the data in the now fully sorted and merged sort key file and from the records in the INPUT file. The sort train number corresponds to the current state of progress as measured against the number of trains generated by the next to the last intermediate merge pass.

MERGE FILE OVERFLOW

This message indicates not enough disk space is available for the merge file.

OUTPUT FILE OVERFLOW

This message indicates not enough disk space is available for the output file.

CHAPTER 39. SUR COMMAND

39.1 Purpose

When a specific disk is used for more than one purpose, some inconveniences occasionally turn up. Assume for a moment that a user has a disk which is being used for program generation on each of two more or less unrelated projects. When he uses the CAT command, for instance, he will normally see a whole range of files, some of which are not related to the project in which he is currently interested. At times like this, it would be convenient to logically partition the directory so that a user would only have a portion of it, that portion which contains the files with which he is currently working, available to him at one time.

39.2 About Subdirectories

The use of the SUR (Subdirectory Utility Routine) command allows the user to logically partition the directory on a given disk into several smaller subdirectories. Each such subdirectory can then contain zero or more files, up to the combined maximum of 256 files per logical drive. Each subdirectory on a disk has a unique name. Two subdirectories always exist on all drives; these are called SYSTEM and MAIN. The names for the other subdirectories are assigned by the user as he establishes them, and follow the same rules as for any standard DOS file name. As a subdirectory is created, the name specified by the user is related to a unique number which is referred to as the subdirectory number. The relationship between subdirectory names and subdirectory numbers is not unlike the relationship between DOS file names and physical file numbers. A given subdirectory may have different numbers on different drives, even though the subdirectory name is the same.

Up to thirty-one subdirectories may exist on a single disk, including the two "fixed" subdirectories, SYSTEM and MAIN. Thus the user is allowed to create up to twenty-nine additional subdirectories.

It is important to realize that subdirectories are not a way of getting more than 256 files on a drive. This they cannot do. The thing that subdirectories are good for is partitioning the directory and restricting the scope of a file name. This allows several files of the same name to exist on one disk at the same time, without causing the DOS to become confused as to which is the one to be referenced at any time. The way the DOS achieves this is that each of the files is in a "different subdirectory", and hence is uniquely identified even though the name and extension may be identical.

39.2.1 Creation of Subdirectories

Subdirectories are created with the SUR command. All that is required is to specify a name for the proposed subdirectory and request its creation. Creation of a subdirectory does not actually result in any real change to the directory on disk at all. All it does is to cause the specified name to be entered into a table in SYSTEM7/SYS (yes, that's why SYSTEM7 isn't write protected), kept on disk, which relates each subdirectory name with its subdirectory number. The user is allowed to specify which drive on which he wishes to create the subdirectory; if he does not indicate a specific drive, the named subdirectory is placed onto all on-line drives, if possible.

39.2.2 Deletion of Subdirectories

Subdirectories are deleted with the SUR command. The user specifies the name of the subdirectory he wishes to remove and requests its deletion. Deletion of a subdirectory does not result in KILLing the files within the range of that subdirectory. If a subdirectory to be deleted contains one or more files, the files are first moved from that subdirectory to the one called MAIN before the named subdirectory is deleted. The user is allowed to specify from which drives the subdirectory is to be deleted; if he does not indicate a specific drive, the named subdirectory is deleted from all on-line drives on which it appears.

39.2.3 Being "in a Subdirectory"

The user can define at any time which of the subdirectories on each of his disks contain the current files he is interested in. This is done with the SUR command by specifying the name of the subdirectory containing the files of current interest. This action causes him to be placed "into" the named subdirectory on the drive specified. (If no specific drive is mentioned, he will be placed "into" the subdirectory specified on all on-line drives containing a subdirectory with the given name). It is appropriate to point out that the current subdirectory on each drive need not have the same name; for example, the user could easily be in subdirectory PROGRAMS on drive zero and in subdirectory DATABASE on drive one at the same time.

Once in a specific subdirectory on a drive, that state does not normally change until the user requests being placed into a different subdirectory (again via the SUR command) or re-boots the DOS. Rebooting the DOS causes the user to be placed into the subdirectory named SYSTEM on all drives.

39.2.4 Scope of a File Name

When a program accesses a file under DOS, it tells DOS the name and extension of the file it is looking for and either indicates one specific drive which the DOS is to search for the file, or requests that the DOS look on all on-line drives. In order for the DOS to "find" the given file, the DOS must find a file whose name and extension exactly match the ones specified by the requesting program. If no such file can be found, the DOS returns indicating that the specified file cannot be found and therefore probably does not exist.

When subdirectories are in use, this matching of name and extension is expanded so that in addition to a file's name and extension matching those specified by the requesting program, the file must also be within either the current subdirectory (for that drive) or the one called SYSTEM in order to be "found".

Therefore the scope of a file name can be more or less defined via the following: when a user is in subdirectory X on drive Y, files can be "seen" by his program only if they are in either subdirectory X or subdirectory SYSTEM. Files in any other subdirectory will not appear to exist.

39.2.5 About Subdirectory SYSTEM

It has been shown that files in the subdirectory named SYSTEM are special in that they can be accessed regardless of which subdirectory the user is "in" on a specific drive. Likewise, a special situation also occurs when the user is "in" the subdirectory named SYSTEM. When the subdirectory named SYSTEM is the current subdirectory on a given drive, all files on that drive are accessible regardless of which subdirectory they themselves are actually in.

A little caution must be used when a user is in subdirectory SYSTEM on a disk with multiple files of the same name and extension. The caution is that, although each of the files is still associated with one and only one subdirectory, all of the files on a disk are available when the user is "in" the SYSTEM subdirectory. The result, in this situation, is that one of the files of the desired name and extension will be referenced; which one is referenced is, however, undefined. Therefore, good practice dictates that if a user has more than one file with the same name and extension on some drive, that he make a point of always knowing which subdirectory he is in (and that it is not SYSTEM) if it matters to him which of his files of the same name he references.

39.2.6 Files vs. the User Being "in a Subdirectory"

It is important not to confuse the two distinct concepts of a file being in a subdirectory as opposed to that of [a user] "being in a subdirectory".

