JIMMY

Prime Computer, Inc.

DOC5027-183P Magnetic Tape User's Guide

Revision 18.3

Magnetic Tape User's Guide

DOC 5027-183

First Edition

by

Susan F. Adley

This guide documents the software operation of the Prime Computer and its supporting systems and utilities as implemented at Master Disk Revision Level 18.3 (Rev. 18.3).

Prime Computer, Inc.
500 Old Connecticut Path
Framingham, Massachusetts 01701

HOW TO ORDER TECHNICAL DOCUMENTS

U.S. Customers

Software Distribution
Prime Computer, Inc.
1 New York Ave.
Framingham, MA 01701
(617) 879-2960 X2053, 2054

Prime Employees

Communications Services
MS 15-13, Prime Park
Natick, MA 01760
(617) 655-8000, X4837

Customers Outside U.S.

Contact your local Prime
subsidiary or distributor.

INFORMATION Systems

Contact your Prime
INFORMATION system dealer.

PRINTING HISTORY — Magnetic Tape User's Guide

Edition	Date	Number	Supports Rev
First Edition	December, 1981	DOC5027-183	18.3

SUGGESTION BOX

All correspondence on suggested changes to this document should be directed to:

Susan F. Adley
Technical Publications Department
Prime Computer, Inc.
500 Old Connecticut Path
Framingham, Massachusetts 01701

Contents

About This Book

The Magnetic Tape User's Guide is for both new and experienced users of magnetic tapes. It contains information on magnetic tape and tape device characteristics, and PRIMOS magnetic tape commands and subsystems. For more detailed magnetic tape information for the System Administrator and operator, refer to the System Administrator's Guide.

HOW TO USE THIS BOOK

Part I (Chapters 1 to 3) introduce magnetic tape characteristics and related concepts. New magnetic tape users should read these chapters first as introductory material before moving on to later chapters. Those who have used tapes before might use Part I for reference purposes.

Part II (Chapters 4 to 6) covers PRIMOS magnetic tape commands. Chapter 4 describes commands for user control of tape drives. Chapter 5 describes commands for operator control of tape drives. Chapter 6 covers commands that initialize magnetic tapes.

Part III (Chapters 7 to 9) covers PRIMOS magnetic tape subsystems: MAGSAV, MAGRST, PHYSAV, and PHYRST. In addition, Chapter 7 is entirely devoted to the new MAGNET subsystem.

The Appendixes contain additional information on MAGNET edit tokens, translation tables, command line options, and batch jobs. Also included are ASCII, EBCDIC, and BCD character set tables, and notes on pre-Rev 18.3 MAGNET.

ACKNOWLEDGEMENTS

The author wishes to thank members of Prime's Technical Publications Department, Prime Engineering, and all non-team members who reviewed, contributed to, and assisted with this book.

PRIME DOCUMENTATION CONVENTIONS

The following conventions are used in command formats, statement formats, and in examples throughout this document. Command and statement formats show the syntax of commands, program language statements, and callable routines. Examples illustrate the uses of these commands, statements, and routines in typical applications. Terminal input may be entered in either uppercase or lowercase.

Convention	Explanation	Example
UPPERCASE	In command formats, words in uppercase indicate the actual names of commands, statements, and keywords. They can be entered in either uppercase or lowercase.	SLIST
abbreviations	If a command or statement has an abbreviation, it is indicated by underlining.	LOGOUT
lowercase	In command formats, words in lowercase indicate items for which the user must substitute a suitable value.	LOGIN user-id
underlining in examples	In examples, user input is underlined but system prompts and output are not.	OK, stat units USR=SUSANA ENA NO FILE UNITS OPEN OK,
Brackets [ ]	Brackets enclose a list of one or more optional items. Choose none, one, or more of these items (0 - n).	SPOOL [-LIST -CANCEL]

Braces { }	Braces enclose a vertical list of items. Choose one and only one of these items.	CLOSE { filename / ALL }
Ellipsis ...	An ellipsis indicates that the preceding item may be repeated.	item-x[,item-y]...
Parentheses ( )	In command or statement formats, parentheses must be entered exactly as shown.	DIM array (row,col)
Hyphen -	Wherever a hyphen appears in a command line option, it is a required part of that option.	SPOOL -LIST
(CR)	The (CR) symbol indicates a single carriage return which is generated by hitting the RETURN key on most terminals.	

PART I

Introduction to Magnetic Tapes

1 Magnetic Tapes

INTRODUCTION

Magnetic tapes used in computer applications are similar to those used in tape, cassette, and cartridge recorders. Instead of recording music, however, computer tapes record digital information. This information can be almost anything — social security records, test results, a data base, or even an encyclopedia.

Magnetic tapes are very inexpensive compared to other storage media such as magnetic disks and large dynamic RAM main memory. Of these three devices, data on tapes is the slowest to access, while main memory is the fastest. However, magnetic tapes can store more information per dollar than either memory or disk. For these reasons, computer storage is maintained as a hierarchy of some amount of main memory, an amount of disk storage, which is roughly one to three orders of magnitude larger than main memory and virtually unlimited tape storage.

PHYSICAL CHARACTERISTICS

Physically, a computer tape is a long, narrow piece of plastic film coated with iron oxide. To record information, certain areas of the coating are magnetized. The magnetization remains until the tape is remagnetized by recording new information over the old information or until the tape is passed through a bulk-erasing device. Because tapes are susceptible to magnetic fields they must be stored carefully to guard against either accidental or malicious destruction of information.

Just like motion-picture film, computer tape is wound on a reel. The reel usually contains a removable ring on one side called the write-enable ring. (See Figure 7-4.) If the ring is removed, the tape can be read by the computer, but no information can be written. If the ring is not removed, the tape can be read or written.

Dimensionally, a computer tape is 1/2 inch wide and from 600 to 2400 feet long. (600, 1200, or 2400 feet are the usual standard sizes available from suppliers.) The tape reel is ordinarily made of plastic as is the write-enable ring. The tape is not threaded into the reel; instead, it is wound onto the reel until there is sufficient friction to prevent the tape from slipping.

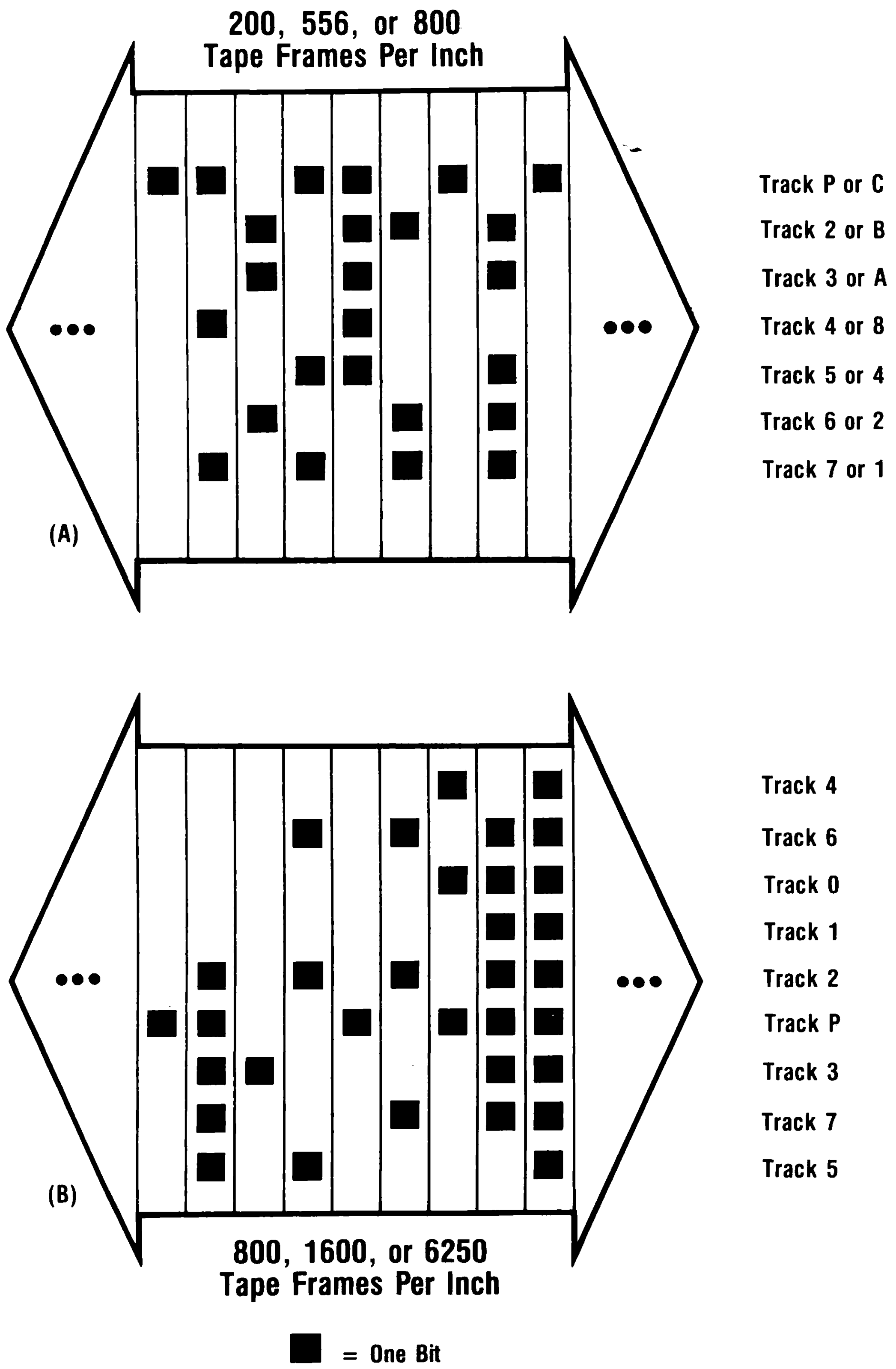
DENSITY, TRACKS, AND FRAMES

The information on computer tapes is recorded as a series of magnetized flux changes which can be placed as close together as several hundred to several thousand per inch. The most common amounts are 200, 556, 800, 1600, and 6250 per inch. This is called the tape density and is usually written as either bpi (bits per inch) or cpi (characters per inch).

To store more information, computer tapes contain several parallel tracks, running the length of the tape. This is similar to stereo or quadrophonic audio recordings where a tape has several channels. Computer tapes usually contain either seven or nine tracks; a tape with seven tracks is called a seven-track tape. Nine-track tapes contain nine rows of magnetic spots (called bits).

Tracks run the length of the tape. Characters are written across the tape. That is, each character is recorded as a pattern of bits, one bit per track. The area needed to hold one bit per track across the width of the tape is called a tape frame (or tape character). A seven-track tape frame has seven bits, one for each track, while a nine-track tape frame has nine bits, one for each of its nine tracks. Nine- and seven-track tapes are both the same width; nine-track tapes have narrower tracks.

The various tape concepts just discussed are summarized in Figure 1-1.



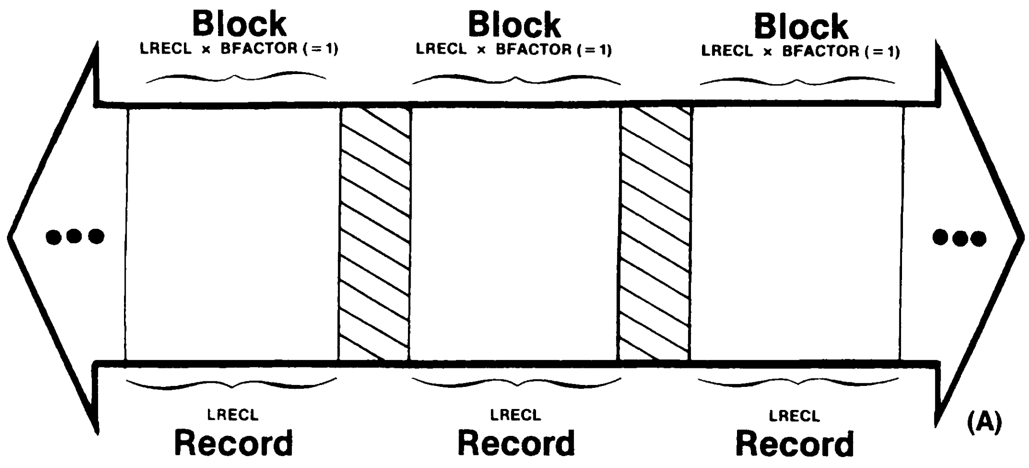
Seven-Track Tape (A), Nine-Track Tape (B) Formats
Figure 1-1

RECORDS, GAPS, AND BLOCKS

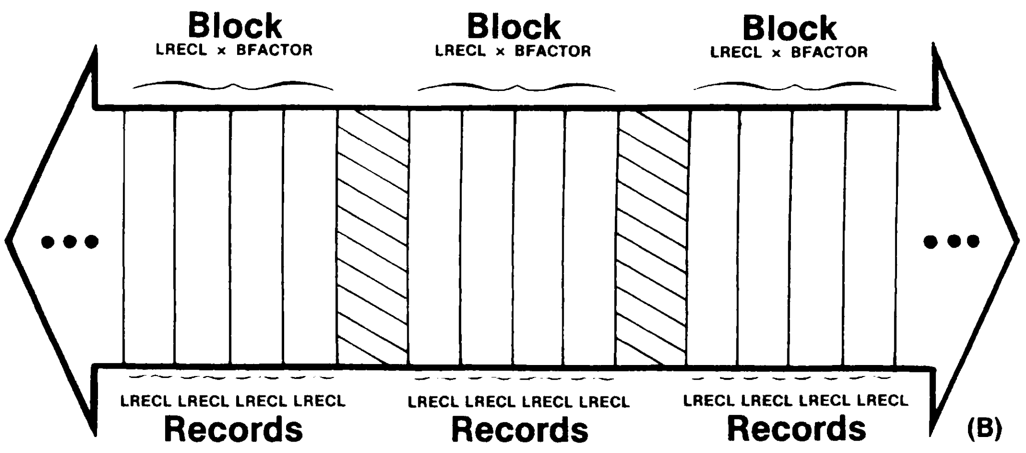
The computer can only read and write a certain number of frames during a single read or write operation. The minimum is usually about 10 frames; the maximum on Prime computers is 12,288 frames. A group of frames read or written at one time is called a physical tape record or physical block. Physical records are separated by inter-record gaps (IRGs) or "dead" space where no useful information is recorded. The inter-record gap contains no useful information and is at least 1/2 inch long, and may sometimes be as long as 3 inches. Records are preceded by a preamble and followed by a postamble, which are both written by the hardware and are not user-visible. The preamble alerts the tape drive that a record immediately follows, while the postamble signals that the end of a record has been reached. In addition, the postamble contains some "check" information used to validate whatever was just read or written.

If you wish to write many small records, there will be a high percentage of inter-record gaps on the tape. For example, if 80-character records are written at a density of 800 bpi, the records will each occupy 1/10 of an inch. The IRGs will be at least five times as large as each record. The solution to this problem is called blocking. The maximum size of Prime tape records is 12,288 characters (12K bytes or 6K words). The 80-character logical records mentioned above could be "blocked" 153 per physical record. The blocking factor in this case is 153. Forty-eight characters are unused, so the physical records written would be 12,240 characters long. When all logical records are the same size, they are called fixed-length records. Figure 1-2 illustrates both blocked and unblocked fixed-length records.

The tape drive "knows" nothing of blocking; the 12,240 characters appear to be one large physical record. User programs must perform the blocking and unblocking of physical-to-logical and logical-to-physical translations.



(A)



(B)

■ File Marker

□ Beginning and/or End of Tape Marker

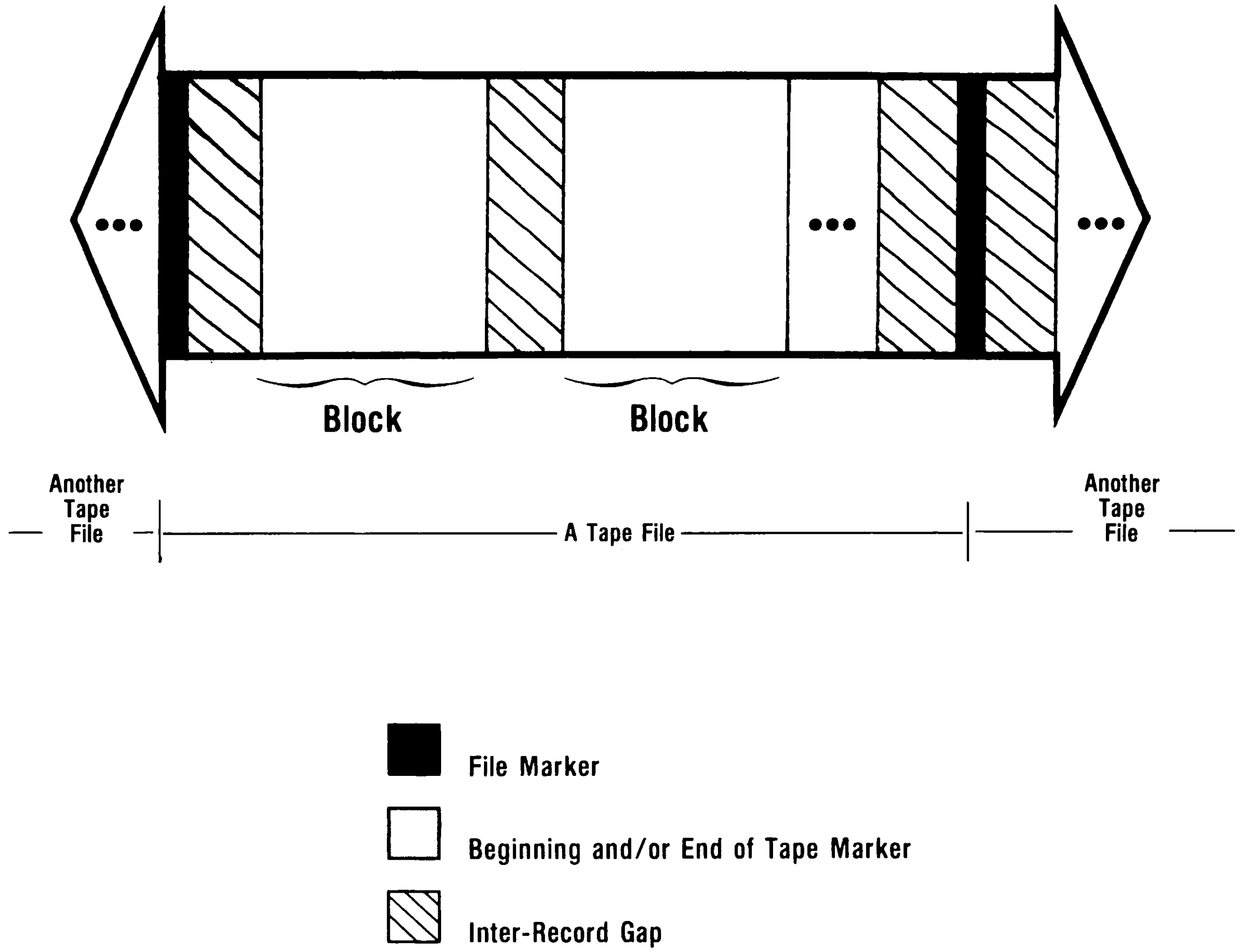
▨ Inter-Record Gap

Fixed Length Records (A) Unblocked (B) Blocked
Figure 1-2

TAPE FILES

Blocks, in turn, are grouped together to form a tape file. The file limits are marked or delimited by special control characters which can be placed on the tape. These characters are called file markers or tape markers. A file marker is actually composed of about 3 inches of "dead" space followed by a special file marker control character. The entire file marker area is at least 4 inches long (the "dead" space area may be longer). The term for the "dead" space area plus the file marker control character is file marker gap, but the term file marker (or tape marker) will be used in this manual.

When a tape contains blocks that are separated by file markers, the tape is said to have a logical tape format. The logical tape format is shown in Figure 1-3.



Logical Tape Format
Figure 1-3

TAPE DRIVES

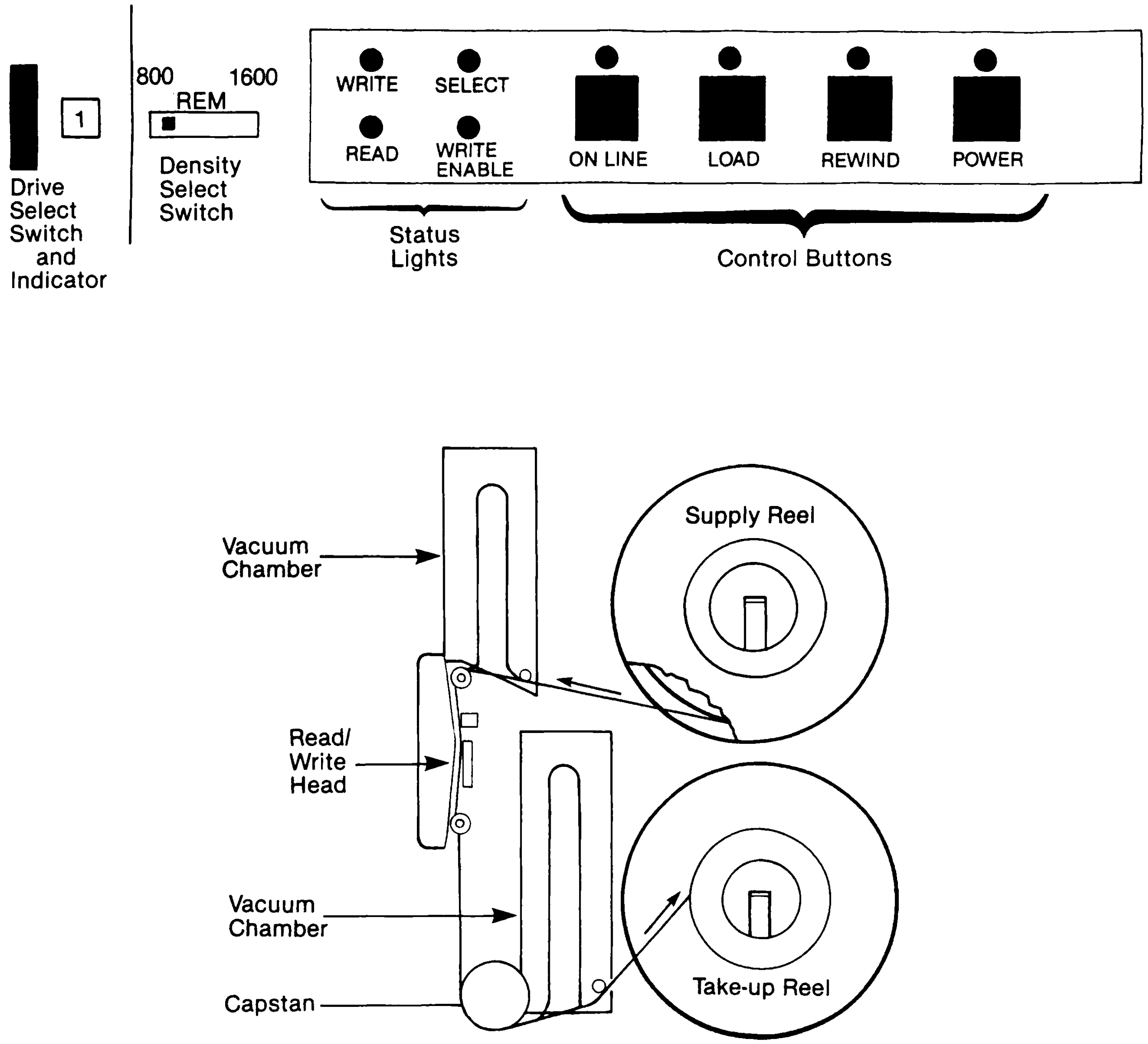
Tapes are read and written by a machine similar to a tape recorder. Like a tape recorder, a tape drive has two hubs on which the supply and take-up reels are placed. The supply reel contains the tape to be read or written. Just as on a motion-picture projector, the take-up reel receives the tape after it has been read or written.

As the tape is processed, it unwinds from the supply reel, passes through a vacuum chamber, moves under the read/write heads, passes through another vacuum chamber, around the capstan, and winds onto the take-up reel. The vacuum chambers ensure that there is never too much or too little tension placed on the tape. The capstan controls the tape speed. The read/write heads either read from or write information to the tape. In addition to these two operations, tape drives can rewind to the beginning of the tape, move forward or backward at several different speeds, provide status information about tape operations, and perform several other functions.

A few feet after the beginning and a few feet before the end of the tape are two silver metallic strips called the beginning-of-tape (BOT) marker and the end-of-tape (EOT) marker. These markers are automatically sensed by the tape drive to ensure that the tape is never pulled off either reel.

Tape density is set either by a switch on the tape drive itself, through a program on the computer, or both. These various tape drive features are shown in Figure 1-4.

There are several other tape drive features that should be noted. One is the speed, measured in inches per second. This is the rate at which the tape drive can move the tape and is typically 45, 75, or 125 inches per second (ips). The distance from the beginning-of-tape marker to the first inter-record gap is about 40 inches. The preamble and postamble have several different formats. These are called the phase-encoded (PE), non-return to zero inverted (NRZI) and group code recording (GCR) techniques. GCR is available only on 6250 drives on Prime systems; PE is available only at 1600 bpi; NRZI is available on 800 bpi drives only. For more information regarding tape recording techniques, refer to the appropriate American National Standards Institute (ANSI) manuals listed in the bibliography.



Tape Drive Features
Figure 1-4

PARITY

Most computers store information in either six- or eight-bit "packets" called bytes. The Digital Equipment Corporation PDP-8 and several Control Data Corporation machines are examples of six-bit byte machines. All Prime computers, IBM's 360/370 series, and Hewlett-Packard's HP-2100 are examples of eight-bit byte computers. Seven-track drives were originally developed for six-bit byte computers, while nine-track drives were developed for eight-bit byte computers. In each case, there is one more track on the tape than there is in the computer byte. The extra track is called the parity track. Each tape frame thus has a parity bit.

The parity bit is used as an additional "check" on the validity of the information stored on the tape. There are four possible ways a parity bit can be set:

- Always on; error if a parity bit off condition occurs
- Always off; error if a parity bit on condition occurs
- Even parity; the data bits for a frame are added. If the result is even, the parity bit is set to 0, otherwise it is set to 1.
- Odd parity; the data bits for a frame are added. If the result is even, the parity bit is set to 1, otherwise it is set to 0.

Seven-track tapes can be read or written with either the even or odd parity selections. Prime tape drives always write odd parity for nine-track tapes.

CHARACTER CODES

The information stored in a byte may be of two types: a number or a character. Since a byte is eight bits, 256 different numbers or characters may be represented in one byte. The interpretation of the contents of a byte depends on the application program. In the case of characters, the user could assign any number to mean any character desired; however, there are already certain standard combinations that are widely recognized throughout the computer industry. Among these character codes are the American National Standard Code for Information Interchange (ASCII), the Extended Binary Coded Decimal Interchange Code (EBCDIC) and the Binary Coded Decimal Interchange Code (BCDIC or BCD).

ASCII is the code used on Prime computers. EBCDIC is used on IBM systems and BCDIC is used primarily as a seven-track tape interchange code. To write information currently encoded in ASCII to a seven-track tape, it must first be translated to BCDIC, which is a six-bit code. Since ASCII is a seven-bit code (the extra bit is always a "1" on Prime systems), one half of all Prime ASCII characters cannot be translated to BCDIC characters. Among these are lowercase letters and some special characters. EBCDIC is an eight-bit code; one half of all EBCDIC characters, therefore, cannot be translated to ASCII. However, in practice, most of the "extra" EBCDIC characters are not used on IBM or Prime systems anyway.

Some characters cannot be readily translated to meaningful characters in another code. The solution is usually to translate to a character that will be recognized as an "untranslatable" character.

ASCII is a seven-bit code fully defining 128 characters. However, most computers that use ASCII have eight-bit bytes. Prime software sets the remaining (high-order) bit to a 1. This fact must be kept in mind whenever transferring information between Prime systems and other ASCII-based systems that set high-order bits to 0.

BINARY PACKING AND UNPACKING

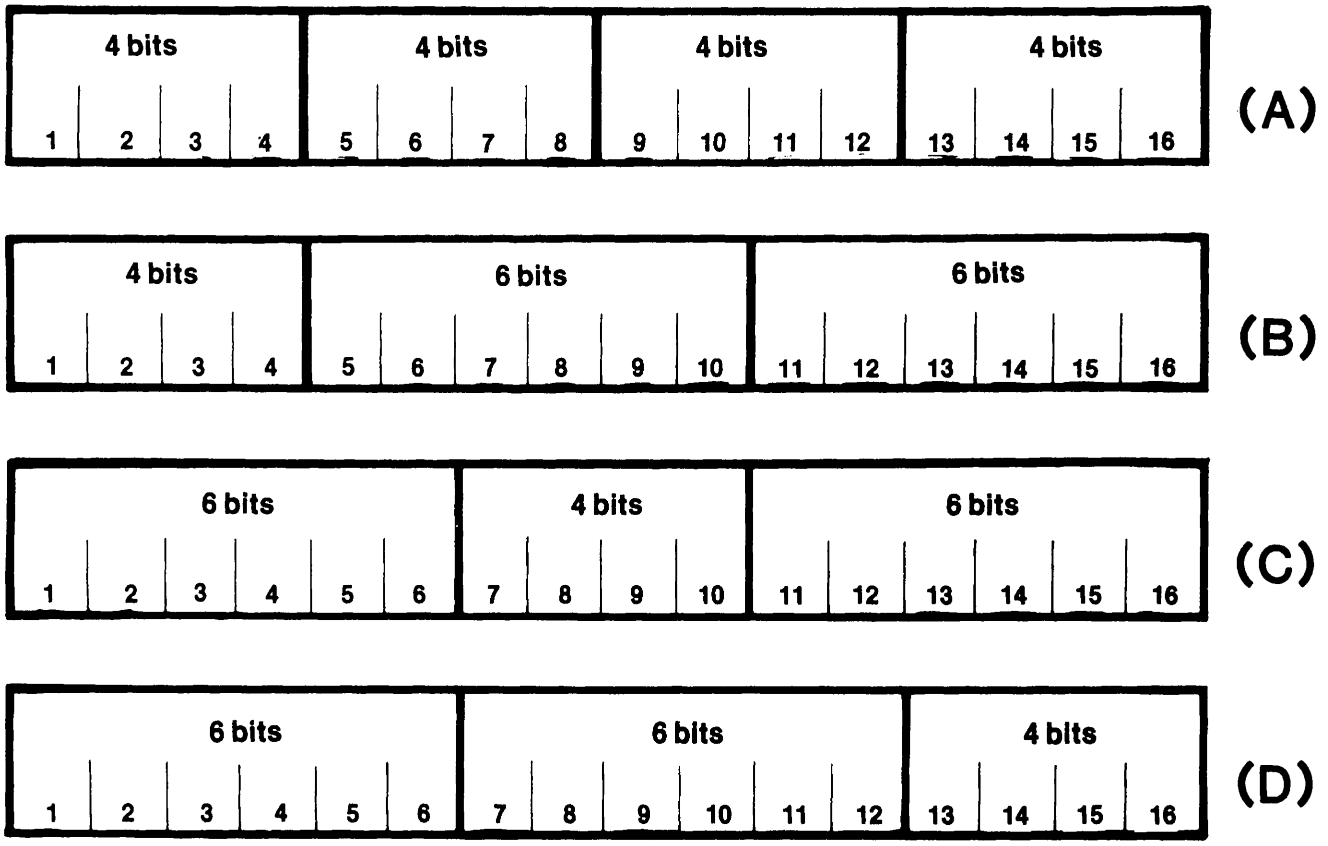
Connecting nine-track drives to six-bit byte sized computers presents no problems when writing because the extra two bits can be set to 0. Read operations, however, will almost certainly cause problems with loss of data. On eight-bit byte computers, seven-track drives present no problems when reading, but a write operation will cause a loss of data because two bits are ignored.

Seven-track drives may be connected to Prime systems. There are several methods of communicating with seven-track drives, including character-code translation (previously discussed) and binary packing/unpacking.

Numeric data occupies 16 bits (two bytes or one word) on Prime systems. The range of values is -32768 to +32767. A computer with six-bit bytes (connected together two per word) can only store numbers in the range -2048 to +2047. To translate numeric information, binary packing/unpacking is employed.

A computer word of 16 bits can be divided into either three or four pieces, each six bits or fewer. The choices are shown in Figure 1-5.

The three-frame formats do not need a fourth frame, so the next word can be placed starting in that frame. Any frame that only has four valid bits can record the other two bits as anything. As long as the computer is told what format is being used, it will correctly ignore any extra bits. Of course, the problem with the binary formats is that the computer words which were 16 bits long are now three seven-track tape frames long which still occupy three eight-bit bytes or 24 bits total, internal to the computer's memory. The unpacking is usually performed inside the computer so that records which were "n" bytes long are expanded to 3*n/2 bytes for three-frame formats, and to 2*n bytes for the four-frame format.



Binary Packing/Unpacking Formats
(A) 4444 Format
(B) 466 Format
(C) 646 Format
(D) 664 Format (commonly used by Prime)

Figure 1-5

2 Record Formats

INTRODUCTION

Chapter 1 dealt with the physical aspects of magnetic tapes, such as bits, frames, records, files, tracks, and density. This chapter deals with logical records and logical record length.

Records, whether stored on magnetic disks or tapes, cards or paper tape, come in one of two forms: either fixed-length or variable-length. Fixed-length records are all the same size. If user data does not completely fill a fixed-length record, empty character positions are usually set to blanks or 0's. Variable-length records, however, can range in size from zero characters to some maximum. Both fixed- and variable-length records may be blocked on magnetic tapes.

Under PRIMOS, the maximum size of a magnetic tape transfer is limited to 6144 words (or 12,288 bytes or characters). Thus, the maximum size of a fixed-length record is 12,288 characters if the blocking factor is 1. If the record size is decreased, the blocking factor may increase: a record size of 80 characters permits a blocking factor of 153 with 48 characters wasted.

Variable-length records can be written in many different forms: IBM standard variable, American National Standards Institute (ANSI) standard variable, IBM standard spanned, ANSI standard spanned, Prime variable-length and break. Each of these formats will be discussed in this chapter.

Standard Fixed-Length Records

The maximum size of fixed-length records under PRIMOS is 12,284 characters. Figure 1-2 in Chapter 1 illustrates this format. All records in a file containing fixed-length records must be the same size except for the last record which may be smaller.

IBM Standard Variable-Length Records

The maximum size of an IBM variable record under PRIMOS is 12,280 characters. Smaller maximum size limits result in larger possible blocking factors. The format for this type of record is a four-character block control word (BCW) followed by records. As shown in Figure 2-1, each record is composed of a four-character record control word (RCW) and data.

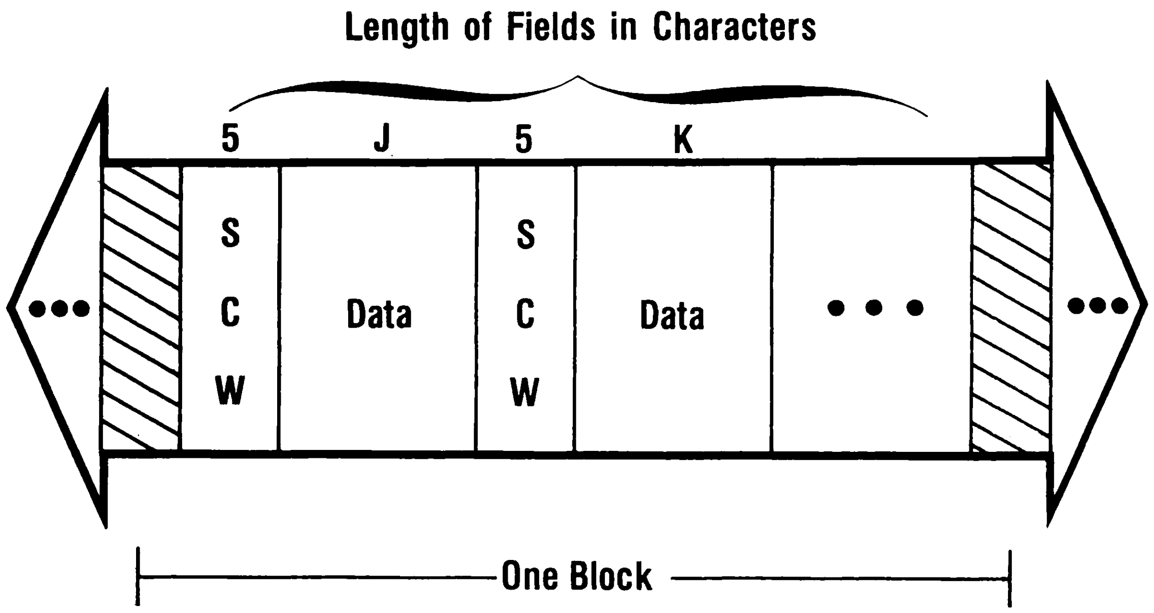
In the diagram, the data portion of one record is given as "J" while the other data record's length is given as "K". These two lengths are not necessarily equal. The only constraint is that the sum of the BCW, all of the RCW's, and all of the data portions must not exceed 12,288 bytes.

The BCW contains the length of the entire physical block or record. The length is stored as an unsigned 16-bit integer in the first two bytes. The second two bytes are reserved and must be set to 0. It is important to note that the BCW includes the length of the BCW itself, which is four bytes. Also, on IBM systems, this number should be between 8 and 32,760. The RCW has the same format as the BCW, but this length is the length of each logical record plus the length of the RCW (four bytes).

ANSI Standard Variable-Length Records

The maximum size of an ANSI variable record under PRIMOS is 12,284 characters. Smaller maximum size limits result in larger possible blocking factors. This format is similar to the IBM variable-length format except that there is no block control word (BCW). As shown in Figure 2-2, each record is still composed of an RCW plus a data portion.

In the diagram, the data portion of one record is given as "J" while the other data record's length is given as "K". These two lengths are not necessarily equal. The only constraint is that the sum of the RCW's and all of the data portions must not exceed 12,288 bytes. The format of the ANSI RCW is not the same as IBM's RCW. The length is an integer, recorded as a four-byte character string, which completely fills the RCW. There are no corresponding reserved bytes as there are in the IBM format.

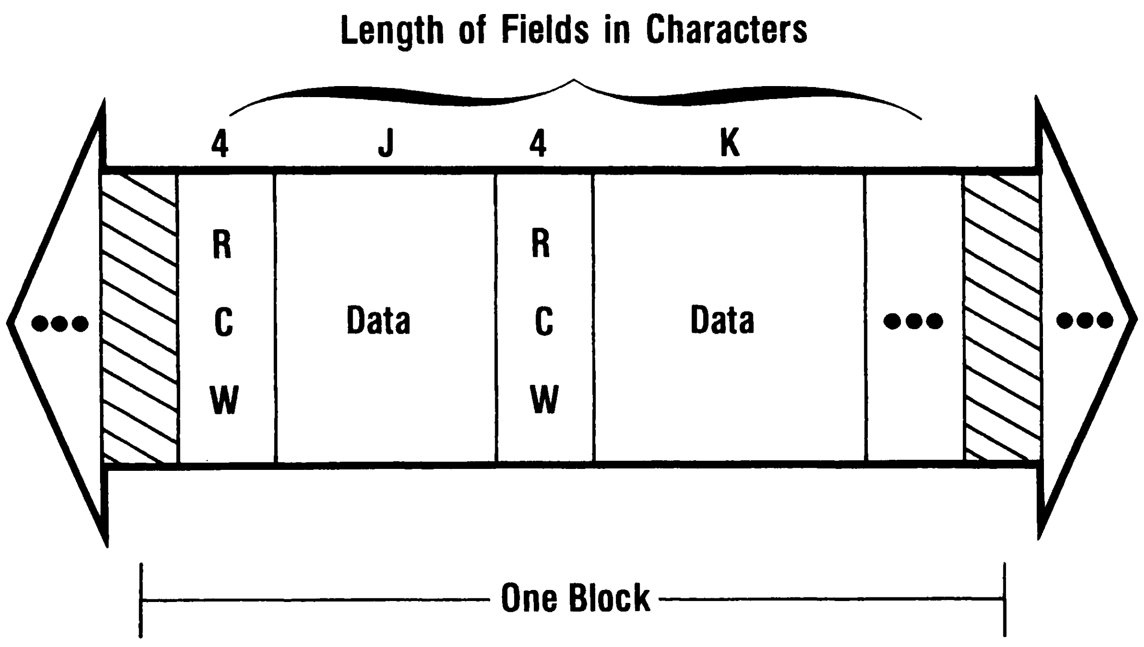


■ File Marker

□ Beginning and/or End of Tape Marker

▨ Inter-Record Gap

IBM Standard Variable-Length Records
Figure 2-1



■ File Marker

▩ Beginning and/or End of Tape Marker

▨ Inter-Record Gap

ANSI Standard Variable-Length Records
Figure 2-2

IBM Standard Spanned Variable-Length Records

Spanned records are similar to the standard variable-length records. Logical record lengths greater than the PRIMOS physical length maximum are possible because a logical record may be continued over two or more physical records. In other words, a 100,000-byte logical record would be written on tape as eight physical tape blocks, each containing the 12,288-byte maximum, and a ninth block containing some fragment. Each "piece" of the logical record is called a segment. Instead of a record control word, each segment is preceded by a segment control word (SCW). The SCW contains two pieces of information: the length of the data segment, including the SCW itself, and a segment descriptor. The format of the IBM SCW is shown in Figure 2-4. The SCC within the segment descriptor indicates whether the associated segment is the first segment of the logical record, the last segment, a middle piece, or the entire logical record. Figure 2-3 illustrates this IBM record format.

Spanned records may be blocked. However, no more than one segment of a record may appear in a block. In Figure 2-3, the data segment with length "J" would be the last segment of a record, while the data segment with length "K" is either the first or only segment of that record.

The SCC is one byte (eight bits) long. The first (high-order) six bits are 0 and the last two bits specify the relative position of the segment. Table 2-1 lists the SCCs and the corresponding segment positions.

Like IBM variable-length records, IBM spanned records contain a block control word. The format is exactly the same as in variable-length records.

ANSI Standard Spanned Variable-Length Records

Spanned records are similar to the standard variable-length records. Logical record lengths greater than the PRIMOS physical length maximum are possible because a logical record may be continued over two or more physical records. In other words, a 100,000-byte logical record would be written on tape as eight physical tape blocks, each containing the 12,288-byte maximum, and a ninth block containing some fragment. Each "piece" of the logical record is called a segment. Instead of a record control word, each segment is preceded by an SCW. The SCW contains two pieces of information: the length of the data segment, including the SCW itself, plus a segment descriptor. The SCC within the segment descriptor indicates whether the associated segment is the first segment of the logical record, the last segment, a middle piece, or the entire logical record. The ANSI spanned format is very similar to the IBM spanned format, except that there are no block control words on each physical block and the segment control word has a slightly different form. Figure 2-5 illustrates the ANSI spanned format.

Spanned records may be blocked. However, no more than one segment of a record may appear in a block. In Figure 2-5, the data segment with length "J" would be the last segment of a record while the data segment with length "K" is either the first or only segment of that record. Note that the ANSI segment control words are each five bytes long. The format of the ANSI SCW is shown in Figure 2-6.

The SCC is one byte (eight bits) long and is a character instead of a binary number as in the IBM SCC, and indicates the relative position of the segment within a multisegment record. Table 2-2 lists the SCCs and the corresponding segment positions.

Prime Variable-Length Format

Prime variable-length format provides Prime FORTRAN and PL/I users with a variable-length capability that is easier to use than the standard formats. In this format, a physical record is equivalent to a logical record (a blocking factor of one). Whatever is contained in a physical tape record actually becomes one logical record. No block, record, or segment control words are necessary, but the maximum record size is limited to 12,288 bytes.

Break Format

Break format is similar to Prime variable-length format except that logical records are not constrained to "fit" into one physical record. The end of a logical record is "signalled" by a carriage-return/line-feed (CRLF). Two consecutive CRLFs indicate a completely blank line. No record, block, or segment control words are necessary, and a logical record has no size limit. The physical tape records are, however, limited to the Prime maximum of 12,288 bytes.

MAGNETIC TAPE STANDARDS

There are two standards for magnetic tape formats. One is a de facto standard used by IBM and recognized by many manufacturers when interchanging tapes with IBM computers. The other is a more formal standard, developed by ANSI.

IBM Format

Use of the IBM standard implies that nine-track tapes are recorded in EBCDIC, while seven-track tapes are recorded in BCDIC. The density for nine-track tapes is 800, 1600, or 6250 bpi while seven-track tapes are recorded at either 200, 556, or 800 bpi. Variable-length IBM records are preceded by an RCW of four bytes. The first two bytes contain the length of the record including the RCW. The second two bytes are not used. Variable-length records may be blocked. Spanned records are preceded by an SCW of four bytes. The first two bytes indicate the length of the record segment including the SCW.

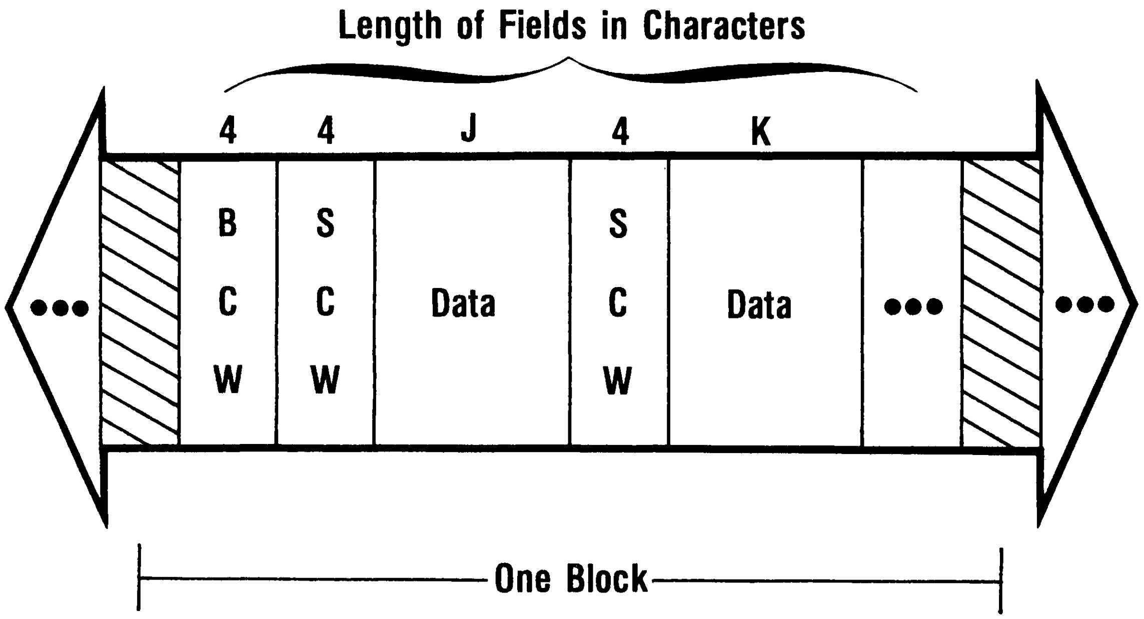
The first four bytes of every variable or spanned block on IBM tapes contain a BCW. The BCW contains the length of the block plus the BCW.

ANSI Format

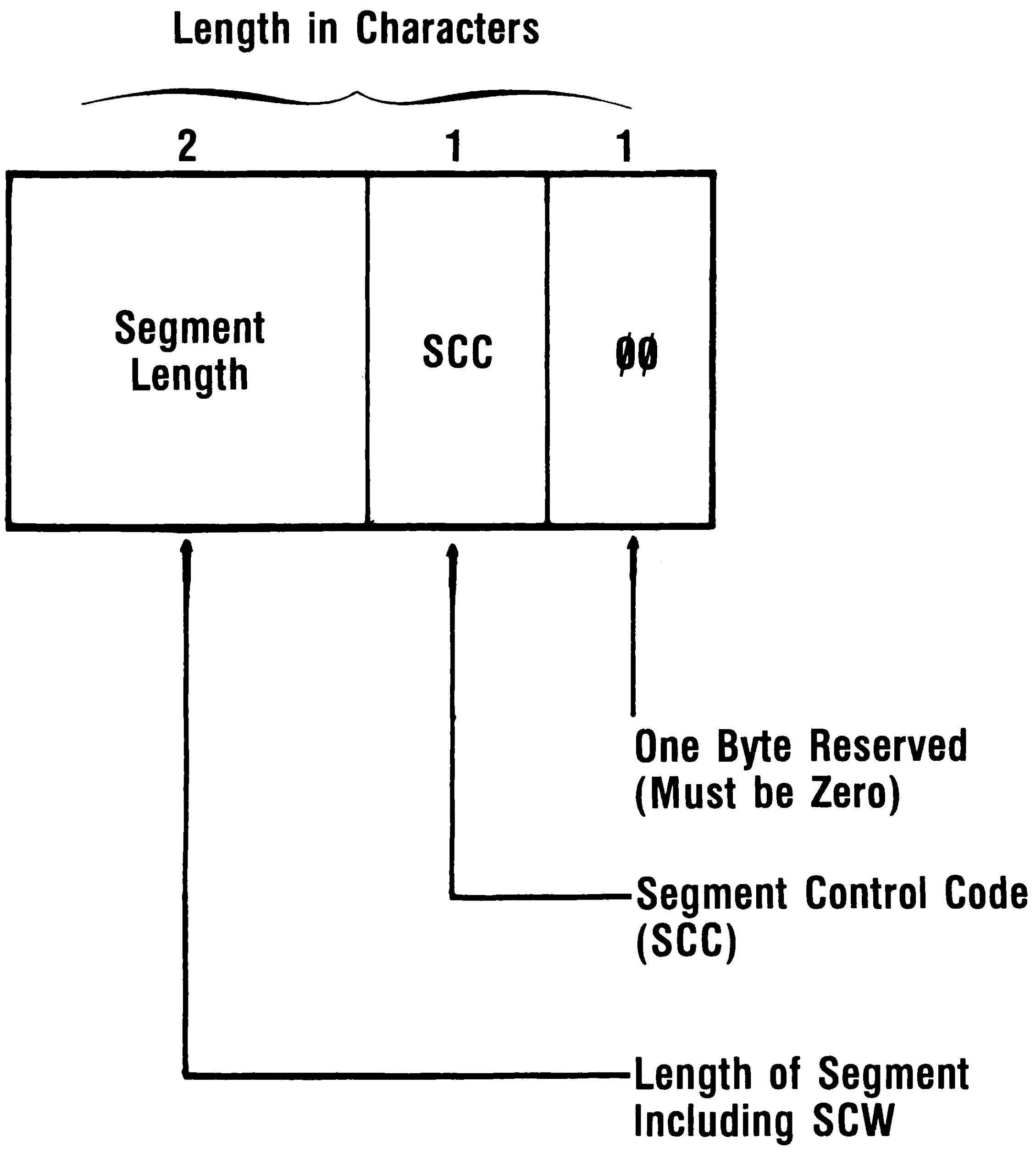
ANSI tapes are very similar to IBM tapes except that support of seven-track tapes is not specified. In addition, there is no BCW for variable or spanned records and the RCWs and SCWs are slightly different. The RCW is four bytes long and contains the length of the record including the RCW as a four-byte-fixed binary (31) or integer*4-number. The SCW is five bytes long.