A file being in a specific subdirectory is a way of saying that the file cannot be accessed when the current subdirectory is neither that specific subdirectory nor SYSTEM. This relationship, that of a file being in a specific subdirectory, is retained more or less permanently; if a file is placed in subdirectory SUBDIR1 today on a disk, the disk can be removed and stored on a shelf; if tomorrow the disk is taken down from the shelf and re-mounted, that file will still be in subdirectory SUBDIR1.

A user being in a specific subdirectory is a way of saying that the subdirectory in question is "the current subdirectory" on one or more logical drives. The "current subdirectory" on a drive is less permanent and reflects the use of the SUR command since the previous time the DOS was loaded.

39.2.7 Getting a File into a Subdirectory

In general, there are three ways to get a file into a given subdirectory. The easiest and probably most common of these is automatic. Whenever a file is created, it is always placed into the current subdirectory on the drive on which it is created.

Once a file has been thus created, it can be moved between subdirectories with the NAME command. The NAME command can take a file within the scope of the current subdirectory and put it into the current subdirectory if it is not already (which is useful if either the source or destination subdirectory is SYSTEM) or can place it into any other subdirectory the user might wish to put it into.

39.3 Usage

The SUR command is parameterized as follows:

```
SUR [<name>][/<function>][:DR<n>][,<new name>]
```

The function performed by SUR is determined by the absence or value of the <function> field and the name field, as described below.

39.3.1 Establishing a "Current Subdirectory"

If the function field is not given, SUR establishes the named subdirectory as the current subdirectory on all drives on which the named subdirectory exists. If the named subdirectory does not exist on one or more drives, the current subdirectory on any such drives is unaffected. If a specific drive is mentioned, then only the current subdirectory on the specified drive is subject to change.

39.3.2 Creating a Subdirectory

If the function field is /NEW, SUR creates the named subdirectory on all drives on which the named subdirectory does not exist. The current subdirectory is not affected by the operation. If a specific drive is mentioned, then the named subdirectory is only created on the specified drive.

39.3.3 Deleting a Subdirectory

If the function field is /DEL, SUR deletes the named subdirectory on any drives on which the named subdirectory exists. If any files are in the named subdirectory, they are moved to subdirectory MAIN before the named subdirectory is deleted. If the subdirectory being deleted is the current subdirectory on that drive, the current subdirectory is also changed to MAIN. Subdirectories SYSTEM and MAIN cannot be deleted. If a specific drive is mentioned, then the named subdirectory is only deleted from the specified drive.

39.3.4 Renaming a Subdirectory

If the function field is /REN, SUR renames the named subdirectory on any drives on which the named subdirectory exists, to the name specified in the new subdirectory name field. If any files are in the named subdirectory, they will be in the subdirectory specified by the new subdirectory name field upon completion of the operation. Subdirectories SYSTEM and MAIN cannot be renamed. If a specific drive is mentioned, then the name of the named subdirectory is changed only on that specified drive.

39.3.5 Displaying Subdirectories

If the subdirectory name field is not given, SUR displays the names of all subdirectories on all on-line drives. The format of the listing is similar to that provided for file names by the CAT command. The number in parentheses to the right of each subdirectory name is the subdirectory number associated with that name (in octal); an asterisk indicates the current subdirectory on each drive. If a specific drive is mentioned, then only the subdirectories present on the specified drive are displayed.

39.3.6 Error Messages

If the drive specification entered on the command line is not within the range of valid drives, the message

```
INVALID DEVICE.
```

will be displayed. If the function field is /REN and the new name is not entered on the command line, the message

```
NAME REQUIRED.
```

will be displayed. If the subdirectory name does not exist, the message

```
- DRIVE X HAS NO SUBDIRECTORY.
```

will be displayed with the drive number (X) displayed in the message. If a specified subdirectory does not exist, the message

```
- NOT IN DRIVE X's SUBDIRECTORY.
```

will be displayed with the drive number (X) displayed in the message. If an attempt is made to create an already existing subdirectory, the message

- ALREADY IN DRIVE X's SUBDIRECTORY.

will be displayed with the drive number (X) displayed in the message. If an attempt to create more than thirty-one (31) subdirectories is made, the message

- DRIVE X's SUBDIRECTORY IS FULL.

will be displayed with the drive number (X) displayed in the message. If an attempt is made to read or write to the directory sector and errors are encountered, the message

    * SYSTEM DATA FAILURE ON DRIVE X *

will be displayed with the drive number (X) displayed in the message. If the function field entered is not /NEW, /REN, or /DEL, the message

    * INVALID EXTENSION *

will be displayed. If an attempt is made to delete subdirectories "SYSTEM" or "MAIN", the message

    * MAY NOT BE DELETED *

will be displayed. If an attempt is made to rename subdirectories "SYSTEM" or "MAIN", the message

    * MAY NOT BE RENAMED *

will be displayed. If a specified drive is off line, the message

    * DRIVE OFF LINE *

will be displayed.

CHAPTER 40. UTILITY/SYS

All of the standard DOS commands are available as separate programs on disk. However, many of them may also be obtained in an absolute library named "UTILITY/SYS". This has the following advantages:

1. Free up many directory entries and some data space.
2. Makes most of the utility programs available on any disk, i.e., UTILITY/SYS can be on any drive on-line.
3. Makes upgrading the utilities easier, since only one file need be replaced, rather than many separate command files.

To display the members in UTILITY/SYS, enter:

```
CAT *
```

Note: the CAT command also displays the directory of any library (see CAT command).

When keyboard commands are entered, the specified command will automatically be located as either a separate disk file or a member of UTILITY/SYS. Normally a separate file name is first checked, then the library member. To reverse the normal precedence put a leading * or : in front of the command name. For example:

```
*CHANGE SCRATCH/TXT;X
     or
:CHANGE SCRATCH/TXT;X
```

See the section on the Command Interpreter for details on selection of a command from the disk directory or from UTILITY/SYS.

Members of UTILITY/SYS may be extracted to individual command files by use of the LIBSYS15 program. These individual command files can be placed as needed on disks that may not have enough room for all of UTILITY/SYS.

CHAPTER 41. SYSTEM DESCRIPTION

41.1 System Philosophy

The objective of DOS.H is to allow maximum use of the capabilities of a Datapoint disk system with a minimum of effort. The DOS.H disk structure provides dynamic space allocation and full random file access capability. Also provided are an extensive set of utility programs to perform many basic data processing functions. In all system utilities, the operator commands are as simple as possible while providing a versatile program capability. Error codes and program messages are mostly presented in standard English, avoiding complex, incomprehensible messages.