ANSI standard tapes are written using the ASCII character set. Binary tapes contain binary numbers. Different computer manufacturers represent binary numbers in different ways. For example, some computers use ones complement notation, while others use twos complement notation. The IBM System 360/370 computers have a four-byte floating point format which uses base 16. The exponent occupies seven bits of the first byte. Prime has a four-byte binary floating-point format in which the exponent occupies the last byte. Because of these differences, binary tapes are usually not transportable from one manufacturer's computer to another.

Consecutive segments of a spanned record can be split between two tape reels because of their segment indicators. The end of information on a tape that is part of a multireel file is signalled by a tape marker. The end of information on the last tape volume of a multireel file or a single volume is indicated by two tape markers written one immediately following the other. Files are separated by a single tape marker. The first file on a reel, which is the first or only reel, is not preceded by a tape marker.



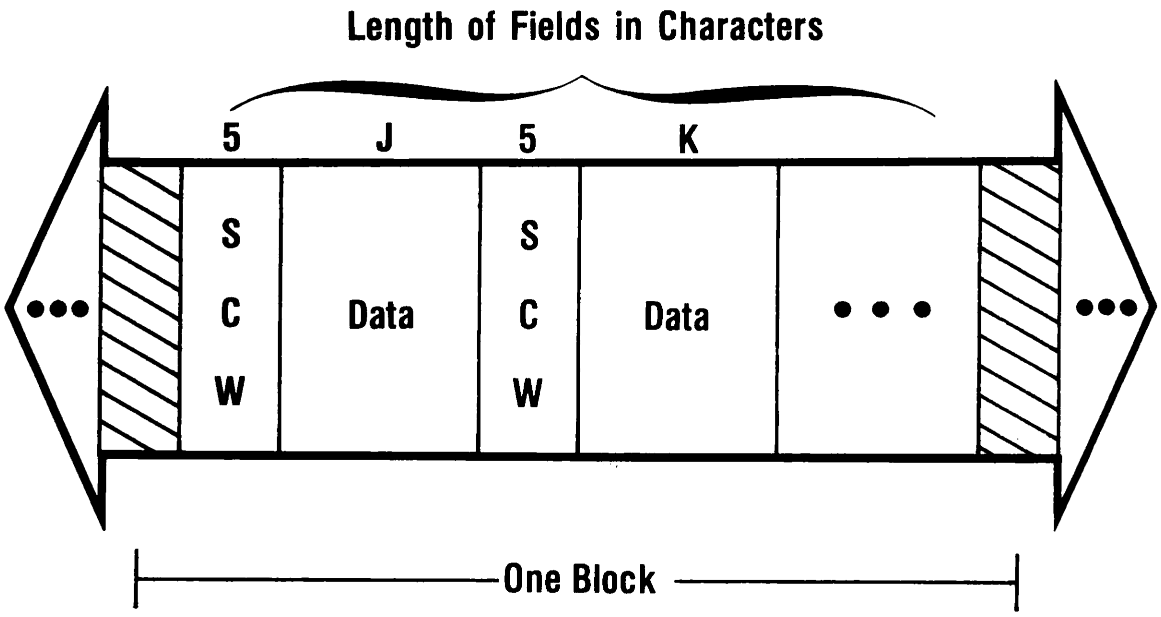
IBM Standard Spanned Variable-Length Records
Figure 2-3


Length in Characters
2
1
1
Segment Length
SCC
ØØ
One Byte Reserved (Must be Zero)
Segment Control Code (SCC)
Length of Segment Including SCW

IBM Segment Control Word (SCW)
Figure 2-4

Table 2-1
IBM Segment Control Codes

Code	Relative Position of Segment
00	Complete logical record
01	First segment of a multisegment record
10	Last segment of a multisegment record
11	Middle segment of a multisegment record (not the first or the last)

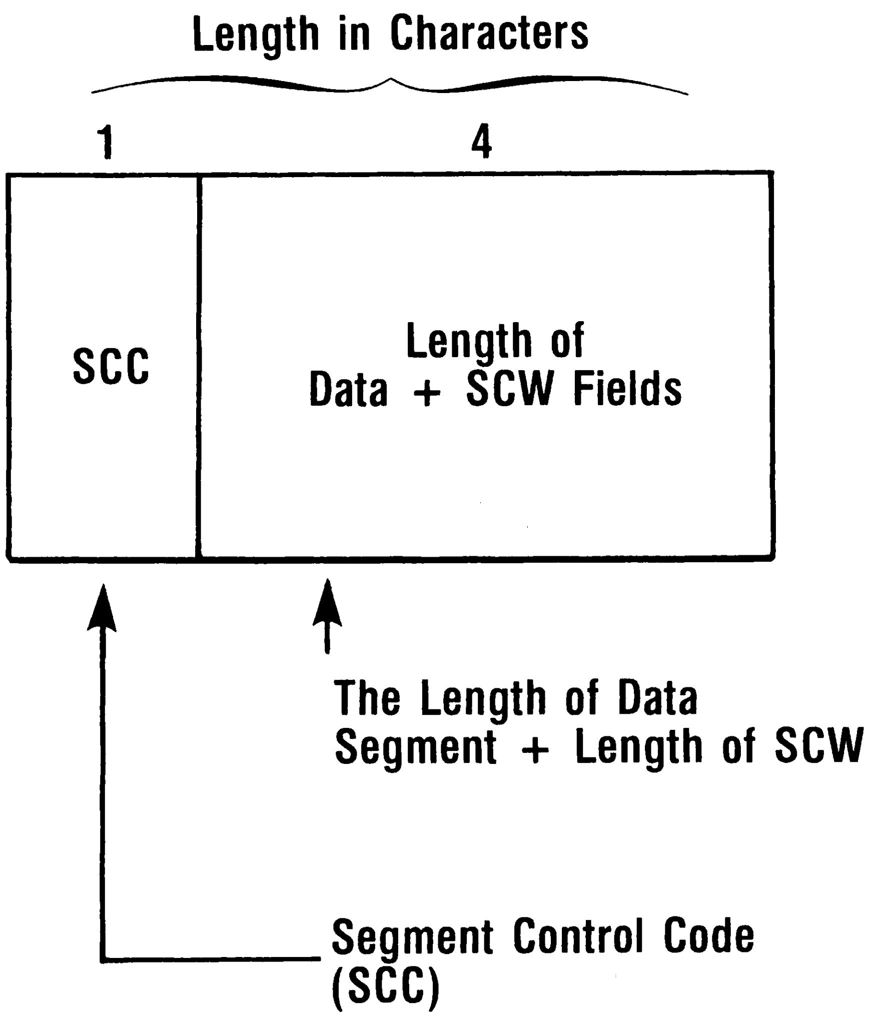


■ File Marker

▨ Beginning and/or End of Tape Marker

▧ Inter-Record Gap

ANSI Standard Spanned Variable-Length Records
Figure 2-5


Length in Characters
1
4
SCC
Length of Data + SCW Fields
The Length of Data Segment + Length of SCW
Segment Control Code (SCC)

ANSI Segment Control Word (SCW)
Figure 2-6

Table 2-2
ANSI Segment Control Codes

Code	Relative Position of Segment
0	Complete logical record
1	First segment of a multisegment record
2	Middle segment of a multisegment record (not the first or the last)
3	Last segment of a multisegment record

3 Magnetic Tape Labels

INTRODUCTION

Tape labels are used to identify and provide processing information about tape reels and the files they contain. As with record standards, there are basically two tape label standards. The first is the de facto standard of IBM. The second is an ANSI standard. The two standards are almost the same, differing only in the naming or length of certain fields within a label. The major difference is that IBM labels are written in either nine-track EBCDIC or seven-track BCDIC, while ANSI labels are written in nine-track ASCII only.

Labels are just tape records. They are usually 80 characters long. The first four characters identify the label type. There are several different label types:

- Volume labels - VOLn, UVLn, EOVn
- Header labels - HDRn, UHLn
- Trailer labels - EOFn, UTLn

Volume labels identify the reel of tape, the owner of the reel, and the accessibility of the reel. Other information may be stored in user fields.

<u>Header</u> and <u>trailer labels</u> identify and provide information about each <u>file on the tape</u>. Such information includes the file-identifier, the creation and expiration dates, block counts and set, sequence, generation and version numbers.

<u>VOLUME LABELS</u>

There are three types of volume labels: VOLn, UVLn, and EOVn labels.

<u>VOLn Labels</u>

VOLn labels appear at the very beginning of a reel of tape. There may be up to nine of these labels, numbered VOL1 through VOL9. If the tape has any standard labels at all, it must have a VOL1 label as its first record. There need not be more than one VOLn label, but if there are any additional VOLn labels, they must appear in sequence, VOL2, VOL3, etc. Certain IBM operating systems allow only up to eight of these labels (VOL1 - VOL8). The format of the ANSI standard VOL1 label is shown in Figure 3-1. The format of the IBM standard VOL1 label is shown in Figure 3-2.

<u>Other Volume Labels, User File Header, and User File Trailer Labels</u>

The format of the IBM and the ANSI VOL2 through VOL9, as well as the UVL1 through UVL9, UHLn, and UTLn labels are shown in Figure 3-3. Some IBM systems limit the number of VOL and UVL labels, and it is best to check first when writing tapes that are targeted for IBM systems. The UVL labels immediately follow the last of the VOL labels on the tape. The VOL-UVL label group is called the <u>volume header label group</u>.

Character Position	Length	CONTENTS OF FIELDS
1 2 3 4	4	The String "VOL 1"
5 6 7 8 9 10	6	**Volume Serial Identifier (VOLSER)** — identifies a specific reel of tape.
11	1	**Accessibility** — an installation-dependent field indicating the type of reel access. A space indicates no restrictions. Other characters may indicate read-only, execute-only, etc.
12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37	26	Reserved by ANSI for future information; recorded as spaces.
38 39 40 41 42 43 44 45 46 47 48 49 50 51	14	**Owner-Identification** — identifies the owner of the reel.
52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79	28	Reserved by ANSI for future information; recorded as spaces.
80	1	**ANSI Standard Version** — recorded as a "3" to indicate that Prime software conforms to the X3.27-1978 standard.

ANSI Standard VOL1 Label
Figure 3-1

Character Position	Length	CONTENTS OF FIELDS
1	4	The String "Vol. 1"
2		
3		
4		
5	6	**Volume Serial Identifier (VOLSER)** — identifies a specific reel of tape.
6		
7		
8		
9		
10		
11	31	Reserved by IBM for future information; recorded as spaces.
12		
13		
14		
15		
16		
17		
18		
19		
20		
21		
22		
23		
24		
25		
26		
27		
28		
29		
30		
31		
32		
33		
34		
35		
36		
37		
38		
39		
40		
41		
42	10	**Owner Identification** — identifies the owner of the reel.
43		
44		
45		
46		
47		
48		
49		
50		
51		
52	29	Reserved by IBM for future information; recorded as spaces.
53		
54		
55		
56		
57		
58		
59		
60		
61		
62		
63		
64		
65		
66		
67		
68		
69		
70		
71		
72		
73		
74		
75		
76		
77		
78		
79		
80		

IBM Standard VOL1 Label
Figure 3-2

Character Position	Length	CONTENTS OF FIELDS
1–4	4	The String ``VOLn`` (n = 2 - 9). ``UVLn`` (n = 1 - 9). ``UHLn`` (n = 1 - 9) or ``UTLn`` (n = 1 - 9)
5–80	76	This is a user-specified field. It is not checked or processed by **Prime** magnetic tape subroutines or subsystems.

IBM & ANSI VOL2-VOL9 and UVLn, UHLn, and UTLn Labels
Figure 3-3

FILE HEADER LABELS

There are two types of FILE HEADER labels: the HDRn and the UHLn labels.

HDR1 and HDR2 Labels

A labelled tape must include at least a HDR1 label before each file. Up to nine HDRn labels may appear, but just like VOLn labels, they must appear in order. The first UHLn label must appear, if at all, after the last HDRn label. A collection of HDRn and UHLn labels appearing together is called a file header label group. Figure 3-4 shows the format of the ANSI standard HDR1 label. Figure 3-5 shows the format of the IBM standard HDR1 label. The HDR2 label contains additional information about the tape file. Once again, the IBM and ANSI labels have some differences. Figure 3-6 shows the format of the ANSI standard HDR2 label. Figure 3-7 shows the format of the IBM standard HDR2 label.

Other File Header Labels

All of the other HDRn (n = 3, 4, 5, 6, 7, 8, and 9) labels and all of the UHLn labels follow the same format as the VOLn (n = 3-9) and UVLn labels. The first four characters contain the label identifier, and the other 76 characters are reserved for the user. The beginning-of-file label group is terminated by a single file marker.

Character Position	Length	CONTENTS OF FIELDS
1 2 3 4	4	The String ''HDR1''
5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21	17	**File-Identifier** — identifies individual tape files.
22 23 24 25 26 27	6	**File-Set Identifier** — the same information as that in the volume-identifier field on the VOL 1 label. For multi-volume files, it contains the volume identifier of the first tape reel of the set.
28 29 30 31	4	**File-Section Number** — identifies the order of a particular reel in a multi-reel set (''0001''-''9999'').
32 33 34 35	4	**File-Sequence Number** — identifies the order of a file within a set of files created simultaneously (''0001'' - ''9999'').
36 37 38 39	4	**Generation-Number** — indicates file generation or edition (''0001''-''9999'') or spaces)
40 41	2	**Generation-Version** — indicates file version or printing (''00''-''99'' or spaces).
42 43 44 45 46 47	6	**Creation Date** — the day and year the file was created. The format is ''b̸yyddd'' where b̸ = a blank; yy = the year (''00''-''99''); ddd = the day of the year (''00''-''366'')
48 49 50 51 52 53	6	**Expiration-Date** — the day and year the file expires. The format is the same as that for the creation date field.
54	1	**Accessibility** — is installation dependent.
55 56 57 58 59 60	6	**Block Count** — recorded as ''000000''
61 62 63 64 65 66 67 68 69 70 71 72 73	13	**System Code** — identifies the computing system that wrote the file. **Prime** software writes the string ''PRIMOSb̸b̸b̸b̸b̸b̸b̸'' where the ''b̸'' 's represent spaces.
74 75 76 77 78 79 80	7	Reserved by ANSI for future information, recorded as spaces.

ANSI Standard HDR1 Label
Figure 3-4

Character Position	Length	CONTENTS OF FIELDS
1–4	4	The String "HDR1"
5–21	17	**Data Set-Identifier** — identifies individual tape files.
22–27	6	**Data Set Serial Number** — the same information as that in the **Volume-Identifier** field on the VOL 1 label. For multi-volume files, it contains the volume identifier of the first tape reel of the set.
28–31	4	**Volume Sequence Number** — identifies the order of a particular reel in a multi-reel set ("0001"-"9999").
32–35	4	**Data Set-Sequence Number** — identifies the order of a file within a set of files created simultaneously ("0001"-"9999").
36–39	4	**Generation-Number** — indicates file generation or edition ("0001"-"9999" or spaces).
40–41	2	**Generation-Version** — indicates file version or printing ("00"-"99" or spaces).
42–47	6	**Creation Date** — the day and year the file was created. The format is "b̸yyddd" where b̸ = a blank; yy = the year ("00"-"99"); ddd = the day of the year ("001"-"366")
48–53	6	**Expiration-Date** — the day and year the file expires. The format is the same as that for the Creation Date field.
54	1	**Accessibility** — is installation dependent.
55–60	6	**Block Count** — recorded as "000000"
61–73	13	**System Code** — identifies the computing system that wrote the file. **Prime** software writes the string "PRIMOSb̸b̸b̸b̸b̸b̸b̸" where the "b̸" 's represent spaces.
74–80	7	Reserved by IBM for future information; recorded as spaces.

IBM Standard HDR1 Label
Figure 3-5

Character Position	Length	CONTENTS OF FIELDS
1–4	4	The String "HDR2"
5	1	**Record Format** — "F" = fixed length; "D" = variable length; "S" = spanned length.
6–10	5	**Block Length** — maximum number of characters in a block.
11–15	5	**Record Length** - actual length of fixed-length logical records; maximum length including RCWs of variable-length logical records; maximum length *excluding* SCWs of spanned logical records (a record length of "00000" may indicate a record length > "99999").
16–50	35	Reserved by ANSI for future information; recorded as spaces.
51–52	2	**Buffer-Offset** — length in characters of any field preceeding the first record in a block.
53–80	28	Reserved by ANSI for future information; recorded as spaces.

ANSI Standard HDR2 Label
Figure 3-6

Character Position	Length	CONTENTS OF FIELDS
1–4	4	The String "HDR2"
5	1	**Record Format** — the format is the same as that for the ANSI HDR2 field. "U" is also accepted for undefined length.
6–10	5	**Block Length** — same as corresponding ANSI HDR1 fields.
11–15	5	**Record Length** — same as corresponding ANSI HDR1 fields.
16	1	**Tape Density** "0" = 200 BPI; "3" = 1600 BPI; "1" = 556 BPI; "4" = 6250 BPI; "2" = 800 BPI;
17	1	**Data Set Position** "1" = multi-volume file; "0" = single-volume or first part of multi-volume file
18–34	17	**JOB/JOB STEP ID** — not processed on input. The string "PRIMOSƀƀƀƀƀƀƀƀƀƀƀ" (ƀ = space) is written on output.
35–36	2	**Tape Recording Technique** — for 7-track tapes only; set according to the tape parity.
37	1	**Control Character** — "A" = ASCII; "M" = IBM; Blank = none.
38	1	**Reserved** — recorded as a space.
39	1	**Block Attribute** — "B" = blocked records; "S" = spanned records; "R" = blocked and spanned; blank = not blocked and not spanned.
40–80	41	Reserved by IBM for future information; recorded as spaces.

IBM Standard HDR2 Label
Figure 3-7

FILE AND VOLUME TRAILER LABELS

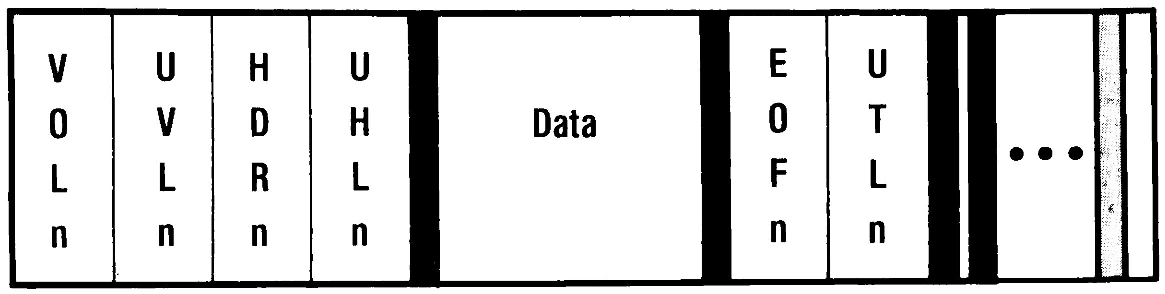
The EOF1 label follows exactly the same format as the HDR1 label except that the block count field contains the number of blocks that were written in the file. The EOF2 label follows exactly the same format as the HDR2 label. EOF3-EOF9 and UTLn labels follow the same format as VOL3-VOL9 and UVLn labels. The first four characters identify the label type and the other 76 characters are reserved for the user. There is no limit on the number of UHLn or UTLn labels that may be used.

The EOVn labels are written whenever it becomes necessary to mount another tape when a currently mounted tape has reached its EOT marker. While it is theoretically possible for a user to provide EOV3-EOV9 as well as UTLn labels, Prime software will not allow this at this time. On an EOT during output, the EOV1 and possibly EOV2 labels are written, followed by the appropriate number of file markers and then the tape is rewound. The EOV1 format is the same as the EOF1 format. The EOV2 label is the same as the EOF2 label except for the "EOV2" identifier.

LABEL GROUPS

Tape labels are written together in groups called _label groups_. The first information that must appear on a labelled tape is the _volume header label group_. Volume labels consist of at least the VOL1 label. VOL2 through VOL9 labels may also be present followed by UVL1 through UVL9, if desired. The next information that appears on the tape is the first file's _file header label group_, which consists of a HDR1 label possibly followed by HDR2 through HDR9 and any number of UHL labels. These labels are followed by a file marker, then the actual data records for the file. The file is followed by: a file mark, an _end-of-file label group_ consisting of EOF1 and possibly EOF2 through EOF9, and then any number of UTLn labels. If the file does not fit on one physical tape, the EOFn labels are replaced by EOVn labels. The last labels for a file are followed by a single file marker unless the file is the last file on the tape and ends before the end-of-tape marker, in which case two marks are written. The label and tape file formats are shown in Figures 3-8, 3-9, 3-10, and 3-11.

If all the file header and file trailer groups on a tape contain only HDR1 and EOF1/EOV1 labels, then the tape is said to have _Level 1 labels_ only. If the tape contains HDR2 and EOF2/EOV2 labels in addition to the Level 1 labels, the tape is said to contain _Level 2 labels_. A tape that contains Level 2 labels and HDRn, EOFn, EOVn (n>2), UHLn, or UTLn labels is said to contain _Level 3 labels_. The terms _Level 1 tape_, _Level 2 tape_, and _Level 3 tape_ are also sometimes used. A _Level 4 tape_ may contain all Level 3 labels plus spanned records.