DOS.H is not a supervisory system; it imposes practically no overhead. The DOS.H facilities provide a base for DATABUS, DATAFORM, and many other Datapoint systems.

41.2 System Features

To minimize the space used by infrequently used code, DOS.H uses disk-resident overlays for the disk file opening, closing, and allocation routines. Most of the system error messages also reside in an overlay, allowing fully descriptive messages without using a prohibitive amount of memory. A set of short utility routines (DOS Functions) uses a separate overlay area.

The operating system uses one or two diskette controllers with at least one physical diskette drive attached, and may have up to four controllers with 9320 cartridge disks added. Each "on-line" drive -- a drive containing a disk ready to read -- is assumed to contain a valid DOS.H disk, which will have all necessary system tables and files present and in correct format. This assumption on the part of the system requires caution on the part of the operator if a disk not fitting this description is mounted. If, for instance, a disk has been mounted to be DOSGENed, the operator must not run any programs that will attempt to use the disk before it has been initialized, or an abort will occur indicating SYSTEM DATA FAILURE.

DOS.H is designed to be run interactively by an operator at the processor console. The operator generally enters commands from the keyboard, which the operating system interprets and executes. During execution, status information needed by the executing program is requested from the operator via CRT messages expecting a keyed response.

A DOS utility program (CHAIN) allows execution of predefined processes automatically in a non-interactive fashion, so no operator attention is required. Other utility programs extend this automatic capability such that the system can be made almost completely operator-independent if desired.

CHAPTER 42. SYSTEM STRUCTURE

42.1 Disk Structure

42.1.1 Introduction

DOS.H 2.5.1 supports the standard IBM format diskettes, and with the 64k system, the 9320 cartridge disk. This support is transparent; the user usually need not be concerned with the type of disk being accessed. Of course there are certain times when the disk type should be considered, for example when using REMAP, DOSGEN, BACKUP, and at system loading time. Reference should be made to the appendix on Disk Comparison for details on the difference in structure between diskettes and 9320 disks.

A 9320 disk pack is divided into four logical drives,while a diskette is by itself a single logical drive. Each logical drive is a self contained data structure containing up to 256 files and the necessary system information to access the files and manage file space on the disk. No system information on a drive references any other drive, with the possible exception of ISI (Index) files; these may be the index to a file on another drive.

The basic unit of disk storage is the sector, a group of 256 characters. All disk input/output is done in sectors. The smallest allocatable unit of disk space, and the unit the system deals with, is the cluster. A cluster is a group of sectors, the number of which depend on the disk type. The basic unit of user storage is the file, which is allocated in clusters.

Two system tables resident on each logical drive manage cluster allocation. The Cluster Allocation Table (CAT) maps those clusters which are allocated to a file. When space is needed for a file, the CAT is searched and free contiguous clusters are given to the files. The Lockout CAT maps those cylinders that were determined to be physically bad by DOSGEN, and those cylinders which the user requested DOSGEN to lock-out.

File space is mapped in segments, a segment being a group of contiguous clusters. A segment may not be larger than 32 clusters, but may be less than that as cluster availability and file space requirements dictate.

The first sector of every file is the file's Retrieval Information Block (RIB), which maps the segments of the file. File space is allocated in segments, and the CAT is searched for up to 32 contiguous clusters to make up the segment. When file space is de-allocated, unused clusters of the segments are returned to the CAT.

The Directory contains the names of files and pointers to their RIBs, thus completely defining a file. Since directory searches are done frequently, there is another system table associated with the directory to speed searches. On diskette, the CAT contains Directory Mapping Bytes (DMB), which are a count of the number of files in each of the 16 directory sectors. On 9320 disks, the Hashed Directory Index (HDI) is used to locate a file in the directory.

All of the system tables, the CAT, Lockout CAT, RIB, Directory, and HDI are kept in duplicate for backup purposes.

42.1.2 Disk Space Management: CAT and Lockout CAT

The Lockout CAT indicates locked out cylinders--cylinders which will not be used by the DOS. Cylinders are automatically locked out at DOS generation if they are found bad by the surface verification. Cylinders may be manually specified for lockout during system generation or by using DSKCHK15. Cylinder 0 is always locked out for system use. Each byte of the Lockout CAT represents a cylinder: byte 0=cylinder 0, byte 1=cylinder 1, byte 2=cylinder 2, etc. The byte value is 0377 (017 on diskettes) if the cylinder is locked out, and is 000, otherwise. Thus, on diskettes, each of the four low-order bits of the byte maps a cluster on the cylinder.

The CAT indicates available space for the DOS; CAT updates are performed automatically as space allocation or deallocation is performed. As in the Lockout CAT, each byte of the CAT represents a cylinder. Each bit of a byte represents a cluster of the cylinder: bit 7=cluster 0, bit 6=cluster 1, etc. (For diskettes, bits 7-4 are zero, bit 3=cluster 0, bit 2=cluster 1, bit 1=cluster 2, and bit 0=cluster 3). If a bit is set (1), the cluster it represents is either in use by a file or locked out; if a bit is clear (0), the cluster is free.

The CAT and Lockout CAT observe some fixed format rules:

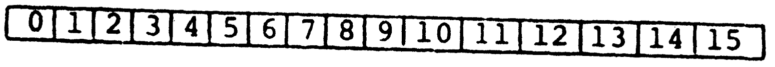
Byte 0 is always 0377
Bytes 1 through n (where n is the number of
    cylinders on the disk - 1) may be any values
    as described above
Bytes n+1 through 0356 are 377.
Bytes 0357-0376 are directory mapping bytes, which
    indicate the number of files allocated in the
    corresponding directory track. Values range from
    0-020. These are only present on diskettes.
    On disks, these bytes are 0377s.
Byte 0377 is any value. This is the auto-execute PFN and
    is normally zero.

42.1.3 Files: HDI, Directory, Mapping Bytes, RIB

The Hashed Directory Index (HDI) provides access to, and controls allocation of, the Directory on 9320 disks. Each byte of the HDI represents a directory entry, offset from the beginning of the index by PFN. Thus, byte 0=PFN 0, byte 1=PFN 1, and so on. If the value of the byte is 0377 the directory entry it represents is not in use. When a PFN is in use, a hash code (value 0-376) generated from the file name is placed in the byte. This value indicates the PFN is in use, and is used to speed directory searching when a file is being located by name.