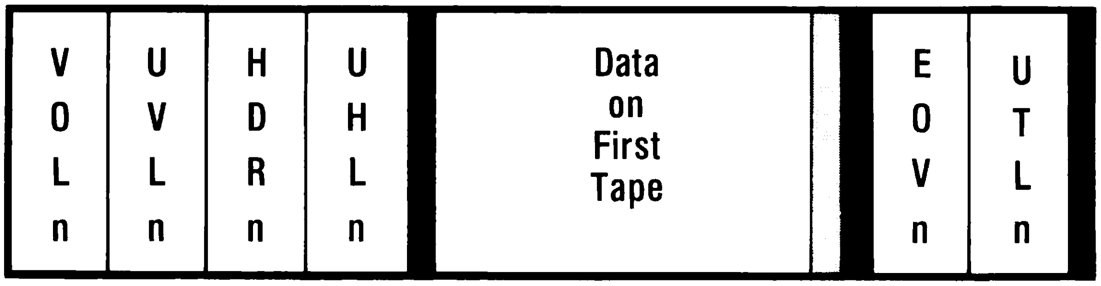


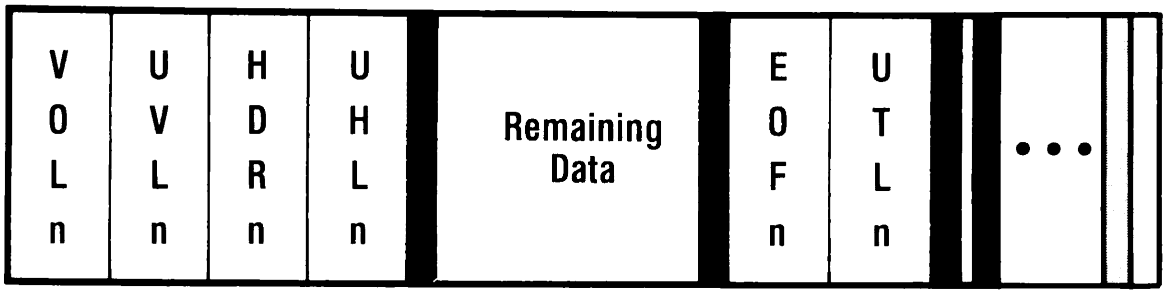
■ File Marker

□ Beginning and/or End of Tape Marker

▨ Inter-Record Gap

Tape Label and File Configuration — Single Reel, Single File
Figure 3-8



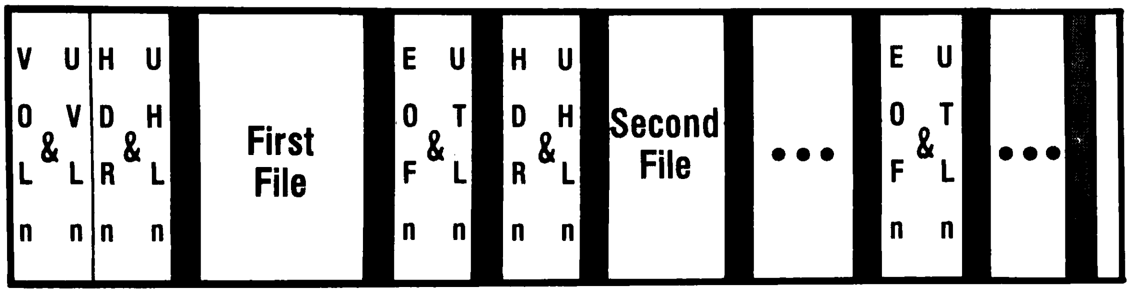


■ File Marker

□ Beginning and/or End of Tape Marker

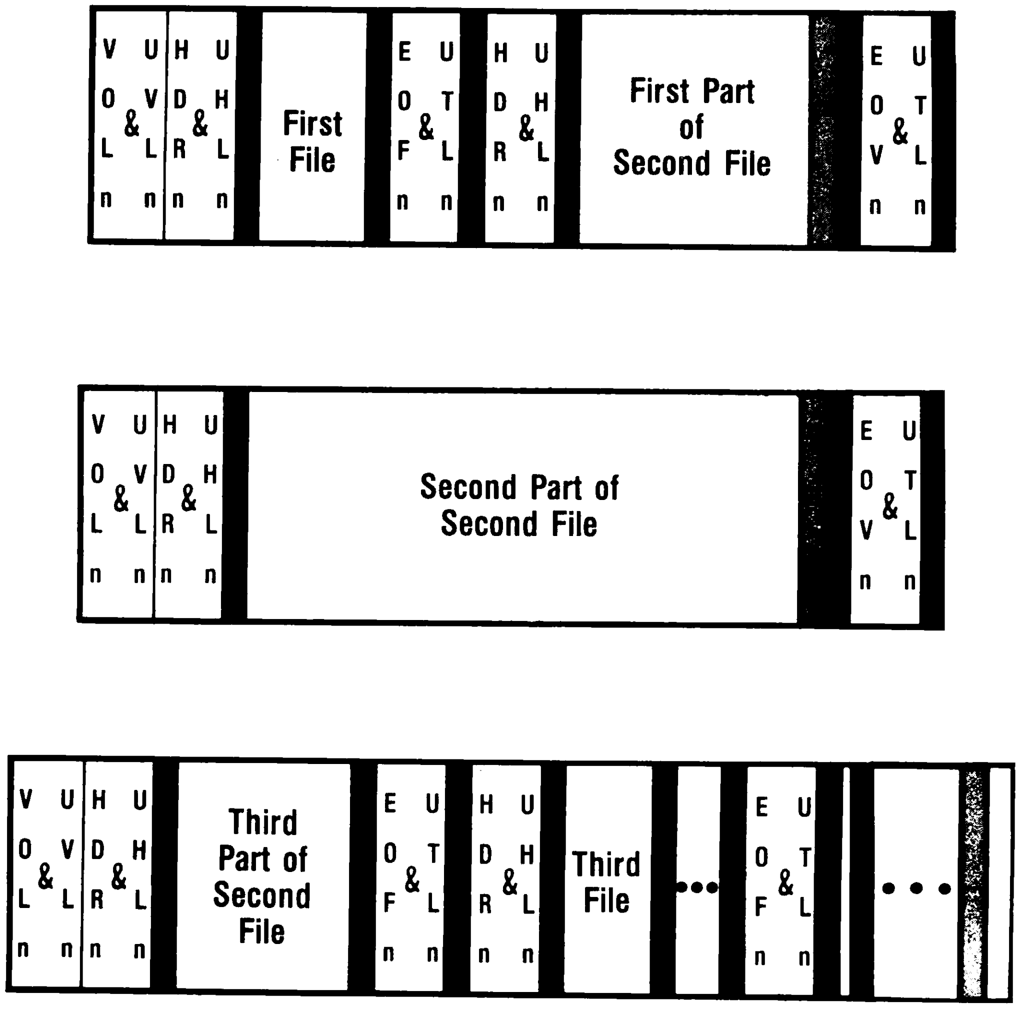
▨ Inter-Record Gap

Tape Label and File Configuration — Multiple Reels, Single File
Figure 3-9



■	File Marker
▒	Beginning and/or End of Tape Marker
▨	Inter-Record Gap

Tape Label and File Configuration — Single Reel, Multiple Files
Figure 3-10



■	File Marker
▒	Beginning and/or End of Tape Marker
▨	Inter-Record Gap

Tape Label and File Configuration — Multiple Reels, Multiple Files
Figure 3-11

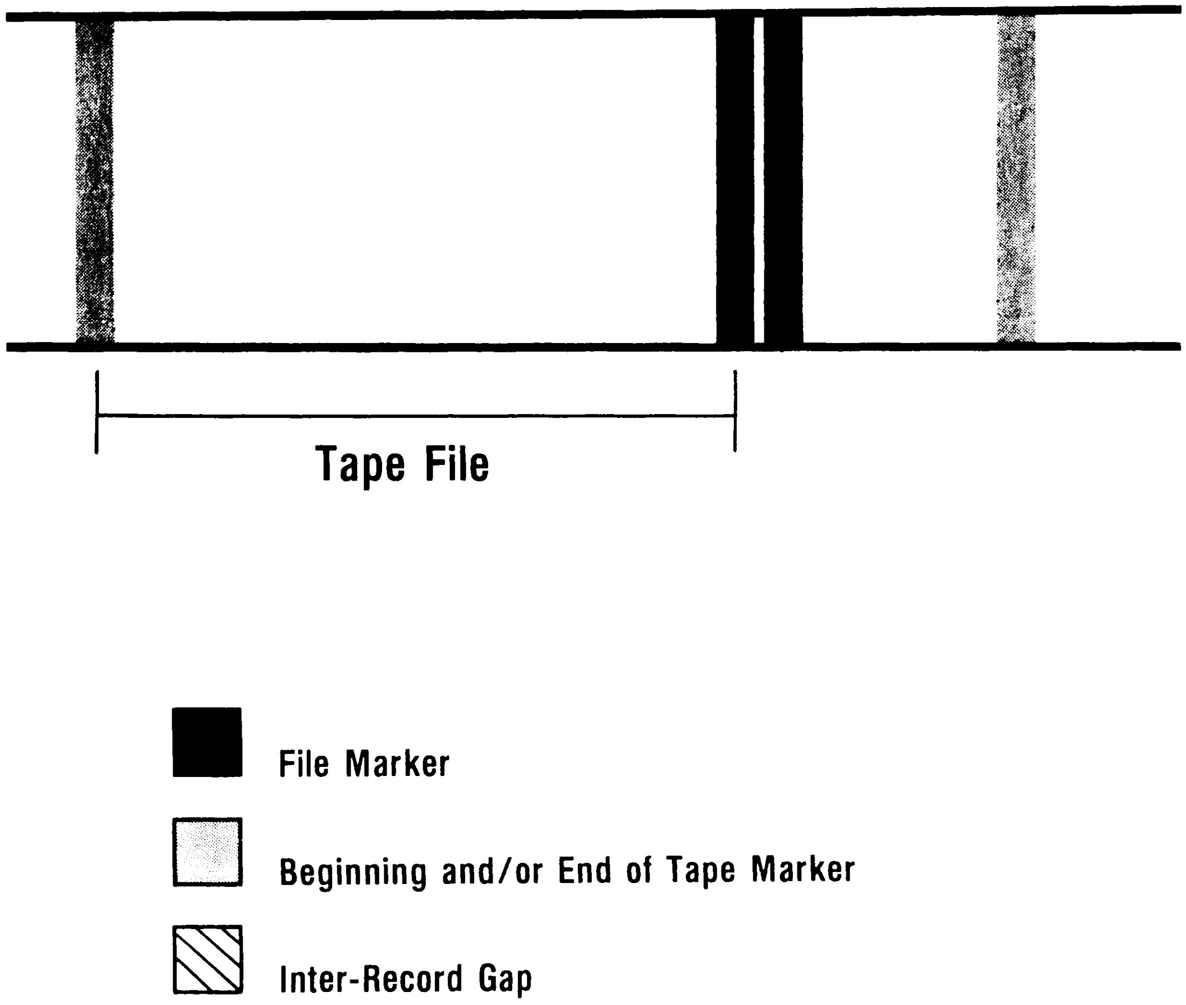
UNLABELLED TAPES

Unlabelled tapes also follow certain standard formats; nevertheless they are harder to process than labelled tapes. There are two reasons for this:

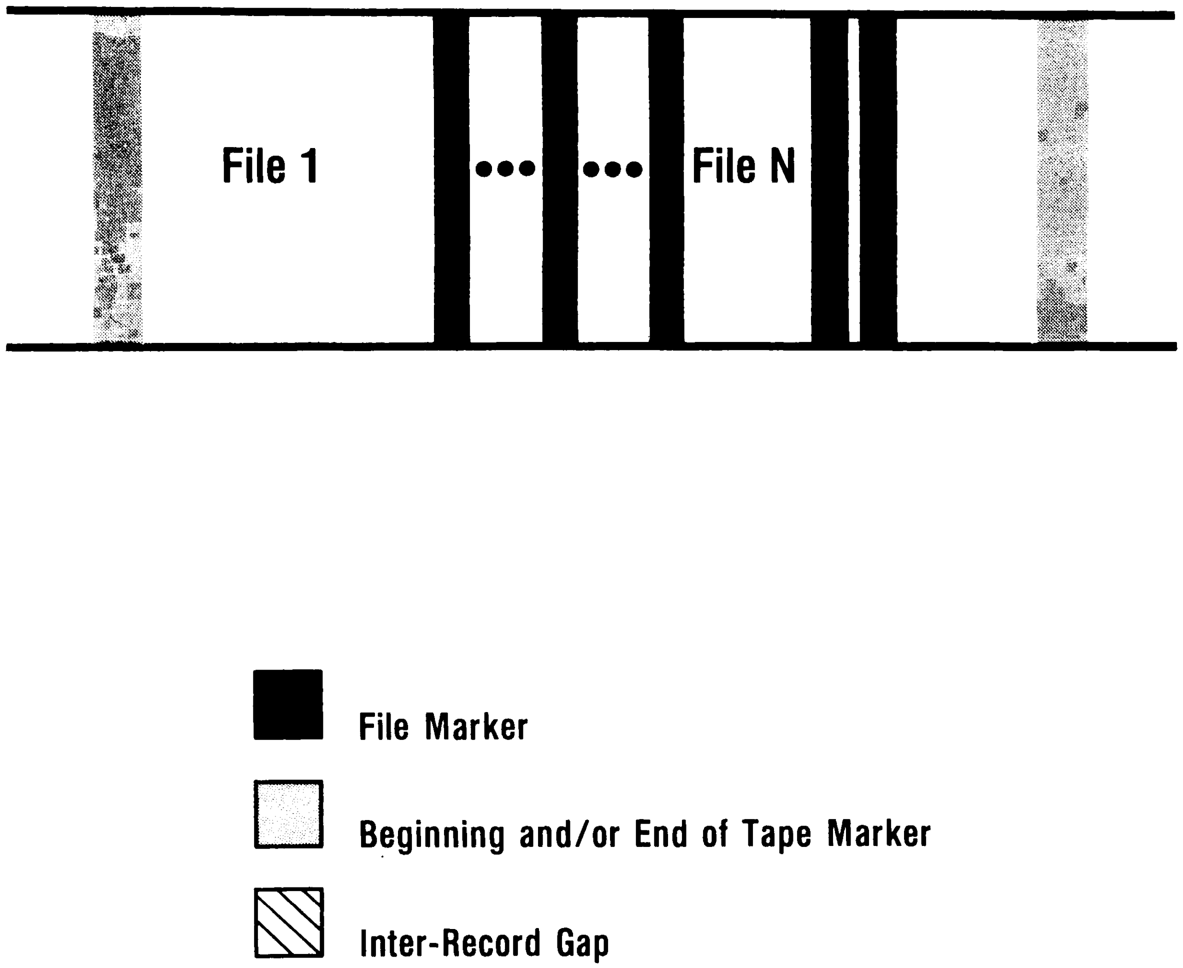
- Tape labels contain much of the information about record formats and lengths, blocking factors, and multireel operations.

- Tape labels provide some protection for error checking and delimit data files better. That is, the labels contain information about how many tape blocks are contained in the file, and files are delimited by four file markers -- two at each end (one before and after the HDRn label group, another set around the EOFn or EOVn group).

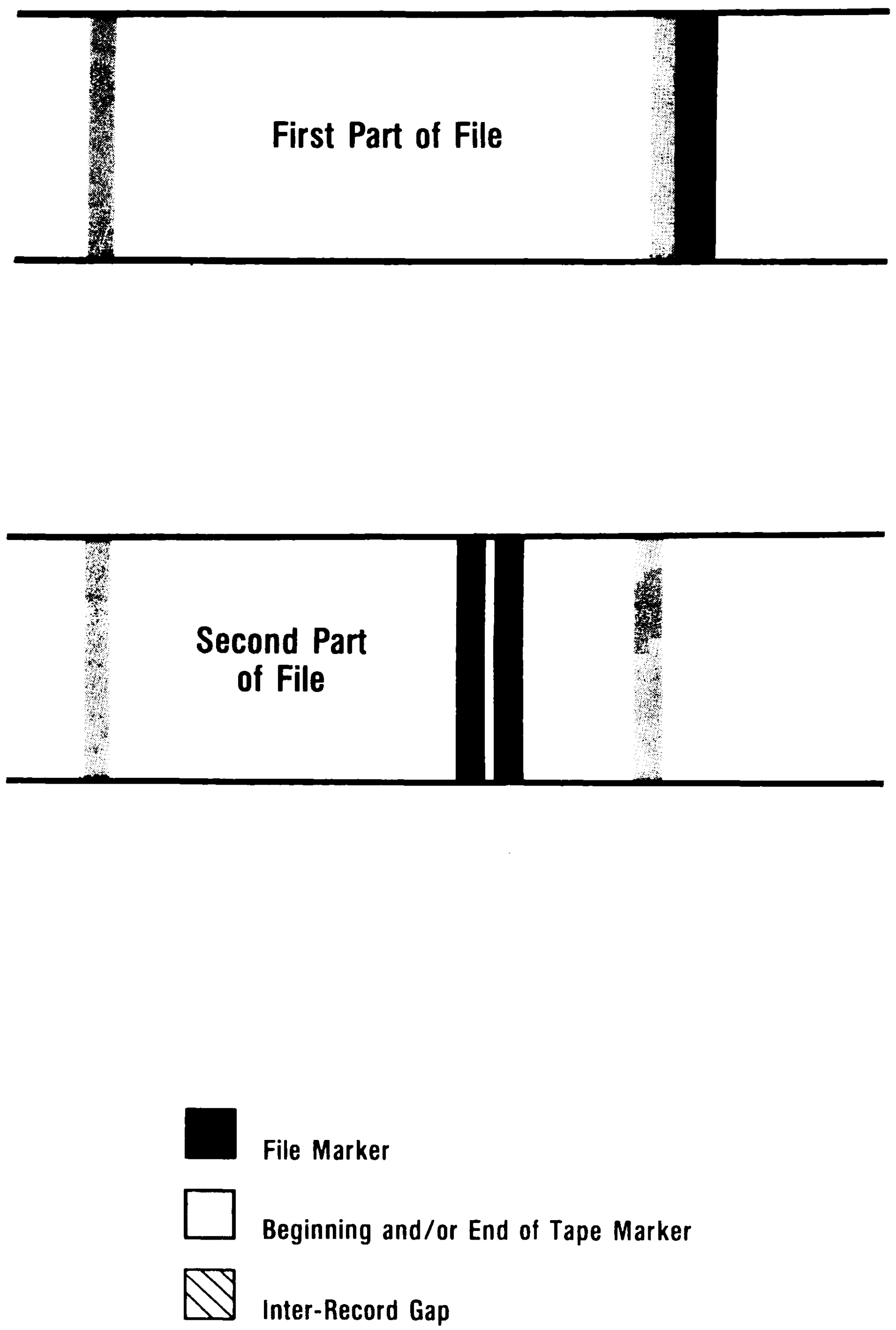
On unlabelled tapes, the end of a file is signalled by a single file marker. If the end of a file is also the end of all the data on the tape, another file marker is written. If the end-of-tape marker is detected when writing, a single file marker is written whether or not the entire file has been written to tape. It is assumed that another tape must be mounted to complete processing. Unlabelled tape file formats are shown in Figures 3-12, 3-13, 3-14, and 3-15.



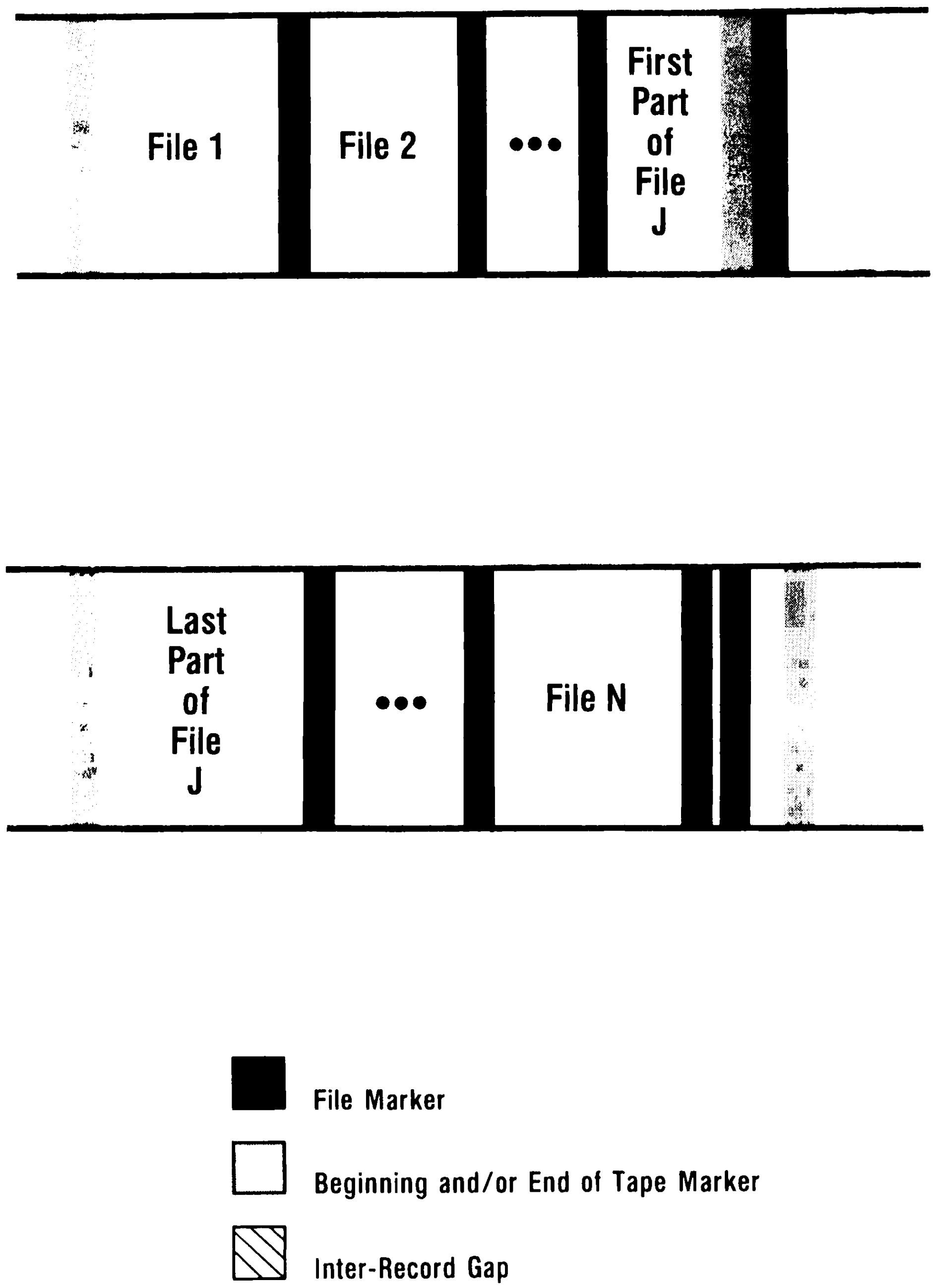
Unlabelled Tape File Configuration — Single Reel, Single File
Figure 3-12



Unlabelled Tape File Configuration — Single Reel, Multiple Files
Figure 3-13



Unlabelled Tape File Configuration — Multiple Reels, Single File
Figure 3-14



Unlabelled Tape File Configuration — Multiple Reels, Multiple Files
Figure 3-15

PART II

PRIMOS Magnetic Tape Commands

4
User Control of Tape Drives

INTRODUCTION

Various PRIMOS commands allow you to control tape drives so that you may perform different magnetic tape operations. With the STATUS DEVICE command you can check on the availability of tape drives. This command is discussed at the end of this chapter. With the ASSIGN and UNASSIGN commands you can:

- Gain control of a physical or logical tape drive
- Request that the operator give you assistance when you need special features
- Relinquish control of a tape drive

THE ASSIGN COMMAND

The ASSIGN command allows you to gain complete control over a peripheral device — a magnetic tape drive. ASSIGN either indicates by number which physical tape drive you want or provides a description of a tape drive that meets your requirements. ASSIGN also allows you to make special requests of the operator; for example, removing the write-enable ring or mounting a tape (both useful in batch jobs).

The command line format for ASSIGN is as follows:

```
ASSIGN     |MTpdn [-ALIAS MTldn]|      [-option(s)]
           |MTX   -ALIAS MTldn  |
```

The arguments and options are:

MTpdn	Identifies the magnetic tape (MT) unit number from 0 to 7, inclusive. pdn is the physical device number assigned to each drive at system startup. Numbers can be obtained from the system operator.
MTX	Tells the operator to assign "any available drive"; must be accompanied by -ALIAS MTldn, which assigns a number (alias) to the drive for reference purposes. The actual drive assigned depends on any other options that appear on the command line.
-ALIAS MTldn	Identifies the logical drive number, from 0 to 7, inclusive. ldn is a user-specified number assigned to a particular physical drive unit; used as an alias for the pdn in subsequent magnetic tape operations.

Note

MAGSAV and MAGRST ask the user for the device number of the drive on which a tape is mounted. Both dialogs assume the number given is a logical device number. Consequently, the internal list of logical device numbers is searched first. If a match is found, MAGSAV/MAGRST interacts with the tape mounted on the corresponding physical drive. Suppose you first assign physical device MT0 as logical MT1, then assign physical MT1 as logical MT0. If you answer 1 to the TAPE UNIT: prompt of MAGSAV (or MAGRST), the subsystem assumes that 1 is a logical device number (ldn). Thus, it attempts to read from or write to the tape mounted on physical device MT0, which you previously assigned as logical MT1. (See Chapter 8 for more information on the MAGSAV and MAGRST subsystems.)

-WAIT	Indicates that you are willing to wait until requested drive is available.

-TPID id	Requests the operator to mount a particular reel of tape, identified by a tape id, which can be up to eight characters long. Use of this control argument requires operator intervention. id is a list of tape identifiers (arguments) describing a particular reel of tape, and/or type of tape drive (name, number, etc.). Identifiers may not begin with a hyphen (-), which is a reserved character indicating the next control argument on the ASSIGN command line.
-RINGON -RINGOFF	You may specify protection rights by: RINGON — Read and write permitted or RINGOFF — Read only; write-protection in effect Requires operator intervention for removal or addition of the write-enable ring.
-800BPI -1600BPI -6250BPI	Particular tape density settings are requested with these options. Most drives can handle 800 and 1600 bpi settings. Requires operator intervention.
-7TRK -9TRK	Indicates seven- or nine-track tape drive; default is nine-track. Requires operator intervention if you specify -7TRK.
-MOUNT	Indicates that a tape is to be placed on the tape drive. This control argument requires operator intervention.