Directory mapping bytes are used to speed directory accesses on diskettes. The mapping bytes are bytes 0357-0376 of the CAT. Each byte represents a directory sector (0-15) and the value in the byte represents the number of entries (0-16) in use in that sector.

The Directory is 16 sectors (logically referenced as 0-15) containing 256 directory entries, 16 entries per sector. A directory entry contains the name, protection, and subdirectory of a file; it also points to the file's RIB. Directory entry format:



Bytes 0-1 are the RIB address/protection. (See

"Addressing Byte Structures".)

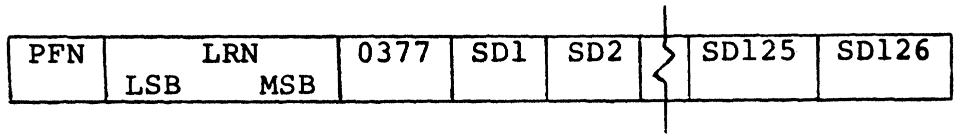
Bytes 2-3 are unused (normally zero)

Bytes 4-11 are the file name. A file name is usually ASCII characters as described in the DISK FILES chapter under File Names, padded with blanks to be eight characters long, but may be any values.

Bytes 12-14 are the file extension. Same format rules as file name.

Byte 15 is the subdirectory number, usually 0377, indicating subdirectory SYSTEM.

A Retrieval Information Block (RIB) maps a file's domain on disk. A file is composed of segments, each segment being composed of contiguous clusters. The RIB contains up to 126 segment descriptors which completely describe the clusters allocated to a file.



Each segment descriptor (SD) is two bytes long (see "Addressing Byte Structures"). A segment descriptor of 0377,0377 indicates the end of the RIB. The fourth byte of a RIB is always 0377. The RIB is always the first sector of the file; the RIB copy is the second sector and is identical to the RIB except that its LRN is 1.

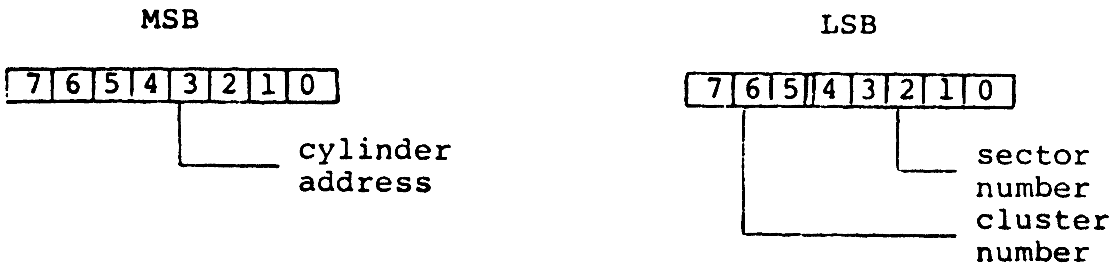
42.1.4 Sector Identification

Every sector of a file contains in its first byte the PFN of the file. The next two bytes are the Logical Record Number (LRN), stored least significant byte first. The PFN and LRN are primarily intended as validation fields when a file record is read. When a file record is written, the PFN and LRN are set correctly; reading a record with a PFN that does not match or an out-of-sequence LRN constitutes a Record Format Error.

Not every sector in the space allocated to a file has this PFN and LRN data. Only sectors that have been used for the file have this information set. Unused sectors may have anything in the first three bytes.

42.1.5 Addressing Byte Structures

42.1.5.1 PDA - Physical Disk Address

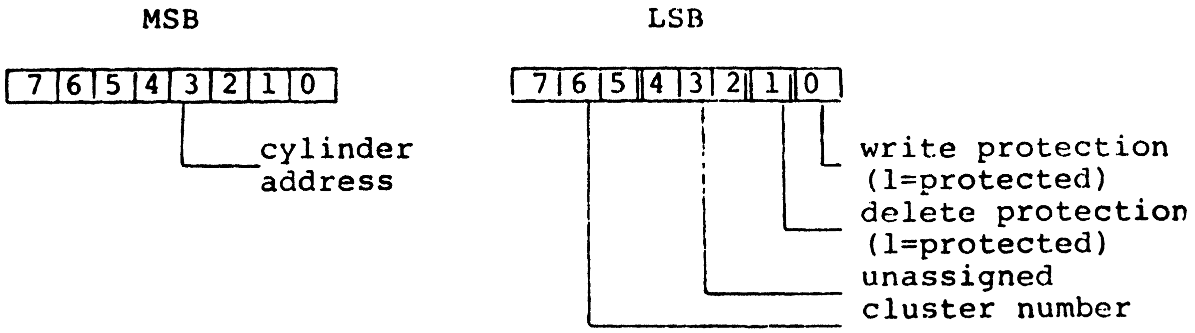


The cluster number references a cluster within a cylinder; values are 0-7 except for diskettes which use values 0,2,4,6 for sectors 0,1,2,3 respectively. The sector number references a sector within a cluster.

Note: This is the DOS.H "PDA" and must not be confused with the hardware disk addressing of the controller.

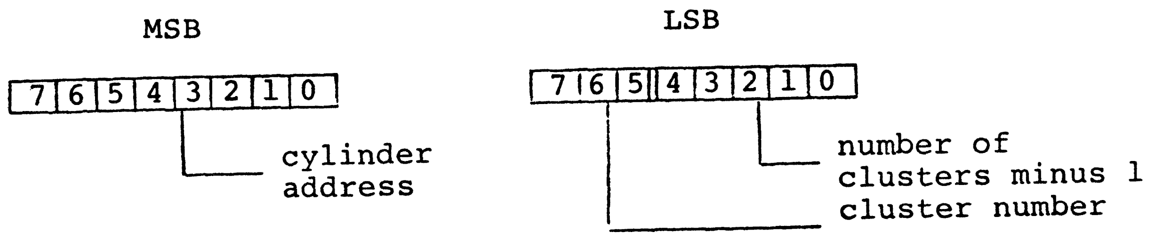
42.1.5.2 RIB Address/Protection

This is used in a directory entry to point to the beginning of a file.



The cylinder address and cluster number, with an assumed sector number of zero, is the PDA of the first sector of the file.