Assigning Tape Drives

You may assign magnetic tape drives in any one of three ways:

- By physical device number (pdn):

  ```
  ASSIGN MTpdn [options]
  ```

- By logical device number (ldn):

  ```
  ASSIGN MTX -ALIAS MTldn
  ```

- By logical device number plus characteristics:

  ```
  ASSIGN MTX -ALIAS MTldn -options
  ```

Assigning Drives by Physical Device Number: When you assign a tape drive by a physical device number, you are requesting that particular tape drive. If the drive is unavailable, the option -WAIT queues your request. In the following example, you assign magnetic tape drive MT1. (1 is the physical device number.) This is the default assignment, and it requires no operator intervention:

```
    OK, ASSIGN MT1
    Device MT1 Assigned.
    OK,
```

When you assign a physical device, you may specify an alias if you wish. However, you are still requesting a particular drive. In the following example, you specify physical device MT6 to be logical device MT2:

```
    OK, ASSIGN MT6 -ALIAS MT2
    Device MT6 Assigned.
    OK,
```

Note that the physical, not the logical, device number is returned. ldn's and pdn's are associated internally in a special table and can be used interchangeably. If you do not request an ldn alias, the default logical device number is the same as the physical device number of the drive.

Assigning Drives by Logical Device Number: When you assign a tape drive by a logical device number, you are requesting any tape drive and calling it number ldn. (You specify the number with the -ALIAS option.) Any free tape drive may then be assigned. If all devices are unavailable, the option -WAIT queues your request for the first available device. This method of assigning tape drives maps physical device numbers into logical device numbers. This is especially useful if you have set up a command file with various commands referring to a particular magnetic tape unit. Assigning a tape unit an alias makes the unit known inside the command file. This logical tape assignment process:

- Provides mapping between physical and logical device numbers. You may arbitrarily assign a tape drive by supplying the drive characteristics. If the particular drive is unavailable, you can use another drive with the same characteristics.

- Allows and optionally requires operator intervention in tape unit assignment.

- Allows you to request operator assistance in mounting a tape or performing other tape operations.

When you assign a tape drive by a logical device number plus characteristics, you are asking for any drive that can handle a particular type of tape (for example, a nine-track tape at 6250 bpi). You give this drive a logical alias. Note the following example:

```
    OK, ASSIGN MTX -ALIAS MT0 -TPID ABC -9TRK -RINGON -6250BPI
```

In this example, you wish to assign any drive with the following characteristics: nine-track; read and write (no protect); and 6250 bpi. You wish to mount tape ABC on the drive if a drive is assigned. MT0 is the number you will use to refer to the assigned drive. All subsequent tape operations of the assigned drive will use MT0 as the unit number. If for any reason no drive is assignable, you will see an error message.

Mounting Tapes: When you assign a tape drive, you may also specify a particular tape to be placed on the drive by using the -TPID control argument. Sometimes, it is necessary to have one tape removed from the tape drive and another tape placed or mounted on the tape drive. This is done by using the -MOUNT control argument. To specify -MOUNT, the tape drive must already be assigned. For example, suppose ADLEY, user number 6, assigns logical drive 7:

```
    OK, ASSIGN MT0 -ALIAS MT7 -800 BPI -TPID GRADES
    OK,
```

Now suppose ADLEY reads or writes the tape with id GRADES, and then wants another tape mounted. The command line would be:

```
    OK, ASSIGN -ALIAS MT7 -MOUNT -TPID EXAMS
    OK,
```

The operator receives a message at the system terminal indicating that user ADLEY wants tape EXAMS mounted. The operator responds to ADLEY's request by using the REPLY command. (See Chapter 5.) ADLEY then receives a message indicating whether or not the mount operation was successful. The mount operation might be unsuccessful, for example, if the operator could not find the requested tape.

ASSIGN Command Messages

Messages you get from using the ASSIGN command are either informational or error-related. All of the ASSIGN command messages are listed and described here alphabetically.

- Bad parameter.

This message indicates an ASSIGN syntax error; your input is invalid.

- Device MTldn Assigned.

This is an informational message letting you know that your logical device has been assigned.

- Device not assigned. MTn

This error message indicates that the device you specified has not yet been assigned.

- Mag Tape Assignment Request Aborted (ASSIGN)
  ER!

This is an error message indicating that the operator cannot handle your assignment request for some reason.

- No Mag Tape Assignment Permitted. (AS)
  ER!

This error message indicates that the operator is not permitting any magnetic tape assignments at this time.

- The device is in use. (ASSIGN)
  ER!

This error message indicates that another user has the specified device assigned.

THE UNASSIGN COMMAND

The UNASSIGN command unconditionally unassigns your peripheral device, which in this case is a magnetic tape drive. This command unassigns only the specified tape drive that you previously assigned with the ASSIGN command. When you complete a magnetic tape operation, it is good practice to release the tape drive for general use as soon as possible.

The command line format for UNASSIGN is as follows:

```
    UNASSIGN MTpdn [-UNLOAD]

                or

    UNASSIGN -ALIAS MTldn [-UNLOAD]
```

If you specify the -ALIAS option, MTn is the logical device number (ldn). In this case you are unassigning the tape drive by the ldn. If you do not specify the -ALIAS option, MTn is the physical device number (pdn). You are then unassigning the tape drive by the pdn.

The -UNLOAD option allows the tape to be rewound and then placed off-line (unloaded). This option is not available on all drives or controllers. Check with your System Administrator or operator to see if you may use -UNLOAD at your installation.

A tape drive can only be unassigned by:

- The user who assigned it when SETMOD -USER has been specified; see Chapter 5 for more information on the SETMOD command.
- The system operator.

The system operator can unassign any drive using the pdn argument; the "-ALIAS ldn" option can be used only if the drive is owned by (i.e., was previously assigned by) the operator.

If an operator unassigns your tape drive, no message will appear at your terminal. Should you subsequently attempt to unassign the same device, an error message will be displayed.

For example, suppose that you assign the following tape drives:

```
    OK, ASSIGN MT1 -ALIAS MT2
    Device MT1 assigned.
    OK, ASSIGN MTX -ALIAS MT0
    Device MT0 assigned.
    OK,
```

You can unassign physical drive MT1 as follows:

```
    OK, UNASSIGN MT1
    OK,

          or

    OK, UNASSIGN -ALIAS MT2
    OK,
```

You can unassign physical drive MT0 as follows:

```
    OK, UNASSIGN -ALIAS MT0
    OK,

            or

    OK, UNASSIGN MT0
    OK,
```

The operator can unassign these drives as follows:

```
    OK, UNASSIGN MT1
    OK,

            or

    OK, UNASSIGN MT0
    OK,
```

UNASSIGN Command Messages

Messages you get from using the UNASSIGN command are either informational or error-related. All UNASSIGN command messages are listed and described here alphabetically.

- Bad parameter.

This message indicates an UNASSIGN syntax error; your input is invalid.

- Device not assigned. MTn

This error message indicates that the device you are attempting to unassign has not yet been assigned.

- Device released.

This is solely an informational message telling you that the device you unassigned is released.

THE STATUS DEVICE COMMAND

The STATUS DEVICE command allows you to see which physical devices (tape drives) are currently in use. The output displays physical and logical device numbers for any assigned magnetic tape drives. Tape drives which are not assigned are not printed in the output. The format for the STATUS DEVICE command line is as follows:

```
STATUS DEVICE
```

A typical display may be the following:

```
OK, STATUS DEVICE

DEVICE   USRNAM   USRNUM   LDEVICE
MT0      ADLEY    1        MT0
MT1      PARIS    13       MT2
```

where:

DEVICE	The physical device name
USRNAM	The user login name
USRNUM	The user number
LDEVICE	The logical device name used by the person to whom the drive is assigned.

5 Operator Control of Tape Drives

INTRODUCTION TO THE COMMANDS

The system operator uses the commands ASSIGN, UNASSIGN, and STATUS DEVICE just as any user, though sometimes with certain modifications. (See Chapter 4 for descriptions of these three commands.) In addition, there are two commands to control tape drive operations that are used solely by the system operator. These are SETMOD and REPLY. They allow the operator to:

- Set various modes of tape operation
- Reply to user tape mount requests

THE SETMOD COMMAND

The SETMOD command allows the operator to establish the mode of tape drive assignment. There are three modes of tape drive assignment:

- User (Default) mode: The operator is permitting users to assign their own tape drives (with the ASSIGN command). The operator intervenes only if special assistance is requested.
- Operator mode: Users must channel all tape drive assignment requests through the operator. All assignment requests appear at the operator's console, and the operator must respond to them. (See THE REPLY COMMAND later in this chapter.)

- No assignment mode: Users are not permitted to assign tape drives at all. In this mode, a user assignment request produces an ASSIGN command message saying that tape drives cannot be assigned.

The command line format for SETMOD is as follows:

```
SETMOD     { -USER     }
           { -OPERATOR }
           { -NOASSIGN }
```

The -USER Option (Default Mode)

The -USER option specifies the default assignment mode. It allows users to assign tape drives by physical device number (pdn), with or without the "-ALIAS ldn" option. All other options require operator intervention.

The -OPERATOR Option (Operator Mode)

The -OPERATOR option gives the operator full control of all tape drive assignments. In this mode, all user-issued ASSIGN commands and their corresponding user numbers are displayed at the operator's console. The operator answers each assignment request (see the REPLY command) with a message sent to the user terminal.

The -NOASSIGN Option (No Assignment Mode)

The -NOASSIGN option forbids tape drive assignment. Also, in this mode, no operator is available to assist with tape assignments or mounts. This mode of operation may be used to indicate temporary absence of the operator in an environment where the operator has total control over tape drive assignment.

Console Messages

Messages displayed at the operator's console, then, depend on the current assignment mode established by the SETMOD command. The message is sent to the operator by the ASSIGN subsystem and either requests the operator to assign a tape drive or mount a tape on a drive. This message repeats periodically until the operator responds with the REPLY command. The format of the message that appears is as follows:

```
***** MAGTAPE REQUEST *****
From usrnam (usrnum): message
```

usrnam	The name of the user requesting the tape assignment
usrnum	The user number
message	The same as the command line which the user typed, less the command word ASSIGN

The following example illustrates default mode assignment (either SETMOD -USER has been specified or the SETMOD command has not been used at all). Assume that user number 6, with user name ADLEY, types the following command line:

```
OK, ASSIGN MT2 -TPID SUEA -6250
OK,
```

The operator sees the following on the system terminal:

```
***** MAGTAPE REQUEST *****
From ADLEY (6) : MT2 -TPID SUEA -6250
```

SETMOD Command Message

There is only one message the operator may receive when attempting to use the SETMOD command:

```
"ccc..." not implemented or improper use of argument.  (SETMOD)
```

This is a syntax error message. All or part of the SETMOD command line is in an improper format.

THE REPLY COMMAND

The REPLY command establishes a link between the ASSIGN subsystem and the operator. It is the operator's method of communicating with each user terminal. REPLY allows the operator to:

- Inform a user that a special request has been fulfilled.
- Deny a request.
- Approve a simple request (in -OPERATOR mode).
- Display the actual pdn assigned when MTX option is specified.
- Request repetition of an ASSIGN message.

The command line format for REPLY is as follows:

```
                       ( RESEND )
REPLY -usrnum -TAPE   { ABORT   }
                      { GO      }
                       ( pdn    )
```

usrnum	Identifies the user number.
TAPE	Indicates a reply to tape operation requests.
RESEND	Requests resending of ASSIGN's message.
ABORT	Indicates that no tape drive is assignable, or the operator wants to abort the request for any reason (drive not available, tape not found, etc.)
GO	Identifies a reply to a tape mount request and assignment of a physical drive. It indicates that the requested tape has been mounted on the specified drive, or that the physical drive which the user picked is assigned and ready.
pdn	Identifies the physical drive number. This is a reply to the request of the assignment of a tape drive. It indicates that a drive has been assigned to the user, and the number of the assigned drive is <u>pdn</u>. This reply is used if the user specified MTX in the ASSIGN command.

All outstanding (unanswered) requests may be repeated by using the -ALL option of REPLY. The operator can display just the most recent, or all, of the outstanding requests made by a specific user. The command line format is as follows:

```
REPLY { -userno }  -RESEND
      { -ALL    }
```

<u>-userno</u> is the number of the user whose requests the supervisor wants to see again. -ALL repeats all the outstanding requests issued by a specific user. -RESEND repeats only the last assignment request issued by a specific user.

The operator may also set the message repetition frequency. The repetition period is the time between the printing of two successive messages from the same request on the operator's console. The command line format is as follows:

```
REPLY -REPEAT seconds
```

<u>seconds</u> is a decimal number specifying the message repeat period in seconds. The default message frequency is 3 minutes (180 seconds).

This example illustrates the operator reply to a simple assignment request. First, the user assigns a tape drive:

```
OK, ASSIGN MTX -ALIAS MT0 -TPID DATA -800
OK,
```

The operator then receives the following message:

```
***** MAGTAPE REQUEST *****
From ADLEY (3) : MTX -ALIAS MT0 -TPID DATA -800
```

The operator mounts the tape DATA on an available drive and sets the density to 800 bpi. If, for instance, the operator chooses MT5 as the available drive, the reply would be:

```
OK, REPLY -3 -TAPE 5
OK,
```

REPLY Command Messages

The REPLY command generates both informational and error messages. All of these messages are listed and described here alphabetically.

- "ccc..." not implemented or improper use of argument. (REPLY)

This is a syntax error message. A string of characters in the command line is in an improper format.

- No outstanding request from all users.

This is an informational message telling the operator that there are no outstanding requests from any users.

- No pending request from user n.

This is an informational message telling the operator that there is no pending request from the specified user.

- User number not specified. (REPLY)

This is a REPLY error message. The operator did not give a user number in the command line.

6 Initializing Magnetic Tapes

INTRODUCTION

The PRIMOS LABEL command is used to initialize magnetic tapes. LABEL writes either IBM (nine-track EBCDIC or seven-track BCD) or ANSI (nine-track ASCII) Level 1 volume labels followed by dummy HDR1 and EOF1 labels. LABEL can also be used to read existing VOL1 and HDR1 labels. ANSI labels are written in accordance with the ANSI standard X3.27-1978. IBM labels are written in accordance with IBM's specifications (IBM manual GC28-6680-5). Any nonstandard labels such as seven-track ASCII or user-defined labels cannot be read or written.

USING LABEL

The command line format for LABEL is as follows:

```
                                 ( -VOLSER )
LABEL MTn [-TYPE type]   [       { -VOLID  }   volume-id]   [-OWNER owner]
                                 ( -VOLUME )
      [-ACCESS access]                                      [-INIT]
```

The meanings of the parameters are:

MTn	The tape drive where the tape to be labelled is located. n is a number between 0 and 7. This keyword is required and must be the first on the command line.
type	-TYPE A nine-track ASCII (ANSI) (this is the default) -TYPE B seven-track BCD (IBM) -TYPE E nine-track EBCDIC (IBM)
volume-id	A one- to six-character string that uniquely identifies this tape reel. If you specify fewer than six characters, they are blank-padded on the right. The keywords -VOLUME, -VOL, or -VOLSER may be substituted for the keyword -VOLID.
owner	One to 14 characters long for ANSI labels, one to 10 characters long for IBM labels. If you specify fewer than 14 (or 10) characters, they are blank-padded on the right. If this keyword is omitted, the default is the user's login name. The keyword -OWN may be substituted for the keyword -OWNER.
access	A single character defining access to this tape. ACCESS is not used by Prime software but is included for completeness. If it is omitted, it is left blank on ANSI labels. ACCESS is ignored for IBM labels.
-INIT	Necessary keyword for tapes previously unformatted.

To read existing labels, type the command:

```
LABEL MTn [-TYPE type]
```

To write labels, type the command:

```
LABEL MTn [-TYPE type] -VOLID volume-id [-OWNER owner]
[-ACCESS access] [-INIT]
```

The following example illustrates how you would use LABEL to write an ANSI VOL1 label on a new tape located on logical device number 3. The owner of the tape is SPARIS, the volume serial ID is SP573, and the access is set to X:

```
OK, LABEL MT3 -TYPE A -VOLSER SP573  -OWNER SPARIS -ACCESS X
```

The next example illustrates the command line to read seven-track IBM labels that already exist on the tape (located on logical device number 6):

```
    OK, LABEL MT6 -TYPE B
```

LABEL Command Errors

Improper use of the LABEL command causes an error message to be printed. These errors are the result of bad syntax in the LABEL command itself or a PRIMOS magnetic tape I/O error.

Syntax Errors: All LABEL command syntax errors are listed and described here alphabetically.

- ***ACCESS IGNORED FOR IBM LABELS (WARNING ONLY)

This is a warning only — processing continues. (Error 13)

- ***DUPLICATE KEYWORD DETECTED

The same keyword was typed more than once. (Error 1)

- ***INVALID LABEL TYPE SPECIFIED

Label type must be one of the characters A, E, or B. (Error 5)

- ***INVALID TAPE UNIT SPECIFIED

Something other than MT0-MT7 was typed. (Error 2)

- ***LABEL OPERATION ABORTED

Error 9, 10, 12, or 14 occurred and the label was not read. (Error 11)

- ***LABEL READ WAS NOT TYPE x

The label read was not of the type specified. (Error 12)

- ***NO MAGNETIC TAPE UNIT SPECIFIED

A magnetic tape unit is required. (Error 6)

- ***OWNER ID SPECIFIED IS TOO LONG

The owner ID cannot be longer than 14 characters. (Error 4)

- ***OWNER ID SPECIFIED IS TOO LONG FOR TYPES B OR E

The owner ID for IBM labels cannot be longer than 10 characters. (Error 8)

- ***UNABLE TO READ TAPE LABEL ON THIS TAPE

A mag tape read error occurred and the label was not read. (Error 10)

- ***UNABLE TO WRITE TAPE LABEL ON THIS TAPE

A mag tape read error occurred and the label was not written. (Error 9)

- ***UNRECOGNIZED KEYWORD, string (CMDL$A)

An invalid keyword (string) appeared on the command line. (Error 15)

- ***VOL1 LABEL ALREADY EXISTS

ANSI standards prohibit the rewriting of VOL1 labels. (Error 14)

* ***VOLUME ID SPECIFIED IS TOO LONG

The volume ID cannot be longer than six characters. (Error 3)

* ***VOLUME ID WAS NOT SPECIFIED

When writing labels, a volume ID is required. (Error 7)

PRIMOS Errors: The PRIMOS magnetic tape I/O errors are listed and described here alphabetically.

* subr BADC

LABEL improperly called magnetic tape subroutines.

* subr EOF

Indicates end of file on the magnetic tape.

* subr EOT

Indicates end of tape.

* subr HERR

Tape drive hardware error.

* subr MTNO

The tape drive is not operational.

* subr PERR

Parity error on the tape drive.

- MTn NOT ASSIGNED

Use command AS MTn before the LABEL command.

(In the above errors, subr is the name of the mag tape subroutine that reported the error. See The PRIMOS Subroutines Reference Guide for more information regarding these errors.)

If LABEL successfully writes a label, the message TAPE LABEL WAS WRITTEN SUCCESSFULLY is displayed. On read operations, LABEL prints out the volume and owner IDs, creation date, access (ANSI tapes only), and other information.

PART III

PRIMOS Magnetic Tape Subsystems

7 The MAGNET Subsystem

INTRODUCTION TO MAGNET

To transfer data by magnetic tape from a non-Prime operating system to PRIMOS and vice versa, you must write the magnetic tape in an interchange format that both operating systems can understand. MAGNET is the PRIMOS subsystem that allows you to read and write magnetic tapes in these various interchange formats. MAGNET is intended primarily for users who have had prior experience with magnetic tapes.

Some of the facilities that the MAGNET subsystem provides are:

- Declaration and modification of I/O objects
- Linkage of I/O objects with PRIMOS global variables
- Translation to and from various character sets
- User-definable character sets
- Seven-track binary packing and unpacking
- Tape positioning
- Physical tape copying
- Logical data transfer between devices with support of multiple destinations

- Support of unlabelled tapes and of ANSI and IBM standard labelled tapes
- Support of fixed-length records and of IBM, ANSI, and Prime variable-length records
- Support of old (pre-rev 18.3) MAGNET features for the READ, WRITE, COPY, and POSITION subcommands
- Support for tape operations within Batch mode (operator intervention)

INVOKING MAGNET

To invoke MAGNET, you must be at PRIMOS command level. Type MAGNET after the PRIMOS prompt (OK,). At this point there are several PRIMOS command line options you can specify. -SILENT causes only MAGNET severity 2 or 3 errors to be printed at your terminal. You may also specify -USER or -OPERATOR (may be abbreviated to -OPR). These two options direct the printing of MOUNT and DISMOUNT messages on either the operator's terminal or your terminal, respectively. For detailed information on these options, see Appendix B.

After you type MAGNET and possibly a mode option, the subsystem prints a release number and gives you a prompt. For example:

```
OK, MAGNET
[MAGNET, Rev. 18.3]
>
```

After the MAGNET prompt (>), you type a subcommand line that contains one of the 16 MAGNET subcommands. Depending on the subcommand, the line may also contain one or more object-names and/or one or more options. The MAGNET subcommand line format is as follows:

```
> subcommand   object-name(s)   option=(value),...option=(value)
```

There are a few points to remember when typing subcommand lines:

- Input can be up to 1000 characters long.
- You can use either upper or lowercase alpha characters or a combination of both.
- Free format is allowed on the subcommand line. (Spacing is optional.)

MAGNET SUBCOMMANDS

There are 16 subcommands in MAGNET. Fourteen of these are the basic MAGNET subcommands, which define, manipulate, and transfer data within the subsystem. These subcommands are described in the beginning of this chapter. The two other subcommands are used less frequently and are described later in this chapter under ADVANCED MAGNET. Table 7-1 lists each MAGNET subcommand, its type, and its function.

DATA DEFINITION AND MANIPULATION SUBCOMMANDS

There are seven MAGNET subcommands that define and manipulate data within the subsystem. They are DECLARE, LIST, DISPLAY, RENAME, MODIFY, DELETE, and TRANSLATE.

The DECLARE Subcommand

You use DECLARE to define and characterize a given object. The subcommand line format is as follows:

```
                       ( TAPE=(devno)      )
> DECLARE  object-name { DISK=(pathname)   }  [,OPTION=(value)...]
                       ( EXTERNAL=(gvar)   )
```

The object-name identifies either a tape file or a disk file. You always specify an object-name with the DECLARE subcommand. It is good practice to create object-names that are mnemonic. You can use any combination of up to 32 alpha characters, numerals, and the two characters period (.) and underscore (_). Any of these characters may appear first in the object-name. You may declare up to 100 object-names.

An OPTION provides information that describes an object-name. The first option you specify on the DECLARE subcommand line must always be TAPE, DISK, or EXTERNAL. Other options, if any, follow. Table 7-2 lists each MAGNET subcommand option, its type, applicable subcommand(s), and its function.

There are 34 possible options you can specify with the DECLARE subcommand. Nineteen of these are the basic options that you use for most tape operations in MAGNET. These are described here. The other 15 are options you may use less frequently. These are described later in this chapter under ADVANCED MAGNET.

Table 7-1
MAGNET Subcommands

Subcommand	Type	Function
/*	----	Puts comments in your MAGNET dialog.
COPY	Data Transferral	Copies one or more physical files from one tape to another.
DECLARE	Data Definition/ Manipulation	Associates one or more options with an object-name.
DELETE	Data Definition/ Manipulation	Deletes one or more declared object-names.
DISPLAY	Data Definition/ Manipulation	Shows all options associated with a declared object-name.
LIST	Data Definition/ Manipulation	Lists all declared object-names.
LOAD	Data Definition/ Manipulation	Loads a translation table.
MODIFY	Data Definition/ Manipulation	Changes an option value or adds an option to an object-name.

Table 7-1
MAGNET Subcommands (continued)

Subcommand	Type	Function
MOVE	Data Transferral	Moves a logical file from a source object to one or more destination objects.
POSITION	Data Transferral	Positions a tape to a specific file number and record number.
QUIT	——	Exits from MAGNET and returns you to PRIMOS command level.
READ	Data Transferral	Reads a file from a tape and transfers it to a disk file.
RENAME	Data Definition/ Manipulation	Renames declared object-names.
SAVE	Data Definition/ Manipulation	Saves options associated with an object-name in a PRIMOS global variable.
TRANSLATE	Data Definition/ Manipulation	Specifies the translation associated with an object-name.
WRITE	Data Transferral	Writes a disk file onto tape.

The TAPE Option: You specify the TAPE option if the object-name preceding it on the subcommand line refers to a tape file. The value you assign to this option is the logical device number of the tape drive on which the tape is mounted (a number from 0 to 7). You must assign your tape drives at PRIMOS command level before you invoke MAGNET. For more information, see the ASSIGN command in Part II, PRIMOS MAGNETIC TAPE COMMANDS. In the following example, the DECLARE subcommand defines the object-name TFILE to be a tape file. It is located on logical device (tape drive) number 7.

```
    > DECLARE  TFILE  TAPE=(7)
    >
```

The DISK Option: You specify the DISK option if the object-name preceding it on the subcommand line refers to a disk file. The value you assign to this option must be a PRIMOS filename or pathname. In the following example, the object-name DFILE is a PRIMOS disk file, SOCSECNUMS, in the UFD SFA:

```
    > DECLARE  DFILE  DISK=(SFA>SOCSECNUMS)
    >
```

The LRECL and BFACTOR Options: LRECL is the logical record length option. It specifies the number of bytes in each logical record in fixed-length format, or the maximum number of bytes in each logical record in variable-length format.

BFACTOR is the blocking factor option. It specifies the number of logical records per physical tape block. For more information on logical record sizes and blocking factors, see the discussion on RECORDS, GAPS, AND BLOCKS in Part I, INTRODUCTION TO MAGNETIC TAPES. The relationships between logical records, blocking factors, and physical tape blocks are illustrated in Figures 7-1, (A) through (D).

LRECL and BFACTOR together specify the maximum size of a tape file's physical records. You obtain this maximum size by multiplying the value of LRECL by the value of BFACTOR. The maximum size can be no larger than 12,288 bytes (this number is smaller if you are using ANSI or IBM variable-length records). BFACTOR is strictly a tape option; you use it only with tape objects. LRECL is for both tape and disk objects. You must always specify the value of LRECL for tape files. If you do not specify BFACTOR, it defaults to the value of 1.

Table 7-2
MAGNET Subcommand Options

Option	Type	Applicable Subcommand(s)	Function
ACCESS	Tape	DECLARE, MODIFY	Identifies the ACCESS field on HDR1, EOF1, and EOV1 (level 1) labels.
AMOUNT	——	COPY	Specifies the number of files to copy.
BFACTOR	Tape	DECLARE, MODIFY	Specifies the number of logical records per physical tape block.
BUFFERS	Tape	DECLARE, MODIFY	Specifies the number of buffers being used.
BYPASS	Tape	DECLARE, MODIFY	Indicates to MAGNET whether or not to ignore tape labels and identify its files by number.
CHARACTERS	Tape	DECLARE, MODIFY	Specifies whether one or two characters per word are to be read or written.
CREATE	Tape	DECLARE, MODIFY	Identifies the creation date field for Level 1 file labels.
DENSITY	Tape	DECLARE, MODIFY	Specifies density of the tape being read or written.
DISK	Disk	DECLARE, MODIFY, LOAD	Specifies the location of a disk file.
EXCHANGE	Tape	DECLARE, MODIFY	Indicates whether, for each logical record, high and low order bytes within words are to be exchanged.

Table 7-2
MAGNET Subcommand Options (continued)

Option	Type	Applicable Subcommand(s)	Function
EXPIRE	Tape	DECLARE, MODIFY	Identifies the expiration date field for Level 1 file labels.
EXTERNAL	----	DECLARE, MODIFY, SAVE	Indicates that additional options can be found in an external PRIMOS global variable.
FILEID	Tape	DECLARE, MODIFY	Identifies the labelled file to be read or written.
FILENO	Tape	DECLARE, MODIFY	Specifies a file number.
FORMAT	Disk Tape	DECLARE, MODIFY	Identifies the type of records being read or written.
GENERATION	Tape	DECLARE, MODIFY	Identifies the generation field of Level 1 file labels.
LABELS	Tape	DECLARE, MODIFY	Identifies the type of labels on a tape.
LEVEL	Tape	DECLARE, MODIFY	Specifies the amount of labelling on a tape.
LINES	----	LOAD	Specifies the number of lines in a user translation table.
LRECL	Disk Tape	DECLARE, MODIFY	Specifies the number of bytes in each logical record.
MAXIO	Tape	DECLARE, MODIFY	Limits the maximum I/O transfer size.
MODE	----	POSITION	Instructs MAGNET to position either absolutely or relatively.
NEXTCHAIN	Disk Tape	DECLARE, MODIFY	Declares the next object in a chain of tapes.

Table 7-2
MAGNET Subcommand Options (continued)

Option	Type	Applicable Subcommand(s)	Function
OFFSET	Tape	DECLARE, MODIFY	Specifies the size of a field, at the beginning of physical records, that contains control characters to be ignored.
OWNER	Tape	DECLARE, MODIFY	Identifies the owner id field on a VOL1 label.
PARITY	Tape	DECLARE, MODIFY	Identifies the parity of a seven-track tape.
POSTACTION	Disk Tape	DECLARE, MODIFY	Specifies action to perform when closing a tape or disk file.
PREACTION	Disk Tape	DECLARE, MODIFY	Specifies action to perform when opening a tape or disk file.
PREVCHAIN	Disk Tape	DECLARE, MODIFY	Declares the previous object in a chain of tapes.
PRINT	——	COPY	Prints the size of the first physical record in each tape file.
PROTECT	Tape	DECLARE, MODIFY	Prevents writing on a tape.
RECORDNO	Tape	DECLARE, MODIFY	Specifies the number of physical records to be spaced forward or backward.
SEQUENCE	Tape	DECLARE, MODIFY	Identifies the file sequence field on Level 1 file labels.
TAPE	Tape	DECLARE, MODIFY	Defines a tape object.
TRACKS	Tape	DECLARE, MODIFY	Specifies the number of tracks on a tape.
TYPE	——	LOAD	Identifies the type of a translation table.

Table 7-2
MAGNET Subcommand Options (continued)

Option	Type	Applicable Subcommand(s)	Function
VERSION	Tape	DECLARE, MODIFY	Identifies the generation version field on Level 1 file labels.
VISUAL	Tape	DECLARE, MODIFY	Identifies the external visual id of a tape reel.
VOLSER	Tape	DECLARE, MODIFY	Identifies the volume serial id field on a VOL1 label.

For disk objects, LRECL specifies the size in characters of each record in a binary disk file or the maximum size in characters of each record in an ASCII disk file. The value must be an even number between 2 and 12,288. In the following example, you declare X to be a disk file, SUEFILE. Each logical record in the disk file is 80 characters long:

```
> DECLARE X DISK=(*>SUEFILE), LRECL=(80)
>
```

Note

With a binary disk file, you must also specify FORMAT=(FIXED). For ASCII files, you should specify FORMAT=(VAR/PRIME). For more information, see the discussion of the FORMAT option later in this chapter.

In the next example, the tape file EMPNAMES has a blocking factor of 10 logical records per physical record. Each logical record contains a maximum of 80 bytes. Therefore, the maximum physical record size is 800 bytes:

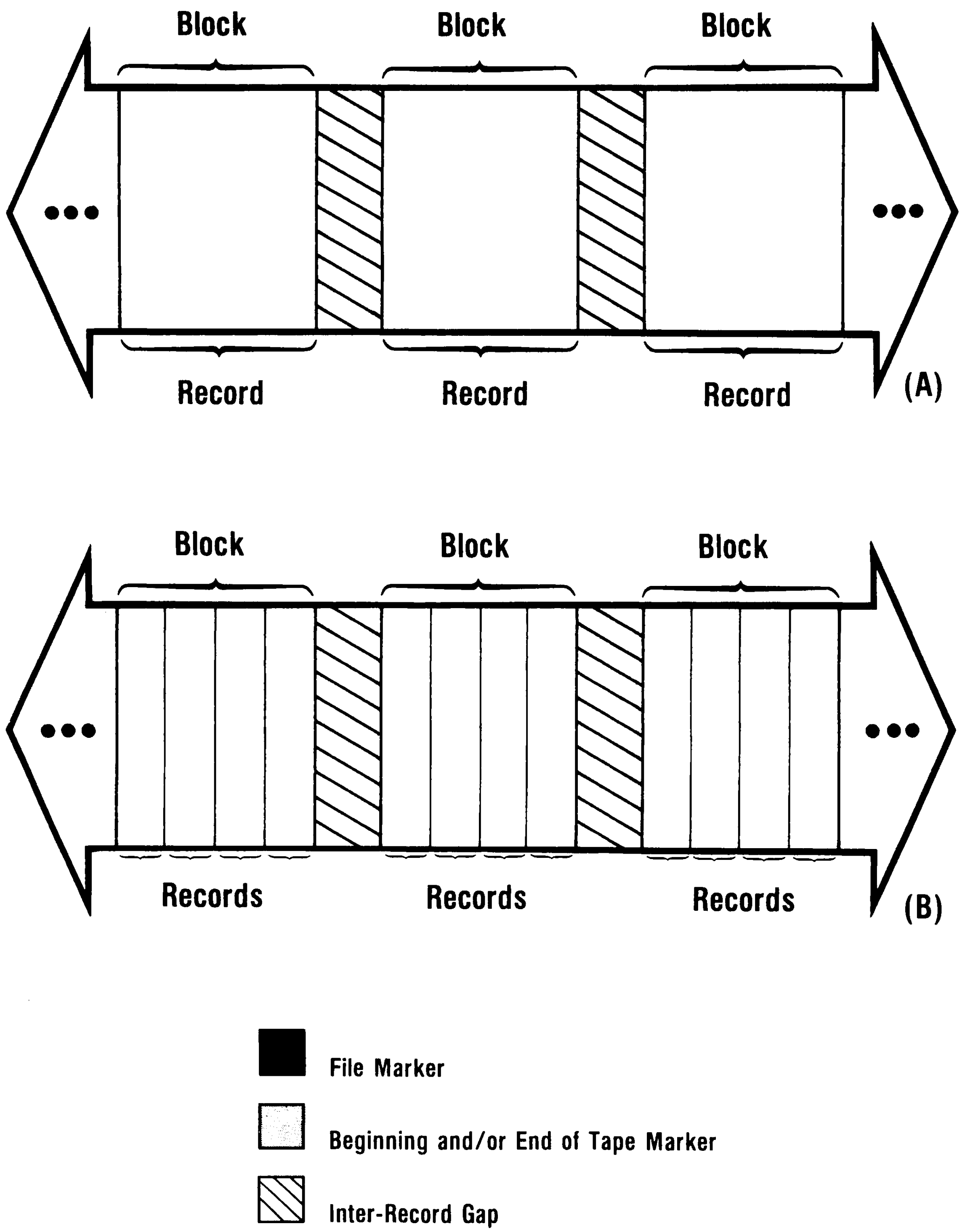
```
> DECLARE  EMPNAMES  TAPE=(3), LRECL=(80), BFACTOR=(10)
>
```

The TRACKS Option: You specify the TRACKS option to identify the number of tracks on your tape. The two possible values are either 7 or 9, for seven- or nine-track tapes respectively. TRACKS is strictly a tape option, and it defaults to the value of 9. In the following example, a tape object, EMPNAMES, is a nine-track tape located on logical device number 6.

```
> DECLARE EMPNAMES TAPE=(6), TRACKS=(9)
>
```

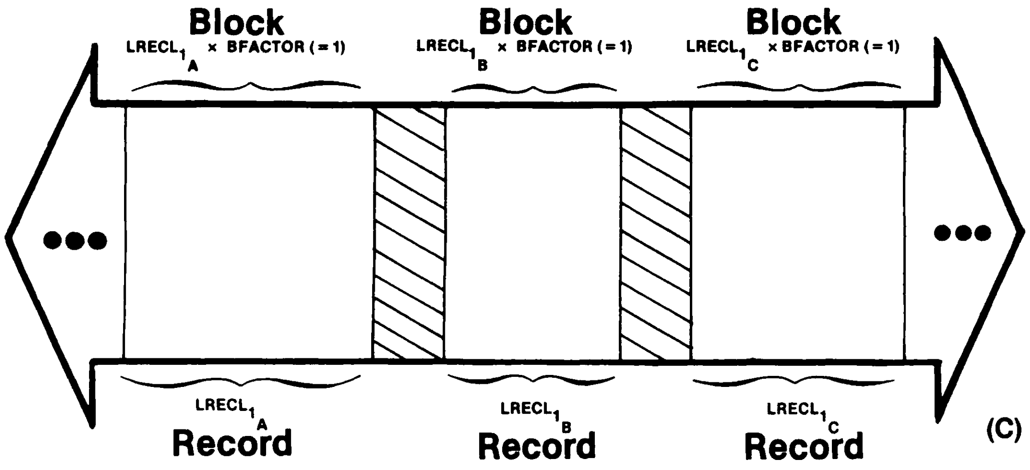
The DENSITY Option: You use DENSITY to specify the density of the tape that you are reading or writing. For seven-track tapes, you can specify either 200, 556, or 800 (bpi) as the value. For nine-track tapes, you can specify either 800, 1600, or 6250 (bpi) as the value. DENSITY is solely a tape option, and it defaults to the value of 800. In the following example, DATAFILE is a tape object located on logical device number 2. It is a seven-track tape with a density of 556 bpi.

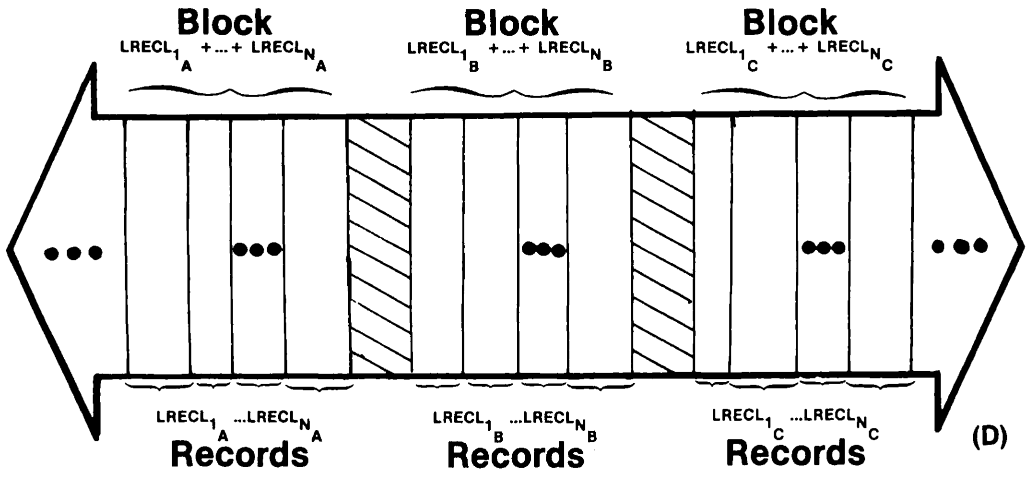
```
> DECLARE DATAFILE TAPE=(2), TRACKS=(7), DENSITY=(556)
>
```



The LRECL and BFACTOR Options
(A) Fixed-length Records, Blocking Factor = 1
(B) Fixed-length Records, Blocking Factor = N

Figure 7-1


Block
$LRECL_{1_A} \times BFACTOR (=1)$
Block
$LRECL_{1_B} \times BFACTOR (=1)$
Block
$LRECL_{1_C} \times BFACTOR (=1)$
$LRECL_{1_A}$
Record
$LRECL_{1_B}$
Record
$LRECL_{1_C}$
Record
(C)


Block
$LRECL_{1_A} + \ldots + LRECL_{N_A}$
Block
$LRECL_{1_B} + \ldots + LRECL_{N_B}$
Block
$LRECL_{1_C} + \ldots + LRECL_{N_C}$
$LRECL_{1_A} \ldots LRECL_{N_A}$
Records
$LRECL_{1_B} \ldots LRECL_{N_B}$
Records
$LRECL_{1_C} \ldots LRECL_{N_C}$
Records
(D)

■	File Marker
░	Beginning and/or End of Tape Marker
▨	Inter-Record Gap

The LRECL and BFACTOR Options
(C) Variable-length Records, Blocking Factor = 1
(D) Variable-length Records, Blocking Factor = N

Figure 7-1

The FORMAT Option: You use the FORMAT option to identify the type of records you are reading from or writing to your tape or disk. There are four possible values you can specify for tape objects:

- FIXED (for fixed-length records)
- VAR/ANSI (for ANSI standard variable-length records, nine-track only)
- VAR/IBM (for IBM standard variable-length records, nine-track only)
- VAR/PRIME (for Prime variable-length records)

There are two possible values you can specify for disk objects:

- FIXED (for fixed-length records, binary disk files)
- VAR/PRIME (for Prime variable-length records, ASCII disk files)

With Prime variable-length records, the logical record length equals the physical record length; you can specify any size up to a maximum of 12,288 bytes. For tape objects it is important to note that you must always specify a value of 1 for the BFACTOR option for this format only. Prime variable-length tape records have no record control word (RCW) or block control word (BCW). (For more information on the Prime variable-length tape format, see Part I, Chapter 2.) These formats are illustrated in Figures 7-2, (A) through (C).

For tape objects, FORMAT defaults to FIXED for a value. For disk objects, FORMAT defaults to VAR/PRIME for a value. In the following example, DEDUCTIONS is a tape object located on logical device number 5. It is a nine-track tape with ANSI variable-length records:

```
> DECLARE DEDUCTIONS TAPE=(5), TRACKS=(9), FORMAT=(VAR/ANSI)
>
```

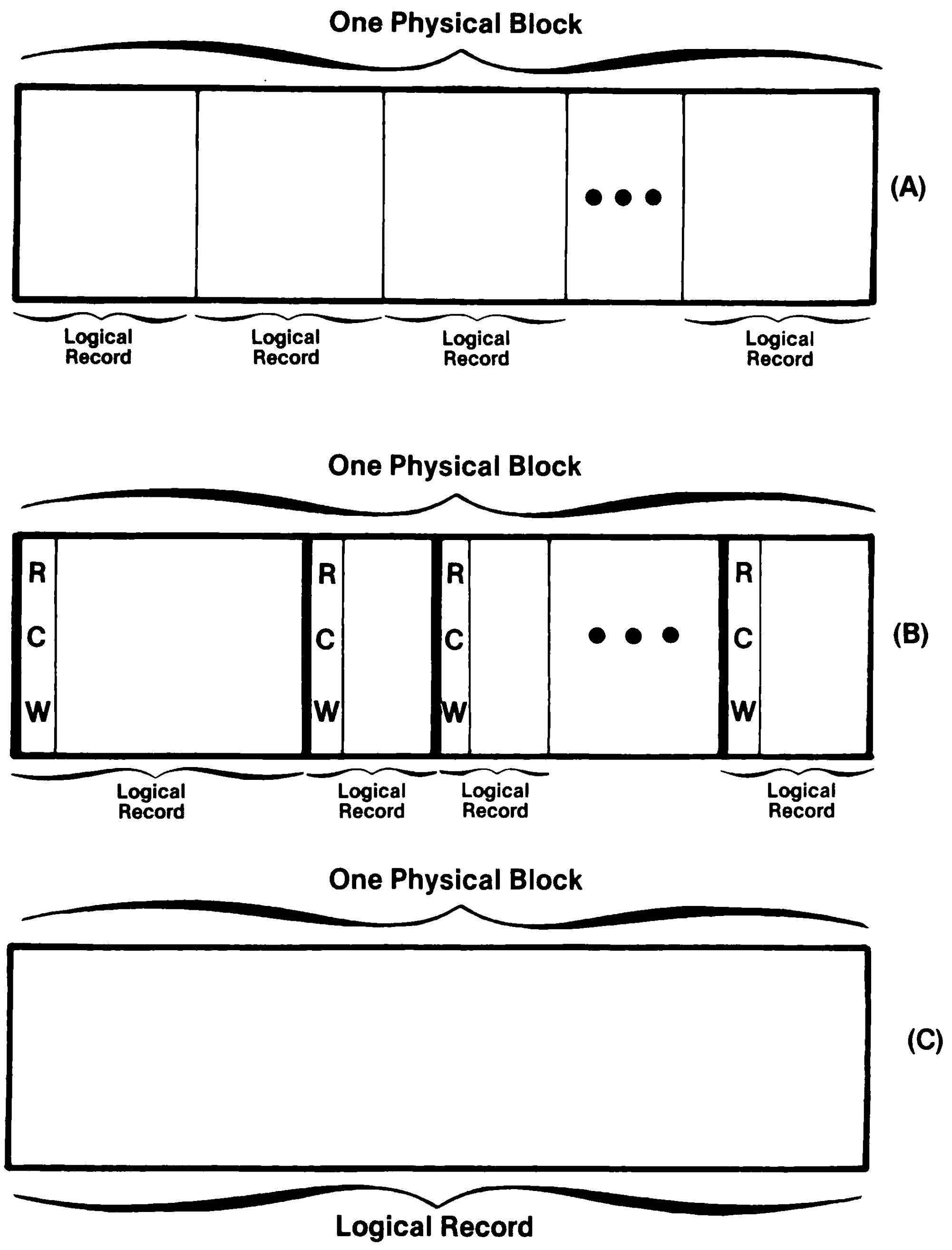
In the following example, X is a disk object that refers to the disk file, SUEFILE. It is a binary file with records 80 characters long:

```
> DECLARE X DISK=(SUEFILE), LRECL=(80), FORMAT=(FIXED)
>
```

Note

With disk files you must also specify a logical record length. For more information, see the discussion of the LRECL option in this chapter.

The PARITY Option: PARITY specifies either odd or even parity for a seven-track tape. It is a seven-track tape option only; nine-track tapes are always written in odd parity. There is no default for this



The FORMAT Option
(A) Fixed-Length Format
(B) ANSI and IBM Variable-Length Format
(C) Prime Variable-Length Format

Figure 7-2

option. You must always specify it when you declare a seven-track tape object. In the following example, STATS is a tape object located on logical device number 2. It is a seven-track tape with odd parity:

```
> DECLARE STATS TAPE=(2), TRACKS=(7), PARITY=(ODD)
>
```

The PROTECT Option: This option is the software equivalent of a write-enable ring on a reel of magnetic tape. PROTECT prevents you from writing on your tape. You specify either YES or NO as the value. If you do not specify PROTECT, it defaults to NO. This option is for tape objects only. In the following example, VITAL_INFO is a tape object located on logical device 0. It is protected from writing:

```
> DECLARE VITAL_INFO TAPE=(0), PROTECT=(YES)
>
```

The LABELS Option: You use the LABELS option to identify the type of tape labels that you are using. There are three possible values you can specify:

- NONE (You are not using any type of labels.)
- ANSI (You are using ANSI standard labels.)
- IBM (You are using IBM standard labels.)

LABELS is strictly a tape option. If you do not specify it, it defaults to NONE. In the following example, INCOME is a tape object located on logical device number 6. It is a nine-track tape with ANSI standard labels written at 800 bpi:

```
> DECLARE INCOME TAPE=(6), LABELS=(ANSI), TRACKS=(9), DENSITY=(800)
>
```

The LEVEL Option: You use the LEVEL option to identify the number of labels present on your tape. LEVEL limits the number of labels written on output; it has no effect whatsoever on input. There are four possible values that you can specify:

- 0 (for an unlabelled tape)
- 1 (for Level 1 labels only: HDR1, EOF1, or EOV1)
- 2 (for Level 1 labels and HDR2, EOF2, and/or EOV2 labels only)
- 3 (for Level 1 labels, HDR2, EOF2, EOV2, and user labels only)

These four levels are illustrated in Figures 7-3, (A) through (D). For more information on these levels of tape labels, see Part I, Chapter 3 of this manual.

Note

> Prime does not yet support user label processing. You may specify a value of 3 but MAGNET treats it as if you had specified a value of 2.

LEVEL is for tape objects only. If you do not specify it, the default is 0. In the following example, MYTAPE is a tape object located on logical device number 3. It has two levels of IBM standard labels:

```
> DECLARE MYTAPE TAPE=(3), LABELS=(IBM), LEVEL=(2)
>
```

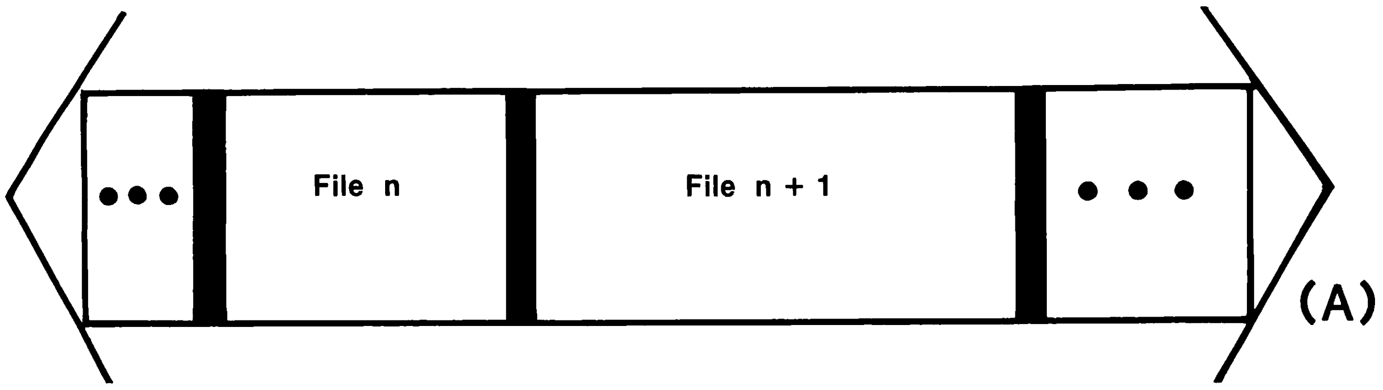
The FILEID Option: You use the FILEID option to identify the name of the labelled file you wish to read or write. FILEID specifies the file-id field of Level 1 labels. The value you specify for this option is a string of no more than 17 characters. If you specify fewer than 17 characters, the value is blank-padded on the right. The string can be any mixture of the following:

- Upper and/or lowercase alpha characters
- Numbers
- Any other characters except right and left parentheses, equal signs, commas, apostrophes, and blanks

FILEID is solely a tape option. If you do not specify it, the default is 17 blanks. In the following example, SYSIN is an IBM standard labelled file located on logical device number 2. The file-id field of the Level 1 labels is SOCSECNUMS:

```
> DECLARE SYSIN TAPE=(2), LABELS=(IBM), FILEID=(SOCSECNUMS)
>
```

The FILENO Option: FILENO identifies the number of the file you wish to work with on your tape. It is strictly a tape option for unlabelled files or for labelled files whose labels are being bypassed. (For more information, see the description of the BYPASS option.) Alternatively, you use the FILENO option with POSITION, a MAGNET subcommand, to specify absolute or relative file positioning on a tape. (For more information, see the POSITION subcommand description later in this chapter.)