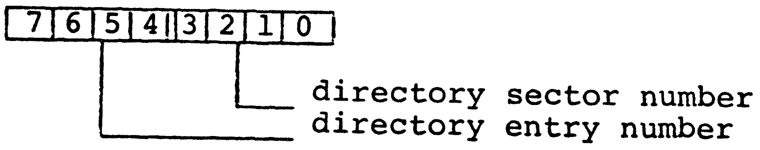
42.1.5.3 Segment Descriptor - used in RIB to define a segment.



The cylinder address and cluster number, with an assumed sector number of zero, is the PDA of the first sector of the segment. The length of the segment in clusters is given by the low-order five bits of the lsb; length can be 1-32 clusters.

42.1.5.4 Physical File Number

The physical file number is used to access the directory, the directory mapping bytes and the HDI.



The directory sector number specifies a sector of the directory (0-15). The directory entry number (0-15) specifies an entry within a sector.

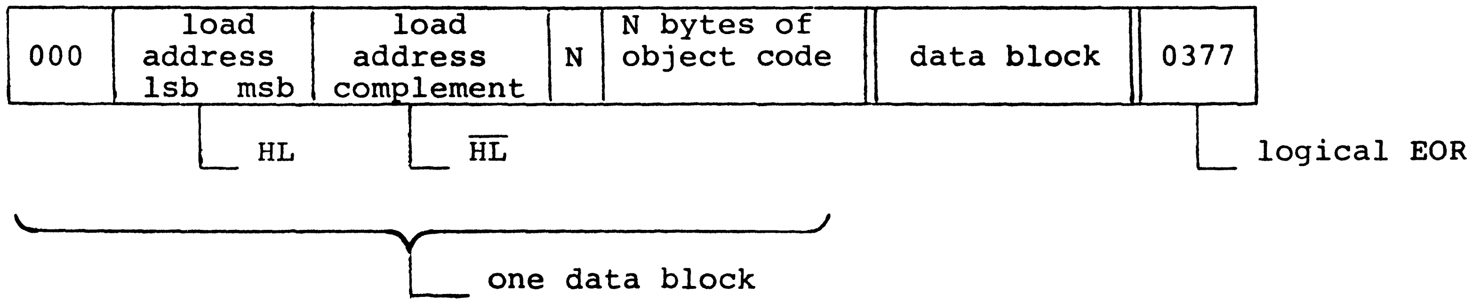
Note: Since directory entries are 020 bytes long, if the low-order four bits of the PFN are set to 0, the resulting value is the byte location of the beginning of the specified directory entry. For example, PFN 0304 references the directory entry beginning at byte 0300 (entry number 014) of sector 4 of the directory.

42.2 Disk Data Formats

The DOS itself does not deal with the user's data below the record level. It only keeps track of where the records are, allowing the user to format the data in any manner he pleases. The user is presented with records that are 253 bytes long, the first three bytes of each sector being reserved for system sector identification as described above. The DOS and its utility programs do make a number of assumptions concerning file structure however, and system operation is much simpler if all files are structured to match these assumptions.

DOS makes assumptions about the structure of text files and about absolute object code files. The structure expected for text files under DOS is described in the chapter on REFORMAT. Any file to be processed by the standard text-handling facilities of DOS must have the standard text format described.

If a file is to be loaded by the system loader, it must be an object code file in the following format:

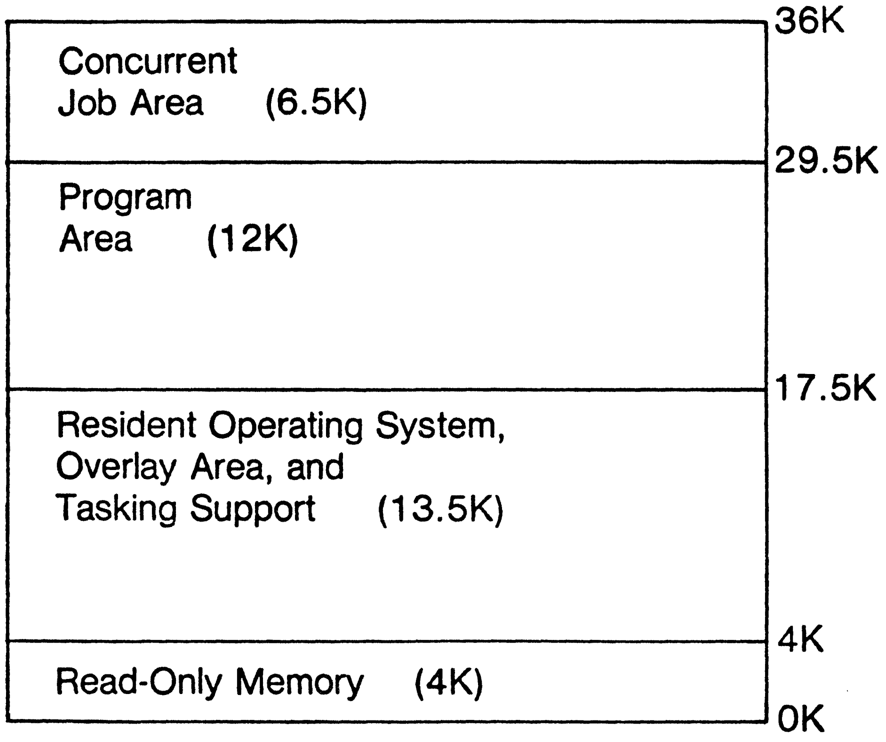


Note that this is the format of output files from Datapoint assemblers. Any number of data blocks may appear in a record. The leading byte of a data block will always be either 0, indicating a block follows, or 0377, indicating end of record. The special case of N being zero is used to indicate end of file, in which case the HL given is taken to be the starting address of the program loaded.

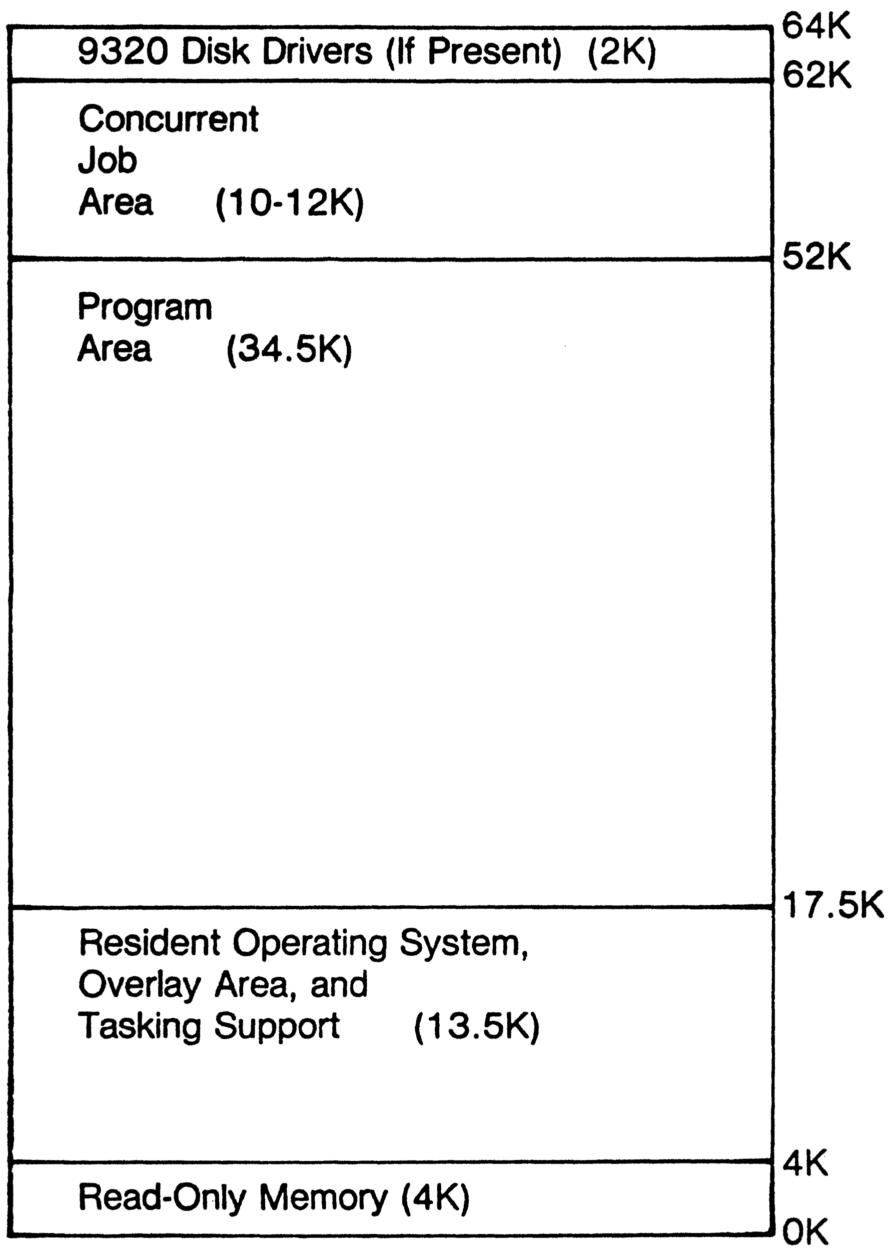
42.3 Memory Mapping

The DOS occupies memory as shown by the following map:

42.3.1 32K System


36K
Concurrent Job Area (6.5K)
29.5K
Program Area (12K)
17.5K
Resident Operating System, Overlay Area, and Tasking Support (13.5K)
4K
Read-Only Memory (4K)
0K

42.3.2 64K System


64K
9320 Disk Drivers (If Present) (2K)
62K
Concurrent Job Area (10-12K)
52K
Program Area (34.5K)
17.5K
Resident Operating System, Overlay Area, and Tasking Support (13.5K)
4K
Read-Only Memory (4K)
0K

42.4 Disk Overlays

DOS uses disk overlays to reduce its main memory requirements. The overlays are in disk files SYSTEM1/SYS through SYSTEM7/SYS. The memory-resident DOS is stored in the disk file SYSTEM0/SYS. These eight files must reside in PFN's 0 through 7, the PFN corresponding to the number in the file name.

The system overlay files load into memory and are loaded by the system as needed. The exception to this is SYSTEM4/SYS, which is the communications/tasking package. It is loaded into memory when the system is first booted, and is not reloaded again until the next manual boot. The functions of the overlays are:

SYSTEM1/SYS	- PREP/ALLOC	- create a new file or allocate more space
SYSTEM2/SYS	- CLOSE	- close a file
SYSTEM3/SYS	- OPEN	- open an existing file
SYSTEM4/SYS	- TASK	- system tasking handler
SYSTEM5/SYS	- ABORT	- display an error message
SYSTEM6/SYS	- SCREEN	- initialize a RAM display screen

SYSTEM7/SYS is the DOS Function overlay. The DOS Functions are short overlay routines and load into a separate area of memory. Also, the first sector of SYSTEM7/SYS is used to store subdirectory names (see the SUR command). When DOS needs an overlay file, it searches for the file on the booted drive only.

42.5 The Command Interpreter

The command interpreter receives command lines from the keyboard, as described in the chapter on Operator Commands, storing the command line in memory in the Monitor Communication Region (MCR$). When the line is terminated (ENTER key, 015), the stored command line is scanned and the indicated command program is loaded and executed.

While the command interpreter is waiting for character entry from the keyboard, it runs a test on the disk buffer memory. As soon as a character is ready from the keyboard, the disk buffer memory test is terminated and the normal keyin routine is entered. Even just striking the CANCEL key will terminate the disk buffer memory test. If an error is detected by the disk buffer memory test, the message "DISK BUFFER FAULT" is displayed and the screen is rolled up one line.

Whenever the command interpreter is entered via its main entry point, which most programs use, it will execute the program set for auto-execution if there is one. This happens whenever the operating system is reloaded. If the keyboard (KBD) key is depressed, auto-execution is not performed.

When a command line has been entered, the command interpreter must attempt to locate and load the specified command program. If the command is obviously bad (a null entry line) the interpreter immediately displays "WHAT?" and waits for a new line. Normally the first field on the command line will be normalized to the form shown below and the file thus specified will be searched for. The sequence of searching for a requested program depends on the format of the command line.

If the operator entered a leading "*" or ":" as part of the command name, a flag called UTILSW (UTILity SWitch) is set, incicating that the specified command is to be located as a member of UTILITY/SYS. If a drive specification was entered as part of the command name, the search goes only to the specified drive, as indicated in the sequence shown below.