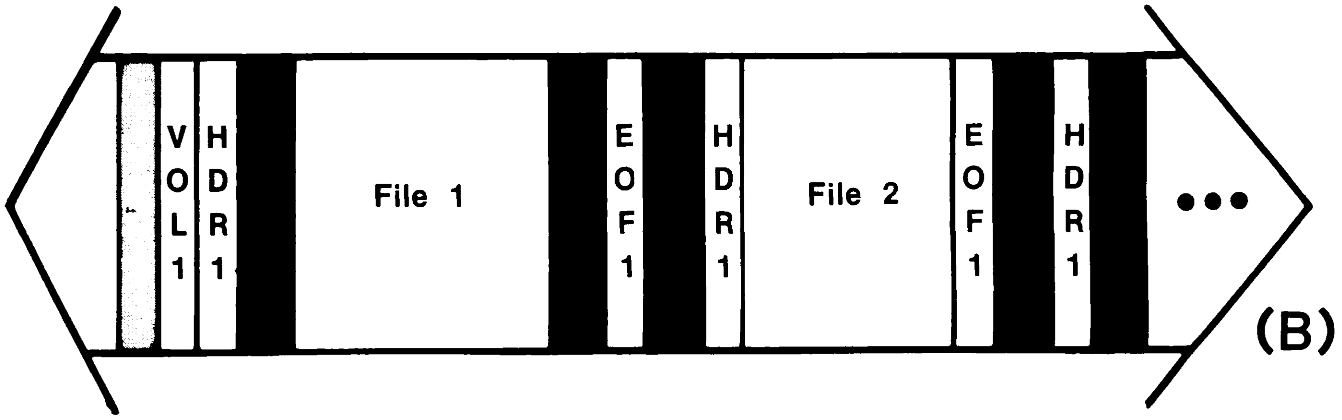
■ File Marker

▒ Beginning and/or End of Tape Marker

▨ Inter-Record Gap

The LEVEL Option
(A) Tape Format for LEVEL = (0)

Figure 7-3



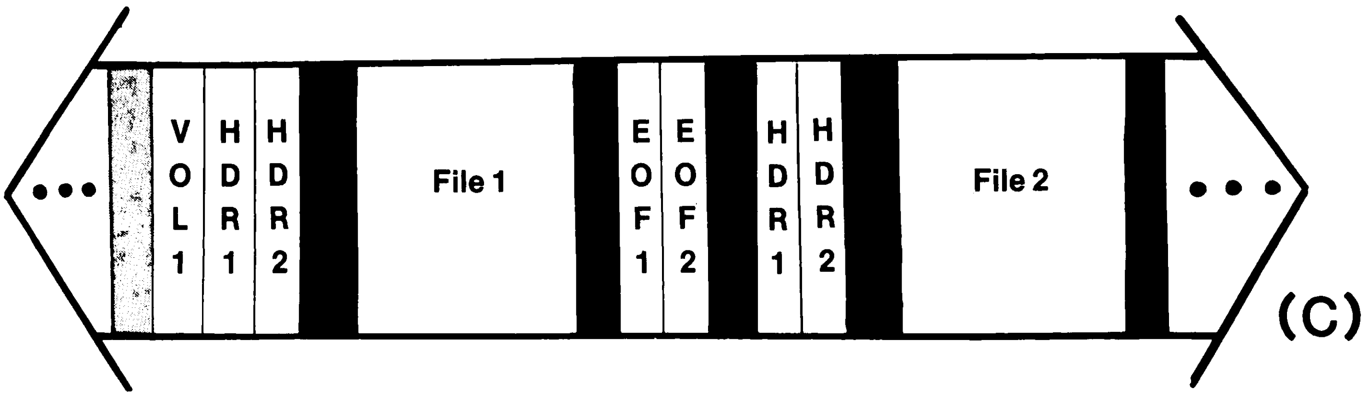
■ File Marker

□ Beginning and/or End of Tape Marker

▨ Inter-Record Gap

The LEVEL Option
(B) Tape Format for LEVEL = (1)

Figure 7-3



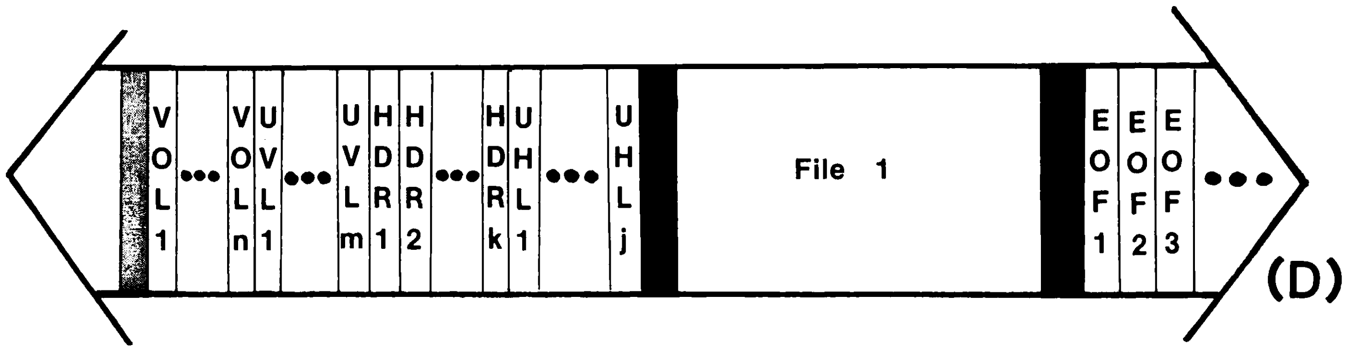
■ File Marker

▒ Beginning and/or End of Tape Marker

▨ Inter-Record Gap

The LEVEL Option
(C) Tape Format for LEVEL = (2)

Figure 7-3



■ File Marker

░ Beginning and/or End of Tape Marker

▨ Inter-Record Gap

The LEVEL Option
(D) Tape Format for LEVEL = (3)

Figure 7-3

If you use this option to specify an unlabelled tape file or a labelled file whose labels are bypassed, the value must be a number greater than or equal to 0. (A value of 0 indicates that you wish to work with the file at your current position on the tape.) If you use FILENO for positioning, the value may be less than 0.

In the following example, OUTFILE is an unlabelled tape object located on logical device number 1. You are working with the second file on the tape:

```
> DECLARE OUTFILE TAPE=(1), LABELS=(NONE), FILENO=(2)
>
```

The OWNER Option: You use the OWNER option to identify the owner identificaton field on a VOL1 label. The value you specify depends on whether your tape is ANSI or IBM labelled. For both types of labelled tapes, the value is a string of characters. You can specify up to 14 characters for ANSI labelled tapes and up to 10 characters for IBM labelled tapes. If you specify a value that contains less than the maximum number of characters for either format, the value is blank-padded on the right. As for the FILEID option, the string can be any mixture of the following:

- Upper and/or lowercase alpha characters
- Numbers
- Any other characters except right and left parentheses, equal signs, commas, apostrophes, and blanks

OWNER is strictly a tape option. If you do not specify this option for an ANSI labelled tape object, it defaults to 14 blank characters. The default for an IBM labelled tape object is 10 blank characters. In the following example, WAGES is an ANSI labelled tape object located on logical device number 4. The owner identification field is SUSANADLEY:

```
> DECLARE WAGES TAPE=(4), LABELS=(ANSI), OWNER=(SUSANADLEY)
>
```

The VOLSER Option: VOLSER identifies the volume serial identificaton field (VOLSER number) on a VOL1 label. The value is a string of no more than six characters that can be a mixture of any of the following:

- Upper and/or lowercase alpha characters
- Numbers
- Any other characters except right and left parentheses, equal signs, commas, apostrophes, and blanks

If you specify less than six characters, the value is blank-padded on the right. The VOLSER option is solely for tape objects. If you do not specify it, the default is six blank characters. In the following example, DATA_TEST is an ANSI labelled tape object located on logical device number 3. The VOLSER id is 63A:

```
    > DECLARE DATA_TEST TAPE=(3), LABELS=(ANSI), VOLSER=(63A)
    >
```

The RECORDNO Option: The RECORDNO option identifies the number of physical records that you wish to space either forward or backward. You use this option only in conjunction with the POSITION and COPY subcommands. (For more information, see the descriptions of the POSITION and COPY subcommands later in this chapter.) The value you specify with RECORDNO is either a positive or negative number (for spacing either forward or backward, respectively). This option is for tape objects only. If you do not specify it, the default is 0. In the following example, BAR is a tape object located on logical device 0. RECORDNO indicates to the POSITION subcommand that you wish to position the tape five filemarkers backward and then 17 physical records forward:

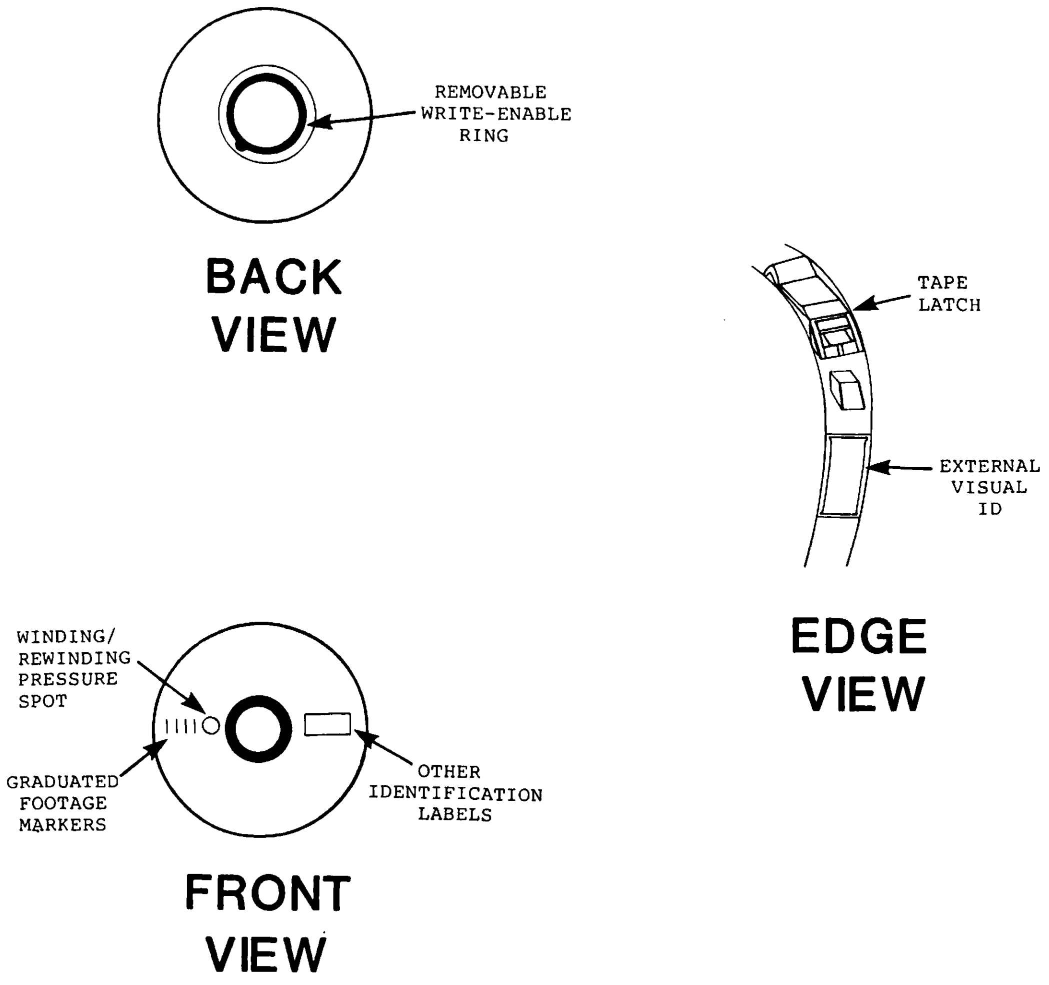
```
    > DECLARE BAR TAPE=(0), FILENO=(-5), RECORDNO=(17)
    >
```

The VISUAL Option: You use VISUAL to specify the external visual identification of the tape reel you are using. The external visual identification is located on the tape reel enclosure, and its position is illustrated in Figure 7-4. VISUAL simplifies the identification of tape reels for MOUNT and DISMOUNT requests. The value is a string of 32 characters that can be any combination of the following:

- Upper and/or lowercase alpha characters
- Numbers
- Any other characters except right and left parentheses, equal signs, commas, apostrophes, and blanks

VISUAL is solely for tape objects. If you do not specify it, the default is 32 blank characters. In the following example, STEVE is a tape object located on logical device number 7. The external visual identification of this tape reel is MYTAPE2:

```
    > DECLARE STEVE TAPE=(7), VISUAL=(MYTAPE2)
    >
```


REMOVABLE WRITE-ENABLE RING
BACK VIEW
TAPE LATCH
EXTERNAL VISUAL ID
EDGE VIEW
WINDING/ REWINDING PRESSURE SPOT
GRADUATED FOOTAGE MARKERS
OTHER IDENTIFICATION LABELS
FRONT VIEW

External Features of a Tape Reel
Figure 7-4

The BYPASS Option: With BYPASS, you indicate to MAGNET whether or not to ignore the labels on your tape and then identify the tape files by number as opposed to file-ids. The value you specify is either YES or NO. There are a few important points to keep in mind when you use this option:

- BYPASS is valid only when you are reading a tape.
- You must always specify, with the LABELS option, the type of labels to be bypassed (either ANSI or IBM).
- BYPASS causes MAGNET to ignore all associated label options (VOLSER, OWNER, FILEID).

This option is for tape objects only. If you do not specify it, the default is NO. In the following example, FILE2 is an ANSI labelled tape object located on logical device number 6. You indicate that you want MAGNET to ignore all the labels and then position at the second logical file on the tape:

```
> DECLARE FILE2 TAPE=(6), LABELS=(ANSI), BYPASS=(YES), FILENO=(2)
>
```

The BUFFERS Option: All magnetic tape I/O is performed through internal buffers. Each tape object must have its own I/O buffers in order to use the READ, WRITE, or MOVE subcommands. You specify the BUFFERS option to indicate how many internal buffers are being used for magnetic tape I/O. The value is a numeral, which may be 0 through 10 only. If you specify BUFFERS with a DECLARE subcommand and then later modify it with a lesser value, the extra buffers are released (returned to the free storage list).

The BUFFERS option is for tape objects only and it defaults to a value of 0. In the following example, you declare a tape object, DOT. It is located on logical device number 2 and has 3 internal I/O buffers:

```
> DECLARE DOT  TAPE=(2), BUFFERS=(3)
>
```

Note

To use a specific tape object with the READ, WRITE, or MOVE data transferral subcommands, you must always specify the BUFFERS option with a value greater than 0. (These subcommands are described in detail later in this chapter.)

The LIST Subcommand

You use LIST to see what object-names you have declared. The subcommand line format is as follows:

```
> LIST
```

MAGNET prints out a chart that shows all declared object-names and the type of the object (DISK or TAPE). In the following example, LIST shows the two declared objects TFILE and DFILE to be tape and disk objects, respectively.

```
> LIST
              object-names                type
-------------------------------------
TFILE                                     TAPE
DFILE                                     DISK
>
```

The DISPLAY Subcommand

DISPLAY shows all options associated with a declared object-name. The subcommand line format is as follows:

```
> DISPLAY object-name
```

The subcommand output shows all possible options that you can specify for an object-name. Tape options appear for tape objects and disk options appear for disk objects. Those options that you previously declared for the specified object-name appear with their values, along with those that automatically default, if you do not specify them. The following is an example of DISPLAY output for a tape object, SYSIN. You first declare SYSIN with four options and their values. The LIST subcommand confirms that SYSIN is a declared tape object. Output from DISPLAY then shows all possible options for SYSIN, but only shows the values for those you declared (TAPE, FILENO, LABELS, and BYPASS) and those you did not declare that default automatically.

```
> DECLARE SYSIN TAPE=(3), FILENO=(5), LABELS=(ANSI), BYPASS=(YES)
> LIST
               object-names                tape
------------------------------------     ----
SYSIN                                      TAPE
> DISPLAY SYSIN
*** Information for TAPE object SYSIN ***
PREV=                                     / NEXT=
PREACTION          =                        POSTACTION         =
TAPE               =      3                 TRACKS             = 9
DENSITY            = 800                    CHARACTERS         = 2
LABELS             = ANSI                   LEVEL              =        0
BYPASS             = YES                    PROTECT            = NO
FORMAT             = FIXED                  PARITY             =
    LRECL              =      0          BFACTOR          =     1
    OFFSET             =      0          BUFFERS          =     0
    MAXIO              =  10000          VISUAL           =
    FILENO             =           5     RECORDNO         =        0
    EXCHANGE           = NONE            FILEID           =
    VOLSER             =                 OWNER            =
    GENERATION         = 0001            VERSION          = 00
    CREATE             =  00000          EXPIRE           =  00000
    ACCESS             =                 SEQUENCE         = 1
    TRANSLATE     = (A*)
    >
```

The next example shows DISPLAY output for a disk object, SYSOUT. You declare SYSOUT with only one option, DISK. The LIST subcommand confirms that SYSOUT is a disk object. Output from the DISPLAY subcommand shows all possible disk options for SYSOUT, but only prints values for DISK, LRECL, and FORMAT. (The values shown for LRECL and FORMAT are default values.)

```
    > DECLARE SYSOUT DISK=(SUE>SOURCE>REV18.3>FILE2)
    > LIST
              object-names                type
    ----------------------------------    ----
    SYSOUT                                DISK
    > DISPLAY  SYSOUT
    *** Information for DISK object SYSOUT  ***
    DISK   = SUE>SOURCE>REV18.3>FILE2
    PREV=                                  / NEXT=
    PREACTION          =                    POSTACTION    =
    LRECL              = 0                  FORMAT        =  VAR/PRIME
    >
```

The RENAME Subcommand

You use RENAME to change the name of a declared object. The subcommand line format is as follows:

```
    > RENAME old-object-name new-object-name
```

In the following example, EMPNOS1 is a previously declared object, as shown in the LIST subcommand output. You rename EMPNOS1 to EMPNOS2, and the new-object-name appears when you give a second LIST subcommand.

```
    > LIST
              object-names               type
--------------------------------         ----
EMPNOS1                                  TAPE
    > RENAME   EMPNOS1   EMPNOS2
    > LIST
              object-names               type
--------------------------------         ----
EMPNOS2                                  TAPE
    >
```

The MODIFY Subcommand

MODIFY changes a declared object or option. For example, you can add an option to an object-name or change the value of an option. The subcommand line format is as follows:

```
    > MODIFY object-name OPTION=(value) [,OPTION=(value)...]
```

On the subcommand line, options can be in any order; you do not have to specify DISK, TAPE, or EXTERNAL as the first option. All options that you can use with the DECLARE subcommand are also valid for MODIFY. There are three important points to remember when you use the MODIFY subcommand:

- You can only modify object-names already declared.
- You can only specify tape options for a tape object.
- You can only specify disk options for a disk object.

In the following example, you declare a tape object, BETA_TEST. You specify the options TAPE, TRACKS, FORMAT, DENSITY, and FILENO with their corresponding values. A LIST subcommand confirms that BETA_TEST is a declared tape object. The output from a DISPLAY subcommand shows the declared options and values for BETA_TEST. You now wish to modify BETA_TEST by adding some options to those you have already declared. With the MODIFY subcommand, you specify the additional options LABELS, LRECL, BFACTOR, and BYPASS. Again, a LIST subcommand confirms that BETA_TEST is a tape object. Now, when you give the DISPLAY subcommand, the output shows not only the options and corresponding values you specified when you first declared BETA_TEST, but also those you specified when you modified it. In both cases with the DISPLAY output, you also see the options that default automatically, if you do not specify them.

```
> DECLARE BETA_TEST TAPE=(0), TRACKS=(9), FORMAT=(VAR/ANSI),
DENSITY=(6250), FILENO=(2)
> LIST
             object-names                      type
__________________________________________   _____
BETA_TEST                                    TAPE
> DISPLAY BETA_TEST
*** Information for TAPE object BETA_TEST ***
PREV=                                       / NEXT=
PREACTION        =                            POSTACTION        =
TAPE             =       0                    TRACKS            = 9
DENSITY          = 6250                       CHARACTERS        = 2
LABELS           = NONE                       LEVEL             =        0
BYPASS           = NO                         PROTECT           = NO
FORMAT           = VAR/ANSI                   PARITY            =
LRECL            =       0                    BFACTOR           =        1
OFFSET           =       0                    BUFFERS           =        0
MAXIO            =  10000                     VISUAL            =
FILENO           =                 2          RECORDNO          =         0
EXCHANGE         = NONE                       FILEID            =
VOLSER           =                            OWNER             =
GENERATION       = 0001                       VERSION           = 00
CREATE           =  00000                     EXPIRE            =  00000
ACCESS           =                            SEQUENCE          =     0
TRANSLATE    = (A*)
> MODIFY BETA_TEST  LABELS=(ANSI), LRECL=(100), BFACTOR=(10),
BYPASS=(YES)
> LIST
             object-names                      type
__________________________________________   _____
BETA_TEST                                    TAPE
> DISPLAY BETA_TEST
*** Information for TAPE object BETA_TEST ***
PREV=                                       / NEXT=
PREACTION        =                            POSTACTION        =
TAPE             =       0                    TRACKS            = 9
DENSITY          = 6250                       CHARACTERS        = 2
LABELS           = ANSI                       LEVEL             =        0
BYPASS           = YES                        PROTECT           = NO
FORMAT           = VAR/ANSI                   PARITY            =
LRECL            =     100                    BFACTOR           =       10
OFFSET           =       0                    BUFFERS           =        0
MAXIO            =  10000                     VISUAL            =
FILENO           =                 2          RECORDNO          =         0
EXCHANGE         = NONE                       FILEID            =
VOLSER           =                            OWNER             =
GENERATION       = 0001                       VERSION           = 00
CREATE           =  00000                     EXPIRE            =  00000
ACCESS           =                            SEQUENCE          =     0
TRANSLATE    = (A*)
>
```

The DELETE Subcommand

This subcommand deletes one declared object-name. When you delete an object-name, you also clear the space it occupied internally within MAGNET. The subcommand line format is as follows:

```
    > DELETE object-name
```

In the following example, output from a LIST subcommand shows that you have two previously declared tape objects, STATS1 and STATS2. You delete STATS2, and a second LIST subcommand reflects the deletion.

```
    > LIST
              object-names              type
    ------------------------------      ----
    STATS1                              TAPE
    STATS2                              TAPE
    > DELETE STATS2
    > LIST
              object-names              type
    ------------------------------      ----
    STATS1                              TAPE
    >
```

The TRANSLATE Subcommand

TRANSLATE indicates, for tape objects only, that you want particular logical records translated to or from industry standard ASCII, EBCDIC, or BCD. Translation is also available for several different seven-track binary formats. For example, suppose you have an 80-character logical record. The first 40 characters specify a person's name in EBCDIC and the last 40 characters specify binary information. You could specify a translation for the first 40 characters to Prime ASCII and the second 40 characters could be read as they are. The subcommand line format for TRANSLATE is as follows:

```
    > TRANSLATE object-name ([repetition factor] edit token list)
```

An edit token specifies how you want particular fields of a logical record translated. An edit token is an alpha character optionally followed by any of the following:

- Upper and/or lowercase alpha characters
- Numbers
- An asterisk
- Any other characters except right and left parentheses, equal signs, commas, apostrophes, and blanks

An edit token list is a group of edit tokens separated by commas. Each edit token or edit token list may be preceded by an optional repetition factor, which is always a number. An example of an edit token is A6, which specifies a translation of six characters to industry standard ASCII. Table 7-3 lists each translation edit token and its function. Facilities are available for inserting fill characters, specifying no translation (reading or writing raw data), and for utilizing user-defined translation tables. (For detailed information on all edit tokens and translation character set tables, see Appendix A. For detailed information on modifying user-defined translation tables, see the description of the LOAD subcommand later in this chapter.)