The first test the interpreter performs is to check the drive specification entered, if any. If the drive specification is invalid, an error message is displayed and a new command line requested. If the drive specified is valid, or if no drive was specified, the interpreter searches for the command as outlined below.

1. If a drive was specified:
   a. If UTILSW is set:
      (1) Open UTILITY/SYS on the specified drive. If the file is missing or if the specified command is not a member, say "WHAT?", else run the program.
   b. If UTILSW is not set:
      (1) Attempt to open the command file on the specified drive. If successful, run the program. Else:
      (2) If no extension was specified in the command name, open UTILITY/SYS on the specified drive and search for the command as a member of the library. Else:
      (3) "WHAT?" and get another command.
2. If no drive was specified:
   a. If UTILSW is set:
      (1) Open UTILITY/SYS on the booted drive and search for the command as a member of the library. Else:
      (2) Try to open command as a file on booted drive.

Else:
(3) Check for command in UTILITY/SYS on any drive.
Else:
(4) Try to open comand as a file on any drive. Else:
(5) "WHAT?" and get another command.

b. If UTILSW is not set:
(1) Try to open command as a file on booted drive.
Else:
(2) If no extension was specified in the command name, open UTILITY/SYS on the booted drive and check for the command as a member of the library.
Else:
(3) Try to open command as a file on any drive. Else:
(4) If no extension was specified in the command name, open UTILITY/SYS on any drive and check for the command as a member of the library. Else:
(5) "WHAT?" and get another command.

The command interpreter uses lexical scanning routines to interpret the entered command line. The command interpreter scans up to four file specifications from the command line. The file name scan is terminated by a semicolon (;) or end-of-string (015). The file specifications are entered in a normalized symbolic form into the corresponding logical file table entries (0 through 3). The normalized form is not the same as normal LFT information, the LFT area simply provides convenient storage for the file specifications. The format of the normalized form is shown here:

DRCODE	(1)	- Drive select code: logical drive number in binary, no drive spec (0377), invalid drive spec (0376)
0377	(1)	- PDN location of normal LFT, set to 0377 to indicate the LFT is closed.
FILENAME	(8)	- File name specified, padded with trailing spaces to 8 characters. Eight spaces if no name given.
FILEEXT	(3)	- File extension specified. Three spaces if no extension entered.
DRSPEC	(3)	- Logical drive specification (spaces if no spec).

When a program receives control from the command interpreter, LFT's one through three (zero was used to load the program itself) contain normalized entries as indicated above, and MCR$ still contains the command as entered, so the program can retrieve information from its command line. If a program is auto-executed, none of this command line information is available, so any program which tests for information as provided above cannot be auto-executed. Conversely, any program intended for auto-execution must not look for command information. The command AUTOKEY is provided to allow automatic execution of programs requiring command line information.

CHAPTER 43. DOS ERROR MESSAGES

When a fatal error occurs during program execution and the error is not handled by the program, DOS aborts the program and displays one of the following error messages:

PARITY FAILURE DURING READ
: A parity fault occurred while a disk data record was being read.

PARITY FAILURE DURING WRITE
: A parity fault occurred while a disk data record was being written.

RECORD FORMAT ERROR
: The physical file number or logical record number in the record read did not match the values contained in the logical file table.

RECORD NUMBER OUT OF RANGE
: The record accessed had a logical record number less than zero or, during reads, was outside the physical space allocated to the file.

WRITE PROTECT VIOLATION
: An attempt was made to write to a file that had its write protection bit set.

DELETE PROTECT VIOLATION
: An attempt was made to delete a file that had either its write or delete protection bit set.

FILE SPACE FULL
: An attempt was made to allocate space when either the disk was physically full or no more segment descriptor slots were available in the RIB for the given file.

DRIVE OFF LINE
: The drive went off line after the file was opened.

FAILURE IN SYSTEM DATA
: An unrecoverable parity error occurred while the system was dealing with one of the disk tables or a Retrieval Information Block (RIB), or a RIB with incorrect format was accessed.

INTERNAL SYSTEM ERROR
The error message routine was parameterized with an invalid error message number!

BAD DRIVE
The drive specification given on the DOS command line is invalid.

SYS. DRIVE O/L
The currently "booted" drive has gone off-line.

CHAPTER 44. 1500 ROMGUIDE

44.1 System ROM Functions

The DATAPOINT 1500 diagnostic tool is a ROM resident program whose immediate accessibility creates a flexible interface between the user and machine. This guide is intended to provide the 1500 User with that information essential to the use of the resident diagnostic system.

44.1.1 Startup Procedure

The following conditions may cause entry to the diagnostic system:

* Entry through a BREAKPOINT set by the diagnostic system.
* Various kinds of hardware errors will also cause the diagnostic system to be entered.

To force entry into the diagnostic system, depress in sequence: DISPLAY, INTERRUPT, RESTART, keeping each key depressed until all three are down. Then release INTERRUPT or RESTART. This will bring up the diagnostic system display, and commands may be entered. It is important to understand that the diagnostic system operates as a task, the same as all the other tasks in the system, although of higher priority. Thus while the diagnostic tool is in the wait state, other tasks may be running. This may be a problem when trying to diagnose programs with multiple tasks, especially those which do unorthodox things with the display.

44.1.2 Display Format

The diagnostic display consists of four lines and occupies the bottom-right corner of the screen.

```
          AAAAAA   : CURADR
          *  NNN   : ASCII,8 BIT OCTAL C[CURADR]
          MMMMMM   : LSB,MSB ADDRESS FORMED AT CURADR.
      nnnnnnn*     : COMMAND INTERPRETER
```

The first (top) line shows the current sixteen bit address.

The second line contains both an ASCII (One character shown as *) and an 8-bit octal (Three characters shown as NNN) representation of the contents of the current address byte.

The third line shows the LSB,MSB of an address when the LSB is pointed to by the CURADR. In other words, CURADR points to a memory location which itself is an address. If it's the LSB, it is combined with the next memory location (MSB), turned around for greater readability and displayed.

44.2 The Command Interpreter

The bottom line of the display is an interpreter used to input commands to the diagnostic tool. The blinking cursor signifies that the Command Interpreter is awaiting user input.

Data is entered serially into the input display buffer. The cursor is displaced to the right successively as this occurs. The Backspace key erases the character most recently entered, shifting the entry cursor to the left one space. The cancel key deletes the entry. All commands are terminated by the ENTER key. In other words, the command keyin works exactly like the standard DOS keyin.

All commands are single characters. Commands which accept input arguments are preceded by the argument, which is entered in octal. Not all commands require an input argument. Illegal input is ignored, evoking a beep from the 1500. Commands are executed when the ENTER key is depressed, after the entire command has been entered. If the command is one which returns to debug, the cursor reappears, and the routine waits for further input.

44.2.1 Command Syntax

This explanation of the command syntax uses the following notation:

nnnn... Indicates an optional sequence of octal digits not to exceed the number of n's given.

If input argument contains more than eight bits of significance special results will occur. In general what will happen is that two bytes of memory will be affected by the command, either a register pair or a memory address in LSB,MSB format.

12345 There are several special commands whose accidental execution is inhibited by the requirement that they contain this unique argument.

44.2.2 Input Command List

nA This is a diskette test command. It will cause the drive (bits 0-1) and controller (bits 2-3) indicated by "n" to seek to track 38 and read sector zero continuously, ignoring any read errors. It is used for aligning the head in the diskette drive. See the notes on error reporting under the "Z" command.

12345R This is a system reset command. It resets the firmware, clears the display, loads an abbreviated character set, and reenters the diagnostic system. The interrupted program will not be able to continue execution after this command is executed.

12345T Start rotating bit memory test. Displays Memory Size and Pass Counter in right-bottom corner of screen. Maintains running display of Test Failures. The memory test may be interrupted by forcing entry into the diagnostic system again. However, it stops only at the end of a complete pass.

nV This is another diskette test. It causes the drive (bits 0-1) and controller (bits 2-3) indicated by "n" to seek to track 76 and then begin reading every sector on the diskette, starting with track zero, sector zero, and stopping if errors are detected. See the notes on error reporting under the "Z" command. The notation 'n' should be 0 for Drive 0, 1 for Drive 1, 4 for Drive 2, and 5 for Drive 3.

nX Start a continuous diskette controller buffer test on conroller indicated by n (n=0 for controller 0; n=4 for controller 1). If an error occurs, the debugger is reentered with the current address set to that of a three-byte status save area. The first byte contains the value read from the buffer, the next byte contains the expected value. The third byte contains the buffer position. To stop the test, push the RESTART and INTERRUPT keys with the DISPLAY key held down.

nZ This is another diskette test. It causes the selected drive (bits 0-1) and controller (bits 2-3) indicated by n to continuously seek between tracks 1 and 2, reading sector zero on each track and ignoring any read errors. This operation may also be stopped by reentering the diagnostic system. In that case, it will stop at the end of the next diskette I/O operation. Its primary use is aligning the Track Zero sensor in the diskette drive.
The A, V, and Z commands may all stop for certain kinds of errors. When and if they do, the current address (in the standard diagnostic display) will point to a three-byte save area. The first byte of this area contains the diskette status. The second byte contains the current track number, and the third byte contains the current sector number.

<Cancel> Cancel entry line.

<BSP> Backspace on entry line.

? Display the current version of the firmware. The current address is set to point to a two-byte area which contains the version number.

44.3 Hardware Error Messages

There are certain error conditions which are detected in the hardware which may cause the diagnostic display to come up. If these occur, the top line of the diagnostic display, in the lower right-hand corner of the screen, has a message which indicates the error. Some of these are caused by software conditions but are indicated as hardware errors. These are as follows:

LRA ERR -- This is caused by an internal error in a system program.

MEM ERR -- This indicates that the system detected a parity error in the memory. This is a true hardware error. To attempt to clear it, the machine should be powered off and back on. If this error persists, run the memory test (12345T), and place a service call.

KBD ERR -- This is another error which is generally caused by an error in a system program.

NMI ERR -- This error is only caused by errors in the system programs.

APPENDIX A. DISK COMPARISON CHARTS

This appendix lists the attributes of the disk types supported by DOS.H on the 1500 processor.

ATTRIBUTE	DISKETTE	9320 DISK
Min. processor required	32K	64K
Disk controller/drive type	1542,1543	9320
Physical drives on system	1-4	0-4
Logical drives per physical drive	1	4
Disk type	IBM diskette 128-byte soft sectored	9320 cartridge
Cylinders used/logical drive	77	49
Tracks used per logical drive	77	196
Sectors/track	13	48
Sectors/logical drive	1001	9408
Bytes/logical drive	256,256	2,408,448
User sectors/logical drive (with system files)	696	9000
User bytes/logical drive	118,176	2,304,000
Sectors/cluster	3	24
Clusters/track	4	2
Clusters/cylinder	4	8
Max. clusters per segment	32	32
Max. sectors per segment	96	768
Max. sectors per file	696	9000
Directory search	DMB	HDI

This table lists the PDAs of sytem tables on the supported disk types. Here, "cylinder" refers to the PDA MSB, and "sector" refers to the PDA LSB. All PDAs are in MSB,LSB octal format.

TABLE PDA	DISKETTE	9320 DISK
CAT	041,014	0,0
Backup CAT	042,014	0,040
Lockout CAT	043,014	0,01
Backup Lockout CAT	044,014	0,041
HDI	--	0,02
Backup HDI	--	0,042
Directory Location	sector 014 cylinders 021 to 040	cylinder 0 sectors 07 to 026
Directory backup location	sector 014 cylinders 01 to 020	cylinder 0 sectors 047 to 066

CUT ALONG LINE FOR MAILING

Manual Name______________________________

Manual Number____________________________

READER'S COMMENTS

Did you find errors in this manual? If so, specify by page.

Did you find this manual understandable, usable, and well-organized? Please make suggestions for improvement.

Name____________________ Date________________

Organization______________________________

Street__________________________________

City_______________ State________ Zip Code____________

All comments and suggestions become the property of Datapoint.

Fold Here

Fold Here and Staple

FIRST CLASS
Permit
5774
San Antonio
Texas

BUSINESS REPLY MAIL
No Postage Necessary if mailed in the United States

Postage will be paid by:

DATAPOINT CORPORATION
DIRECTOR OF SOFTWARE SUPPORT
8550 Datapoint Drive, Mail Station# N60
San Antonio, Texas 78284