In the following example of the TRANSLATE subcommand, you are reading from a tape. You first specify a translation of six characters from industry standard ASCII to Prime ASCII, (P6). You next specify a translation, for 40 pairs of fields, of six characters from industry standard ASCII to Prime ASCII, (P6,40(P6)). You next indicate that you want to delete the next two characters in the field (P6,40(P6,D2)). Lastly, any characters that remain in the logical record are translated from industry standard ASCII to Prime ASCII (P6,40(P6,D2),P*) .

```
    > TRANSLATE BETA_TEST (P6,40(P6,D2),P*)
    >
```

If you wish to change a translation, you do it by simply giving another TRANSLATE command line. For example, to change the translation of BETA_TEST shown above, you might type the following:

```
    > TRANSLATE BETA_TEST (B4,30(B7,D3),B*)
    >
```

Remember that you can always check on the translation you specified when you give the DISPLAY command.

Table 7-3
Translation Edit Tokens

Edit Token	Function
A	Enforces industry standard ASCII.
B	Translates Prime ASCII to and from BCD code.
C	Moves the column position within fields of logical records.
D	Deletes characters in logical records.
E	Translates Prime ASCII to and from EBCDIC code.
F	Specifies fill characters.
G	Specifies seven-track packing code 2424 (see Appendix A).
H	Specifies seven-track packing code 4242 (see Appendix A).
I	Specifies seven-track packing code 2466 (see Appendix A).
J	Specifies seven-track packing code 4266 (see Appendix A).
K	Specifies seven-track packing code 6246 (see Appendix A).
L	Specifies seven-track packing code 6426 (see Appendix A).
M	Specifies seven-track packing code 6624 (see Appendix A).
N	Specifies seven-track packing code 6642 (see Appendix A).
O	Specifies no translation.
P	Enforces Prime ASCII.
Q through U	Reserved for future use.
V through Z	Specify user codes.

DATA TRANSFERRAL SUBCOMMANDS

There are four MAGNET subcommands that transfer data within the subsystem. They are READ, WRITE, MOVE, and COPY. READ, WRITE, and MOVE are logical copy operations. COPY is a physical tape copy operation.

The READ Subcommand

You use the READ subcommand to move information in a tape file (source object) to a disk file (destination object). The tape and disk files must be previously declared object-names. (The pre-Rev 18.3 functionality of READ is still supported. See Appendix C.) The subcommand line format for READ is as follows:

```
    > READ tape-object-name disk-object-name
```

In the following example, you first declare a tape object, SYSIN, which is located on logical device 0. You specify the third file on the tape, which has five records per physical tape block with each record 100 characters long. By default you are working with fixed-length records. There are no tape labels. Within the MODIFY subcommand you use another option, BUFFERS, to specify that three internal buffers are being used by the tape object, SYSIN. To perform a READ operation, your BUFFERS value for the tape object must be greater than or equal to 1. Next you declare a disk object, SYSOUT, which is a disk file, DIMPLE, in the UFD SFA. Finally, your READ subcommand line causes the information in SYSIN to be read into SYSOUT.

```
    > DECLARE SYSIN TAPE=(0), FILENO=(3), LRECL=(100), BFACTOR=(5)
    > MODIFY SYSIN BUFFERS=(3)
    > DECLARE SYSOUT DISK=(SFA>DIMPLE), LRECL=(100)
    > READ SYSIN SYSOUT
    >
```

The WRITE Subcommand

The function of the WRITE subcommand is the opposite of READ. You use WRITE to move information in a disk file (source object) to a tape file (destination object). As with the READ subcommand, the disk and tape files must be previously declared object-names. To perform a WRITE operation, the value you specify with the BUFFERS option for a tape object must be greater than or equal to 1. (The pre-Rev 18.3 functionality of WRITE is still supported. See Appendix C.) The subcommand line format for WRITE is as follows:

```
    > WRITE disk-object-name tape-object-name
```

In the following example, you are working with the same two previously declared objects, SYSIN and SYSOUT. However, you wish to move the information in SYSOUT into SYSIN:

```
> WRITE SYSOUT SYSIN
>
```

The MOVE Subcommand

You use this subcommand to simultaneously move information in a source object into a maximum of seven destination objects. Source and destination objects must be previously declared and may be either disk or tape objects. To perform a MOVE operation, the value you specify for the BUFFERS option must be greater than or equal to 1 for each tape object you use. The subcommand line format for MOVE is as follows:

```
> MOVE source-object-name destination-object-name(s)
```

In the following example, you first declare your source object, NEW_DATA, to be a disk file. You next declare the first of your four intended destination objects, PLACE1. This is a tape file with characteristics as shown. You then declare your next three destination objects, PLACE2 (a tape file), PLACE3 (a tape file), and PLACE4 (a disk file). These files have the characteristics shown. The output from a LIST subcommand shows the objects you have just declared. Finally, you use the MOVE subcommand to simultaneously move the information in NEW_DATA to PLACE1, PLACE2, PLACE3, and PLACE4.

```
> DECLARE NEW_DATA DISK=(SFA>STATS), LRECL=(80)
> DECLARE PLACE1 TAPE=(0), LRECL=(100), BFACTOR=(10), BUFFERS=(3),
FORMAT=(VAR/ANSI), FILENO=(10)
> DECLARE PLACE2 TAPE=(1), LRECL=(80), BFACTOR=(20), BUFFERS=(6)
> MODIFY PLACE2 FORMAT=(VAR/IBM), FILENO=(1)
> DECLARE PLACE3 TAPE=(5), LRECL=(150), BFACTOR=(1), BUFFERS=(9),
LABELS=(ANSI), LEVEL=(2), VOLSER=(597219), OWNER=(PARIS), FILEID=
(SRCPRG1)
> DECLARE PLACE4 DISK=(PARIS>SOURCE>PROGRAM.FTN), LRECL=(80)
> LIST
          object-names               type
--------------------------------     ----
NEW_DATA                             DISK
PLACE1                               TAPE
PLACE2                               TAPE
PLACE3                               TAPE
PLACE4                               DISK
> MOVE  NEW_DATA  PLACE1  PLACE2  PLACE3  PLACE4
>
```

The COPY Subcommand

You use the COPY subcommand to copy information from a source tape to one or more destination tapes. This is a physical, as opposed to a logical, copy function. (The pre-Rev 18.3 functionality of COPY is still supported. See Appendix C.)

Note

> Your tape will be positioned before it is copied. You indicate the starting position on the tape by use of the FILENO and RECORDNO options with the DECLARE and MODIFY subcommands. (These two options are discussed earlier in this chapter.) The FILENO/RECORDNO pair must be an absolute position; the values for both options must be greater than or equal to 0. If they are equal to 0, none of the tapes are pre-positioned. (For more information on tape positioning, see the POSITION subcommand later in this chapter.)

The format of the COPY subcommand line is as follows:

```
                                    ( (n) )           ( (YES) )
>COPY source destination(s) AMOUNT = {     } ,PRINT = {       }
                                    ( (*) )           ( (NO)  )
```

You may specify up to eight object-names in the subcommand line (one source-object-name and from one to seven destination-object-names). The tape objects must be previously declared and must refer to different tape drives. You must always specify the AMOUNT option in the COPY subcommand line.

The AMOUNT Option: The AMOUNT option identifies the number of physical tape files to be copied. (Copying is done from one file marker to another file marker.) There is no default for AMOUNT. You must specify this option on the COPY subcommand line.

Note

> Since the COPY subcommand performs a physical tape copy as opposed to a logical tape copy, tape labels are not recognized.

There are two possible values you can specify for the AMOUNT option, either a numerical value (n) or an asterisk (*). A numerical value identifies the number of physical files that are to be copied. The value can be a number between 1 and 32,767. If you wish to copy everything on your tape until you reach the first filemarker after the physical end-of-tape (EOT), then you specify an asterisk (*) for a value. If the physical EOT (end of tape) is detected, and if you did not specify AMOUNT=(*), and then a single file marker is read, copying stops and a new source tape is requested. (See Appendix B for more information.) Likewise, if the physical EOT is detected on a destination tape, a single file marker is written and a new tape is requested. (See Appendix B.)

The PRINT Option: The PRINT option specifies that the size, in words, of the first physical record of each tape file is printed on your terminal. There are two values you can specify for this option, either YES or NO. The default value is NO.

If you specify PRINT=(YES), a table like the following is printed as the copy operation proceeds:

```
    File #    Words Read    Total Blocks Read
         1           200                  250
         2           350                   97
         3           790                 1123
         .             .                    .
         .             .                    .
         .             .                    .
```

When MAGNET has completed copying a file, the total number of physical records in that file is also displayed in the third column of the table. The PRINT option replaces the "PRINT RECORD SIZES?" question in the pre-Rev 18.3 MAGNET dialog. (See Appendix C for additional information.)

The following example illustrates the use of the COPY subcommand and the AMOUNT option. You first declare a tape object, SYSIN, that is located on logical device 0. The FILENO/RECORDNO option pair instructs MAGNET to rewind the tape and then position it to file number 1, record number 1. You then declare SYSOUT, a tape object located on logical device number 1, and SYSOUT2, a tape object located on logical device number 2. With SYSOUT and SYSOUT2, the FILENO/RECORDNO option pair functions the same way as in the SYSIN declaration. You then give a COPY subcommand line which copies all of SYSIN, the source-object-name, to SYSOUT and SYSOUT2, the two destination-object-names. Remember that all of SYSIN is copied because you specified an asterisk (*) as the value for the AMOUNT option.

```
    > DECLARE SYSIN TAPE=(0), FILENO=(1), RECORDNO=(1)
    > DECLARE SYSOUT TAPE=(1), FILENO=(1), RECORDNO=(1)
    > DECLARE SYSOUT2 TAPE=(2), FILENO=(1), RECORDNO=(1)
    > COPY SYSIN SYSOUT SYSOUT2 AMOUNT=(*), PRINT=(NO)
    >
```

ADDITIONAL SUBCOMMANDS

The POSITION Subcommand

The POSITION subcommand positions a tape to a specific FILENO/RECORDNO pair. Positioning can be either absolute (rewind the tape first then space forward) or relative (space forward or backward from your current position on the tape). Before tape positioning, both the FILENO and RECORDNO options must be set by the DECLARE or MODIFY subcommands. For absolute positioning, the values for both the FILENO and RECORDNO options must be greater than or equal to 1. For relative positioning the two options can have any values. (The pre-Rev 18.3 functionality of POSITION is still supported. See Appendix C.) The subcommand line format for POSITION is as follows:

```
                                  (ABSOLUTE)
    > POSITION  object-name  MODE= 
                                  (RELATIVE)
```

The object-name is the name of the tape you wish to position and must always be a previously declared tape object. You must always specify the type of positioning (absolute or relative) with the MODE option.

The MODE Option: You use the MODE option to specify either absolute or relative tape positioning on a POSITION subcommand line. When you use the POSITION subcommand, you must always specify either ABSOLUTE or RELATIVE for the MODE option.

The following example illustrates the use of both the POSITION subcommand and the MODE option. Assume that you have four physical files with six physical records in each of the files. You first declare T1, a tape object located on logical device 0. You set the values of the FILENO and RECORDNO options to be 3 and 6, respectively. Next you give a POSITION subcommand that specifies absolute positioning for T1. This instructs MAGNET to rewind the tape, space forward two file markers, and then space forward five records.

```
    > DECLARE  T1  TAPE=(0), FILENO=(3), RECORDNO=(6)
    > POSITION  T1 MODE=(ABSOLUTE)
    >
```

There are two ways to rewind your tape. In the following example, you rewind X by giving 1 as the value for both the FILENO and RECORDNO options and then specifying absolute positioning with the POSITION subcommand. You also rewind Y by giving very large, negative values for both FILENO and RECORDNO and then specifying relative positioning on the POSITON subcommand line.

```
> DECLARE X TAPE=(0), FILENO=(1), RECORDNO=(1)
> DECLARE Y TAPE=(1) FILENO=(-1000000) RECORDNO=(-1000000)
> POSITION X MODE=(ABSOLUTE)
> POSITION Y MODE=(RELATIVE)
>
```

The QUIT Subcommand

You use QUIT when you wish to exit from MAGNET and return to PRIMOS. The subcommand line format is as follows:

```
> QUIT
OK,
```

Note

MAGNET ignores anything you type on the subcommand line that follows QUIT.

The /* Subcommand

You use this subcommand to put comments in your MAGNET dialog. It is the same as the PRIMOS comment command. Your comment can be up to 1000 characters long. The subcommand line format is as follows:

```
> /* ccccccccccccccccccc...
>
```

ADVANCED MAGNET

There are advanced MAGNET features, subcommands, and options that you will use less frequently. These features supplement the basic MAGNET functions discussed up to this point in the chapter.

Advanced Options

There are 15 additional options that you can specify with the DECLARE and MODIFY subcommands.

The EXTERNAL Option: When you use MAGNET, there may be specific sets of options that you will want to use more than once. The EXTERNAL option allows you to save and reload these options so that you do not have to retype them each time. You can also use EXTERNAL with the SAVE subcommand. (See the description of SAVE later in this chapter.)

Options are stored for later use in PRIMOS global variables. You can create these sets of options at PRIMOS level by using the PRIMOS command SET_VAR. To do this, your global variable file must be activated. The following example illustrates how you can set up groups of options at PRIMOS level by using SET_VAR.

```
OK, SET_VAR .SYSIN := TAPE=(0), LRECL=(80), BUFFERS=(2),
FILENO=(1)
OK,
```

For a description of PRIMOS global variable commands, refer to The CPL User's Guide.

The value you specify for the EXTERNAL option must be a valid PRIMOS global variable name. It is a string of no more than 32 characters. The first character in the string must always be a period (.). The other characters in the string can be any mixture of the following:

- Upper and/or lowercase alpha characters
- Numbers
- Underscores
- Periods

With the DECLARE or MODIFY subcommands, you may specify EXTERNAL first in the option list on the subcommand line. However, no matter where EXTERNAL appears in the option list, MAGNET ignores any options that follow it. If you specify EXTERNAL as the first option with a DECLARE subcommand, the TAPE or DISK options must appear first within the CPL variable.

Note

When you give the DISPLAY subcommand, any edit tokens you specified with the TRANSLATE subcommand are shown in the output. These edit tokens are not regarded as options and thus cannot be saved or loaded by the EXTERNAL option.

There are a few points to remember when you use the EXTERNAL option:

- A global variable cannot contain an EXTERNAL option. EXTERNAL can only appear within MAGNET.
- Any options that appear within a global variable overwrite any of the same options already declared or modified.
- Global variables cannot contain more than 1024 characters.
- You may specify the same name for an EXTERNAL value and an object.

In the following example, at PRIMOS command level, you first create a global variable file, IO.GVAR. A LIST_VAR command shows that no global variables are defined. You then create two global variables for the file. These are .SYSIN and .SYSOUT. Output from a second LIST_VAR command shows that these two variables are now defined. Next, invoke MAGNET and declare two objects, SYSIN and SYSOUT. You specify the EXTERNAL option for each of these objects, the values being .SYSIN and .SYSOUT, respectively. This instructs MAGNET to get the global variables values in .SYSIN and .SYSOUT and put them into the internal variables SYSIN and SYSOUT. Next, you give the LIST subcommand, which confirms that SYSIN and SYSOUT are declared objects. Finally, the DISPLAY subcommand output for both SYSIN and SYSOUT shows all the values associated with variables SYSIN and SYSOUT.

```
OK, DEFINE_GVAR IO.GVAR -CREATE
OK, LIST_VAR
No global variables are defined.
OK, SET_VAR .SYSIN := TAPE=(0),BUFFERS=(2),LRECL=(80),FILENO=(1)
OK, SET_VAR .SYSOUT := DISK=(TESTDIR>SFA>FILE3),LRECL=(80)
OK, LIST_VAR
.SYSOUT                  DISK=(TESTDIR>SFA>FILE3),LRECL=(80)
.SYSIN                   TAPE=(0),BUFFERS=(2),LRECL=(80),FILENO=(1)
OK, MAGNET
[MAGNET, Rev. 18.3]
> DECLARE SYSIN EXTERNAL=(.SYSIN)
> DECLARE SYSOUT EXTERNAL=(.SYSOUT)
> LIST
          object-names                  type
--------------------------------        ----
SYSIN                                   TAPE
SYSOUT                                  DISK
> DISPLAY SYSIN
*** Information for TAPE object SYSIN ***
PREV=                                  / NEXT=
PREACTION          =                    POSTACTION     =
TAPE               =       0            TRACKS         = 9
DENSITY            = 800                CHARACTERS     = 2
LABELS             = NONE               LEVEL          =     0
BYPASS             = NO                 PROTECT        = NO
FORMAT             = FIXED              PARITY         =
LRECL              =     80            BFACTOR       =   1
OFFSET             =      0            BUFFERS       =   2
MAXIO              =  10000            VISUAL        =
FILENO             =           1       RECORDNO      =      0
EXCHANGE           = NONE              FILEID        =
VOLSER             =                   OWNER         =
GENERATION         = 0001              VERSION       = 00
CREATE             =  00000            EXPIRE        =  00000
ACCESS             =                   SEQUENCE      = 0000
TRANSLATE    = (A*)
> DISPLAY SYSOUT
*** Information for DISK object SYSOUT ***
DISK = TESTDIR>SFA>FILE3
PREV=                                 / NEXT=
PREACTION          =                   POSTACTION    =
LRECL              =     80            FORMAT        = VAR/PRIME
> QUIT
OK,
```

The PREACTION and POSTACTION Options: These options specify the action to be taken when opening and closing disk and tape files. You can, when opening, position to the end of the disk file so that new information read into it will be concatenated at the end. It is also possible to rewind a reel of tape before searching for and opening a tape file. After a tape file has been closed, you may rewind and/or unload the tape reel.

The PREACTION option has two possible values: APPEND, for disk objects; and REWIND, for tape objects. POSTACTION also has two possible values: REWIND and UNLOAD, both for tape objects only. Both PREACTION and POSTACTION default to six blanks.

In the following example, you declare a disk object, SUEA, to be a disk file, SUEFILE. You specify the PREACTION option with an APPEND value. This instructs MAGNET to position SUEFILE at its end upon opening.

```
> DECLARE SUEA DISK=(SUEFILE), PREACTION=(APPEND)
>
```

Note

The PREACTION option with the APPEND value is useful only when you are reading into a disk file.

In the next example, you declare a tape object, STEVEP, to be a tape file located on logical device number 7. Each logical record in the file is 80 characters long. You specify the PREACTION option with a REWIND value. This instructs MAGNET to rewind the tape before opening file number 23.

```
> DECLARE STEVEP TAPE=(7), PREACTION=(REWIND), LRECL=(80),
FILENO=(23)
>
```

Note

The PREACTION option with the REWIND value is useful only if your tape is positioned beyond a desired location.

In the next example, you are working with the same tape object, STEVEP. However, you are now working with file number 42 on the tape, and you specify the POSTACTION option with a REWIND value. This instructs MAGNET to rewind the tape after closing tape file number 42:

```
> DECLARE STEVEP TAPE=(7), POSTACTION=(REWIND), LRECL=(80),
FILENO=(42)
>
```

In this final example, you are again working with file number 42 in the tape object, STEVEP. You specify the POSTACTION option with an UNLOAD value. This instructs MAGNET to rewind and unload (dismount) the tape after closing tape file number 42:

```
> DECLARE STEVEP TAPE=(7), POSTACTION=(UNLOAD), LRECL=(80),
FILENO=(42)
>
```

The PREVCHAIN and NEXTCHAIN Options: It is possible to link together or "chain" a number of tape or disk objects. You would do this if, for example, you wanted to specify which tape should be mounted when you reach the end of the tape you are currently using. If you do not specify chaining and the end-of-tape is detected, MAGNET requests that a new tape be mounted. However, since MAGNET does not initially recognize any characteristics of the new tape, it has no way of knowing if the new tape is the correct one. Consequently, this procedure is only useful for unlabelled tapes.

If you are using labelled tapes and you expect to use a multi-reel file, you must specify chaining. When you declare your tape object, you specify data about the first reel of tape the file is on. You must also specify the NEXTCHAIN option on the DECLARE or MODIFY subcommand lines. The NEXTCHAIN option contains the name of another tape object. When end-of-tape is detected, MAGNET rewinds the tape, looks up the information in the tape object specified by NEXTCHAIN, and requests that the new tape be mounted.

When you specify a chain of tapes, you should include the BUFFERS option only for the first object in the chain. If you specify BUFFERS for any object after the first in the chain, those buffers are released when that tape object is activated. Other options that specify physical characteristics of the tape (such as TAPE, TRACKS, and DENSITY) or logical characteristics of the file (such as LRECL, BFACTOR, FORMAT, and LABELS) should have the same value among all objects in the chain. In order to identify a new tape to be mounted, you should specify an external visual identification by using the VISUAL option. (For more information, see the description of VISUAL earlier in this chapter.)

The NEXTCHAIN option identifies the next object in a chain. PREVCHAIN identifies the previous object, but it is not currently used by MAGNET in this release. In addition, NEXTCHAIN is currently used only for tape objects. Values of both options are names of other tape or disk objects. A tape object cannot point to a disk object or vice versa. Both options are for tape and disk objects, and the defaults are blank characters.

In the following example, you declare a tape object, TAXES, with a number of options. It is the first object in a chain, and the next object in the chain is TAXES2. TAXES2 has exactly the same option values as TAXES except for OWNER, VOLSER, and VISUAL. Note that, since you did not specify NEXTCHAIN for TAXES2, it is the last object in the chain. When end-of-tape is detected on TAXES, the tape is rewound and a message appears on either your terminal or the operator's console (depending on whether you specify -USER or -OPERATOR at PRIMOS command level; see Appendix B). After the new tape specified by TAXES2 is mounted and either you or the operator replies to MAGNET's message, processing continues with the object TAXES2.

```
> DECLARE TAXES TAPE=(0), LRECL=(100), BFACTOR=(20)
> MODIFY TAXES DENSITY=(6250), FORMAT=(VAR/ANSI)
> MODIFY TAXES LABELS=(ANSI), LEVEL=(2)
> MODIFY TAXES FILEID=(1980), OWNER=(SIRAP)
> MODIFY TAXES VOLSER=(SFATAX), VISUAL=(X125)
> MODIFY TAXES BUFFERS=(7), MAXIO=(12000)
> MODIFY TAXES NEXTCHAIN=(TAXES2)
> /*
> DECLARE TAXES2 TAPE=(0), LRECL=(100), BFACTOR=(20)
> MODIFY TAXES2 DENSITY=(6250), FORMAT=(VAR/ANSI)
> MODIFY TAXES2 LABELS=(ANSI), LEVEL=(2)
> MODIFY TAXES2 FILEID=(1980), OWNER=(NEVETS)
> MODIFY TAXES2 VOLSER=(SFATX2), VISUAL=(X126)
> MODIFY TAXES2 MAXIO=(12000)
>
```

The ACCESS Option: ACCESS identifies the accessibility of a tape object. Accessibility is installation-dependent. Prime software does not check the accessibility fields of the VOL1, HDR1, EOF1, and EOV1 labels. Consequently, this option is provided for tape protection purposes on other systems.

The value you specify, the access code, is installation-dependent. In other words, it is not recognized by Prime software but will be recognized by software of another manufacturer. An access code is a single character that may be any one of the following:

- An upper or lowercase alpha character
- A number
- Any other character except right or left parentheses, an equal sign, a comma, an apostrophe, or a blank

ACCESS is strictly a tape option. It defaults to a blank character, which signifies no protection. In the following example, you declare a tape object, SYSIN. It is located on logical device number 3 and has ANSI standard labels. You are working with the file MYFILE on tape number 1254. The access code is Y.

```
> DECLARE   SYSIN   TAPE=(3),   LABELS=(ANSI),    FILEID=(MYFILE),
VOLSER=(1254), ACCESS=(Y)
>
```

Note

There is an accessibility field in both the VOL1 and HDR1 labels. These do not necessarily have the same value. The HDR1 field is set by the ACCESS option. The VOL1 field can be set by the PRIMOS LABEL command when you are initializing your reel of tape. (For more information on LABEL, see Chapter 6.)

The CREATE and EXPIRE Options: These two options specify the creation and expiration dates of a labelled tape file. These dates are placed in the appropriate fields of the HDR1, EOF1, and EOV1 labels, and provide protection. Note that if the current date is earlier than the expiration date, the file cannot be overwritten. All creation and expiration dates are stored in MAGNET (both internally and in label fields) in Julian format.

Note

The first file on your tape should have the latest expiration date. This is because, when you overwrite a file, all succeeding data on the tape is considered invalid.

The values you specify with CREATE and EXPIRE are one of the following:

- A date in the form YYDDD, where YY represents the year, and DDD represents the day of the year (Julian date)
- An asterisk (*), which indicates the current date
- MM/DD/YY, where MM represents the month (January is represented as 01), DD is the day of the month, and YY is the year

CREATE and EXPIRE are strictly tape options. They both default to five 0's. This indicates that the file has already expired. (You have data on the tape that may be overwritten.) In the following example, you declare VITAL to be a tape object located on logical device 0. The tape has ANSI standard labels. With three subsequent MODIFY subcommands you specify additional options for VITAL. The tape has two levels of labels, the owner of the tape is JOHN_RODDY, and the volume serial number is 847519. Finally, the creation date of the tape is the current date, and the expiration date you specify is June 10, 1987.

```
    > DECLARE VITAL TAPE=(0), LABELS=(ANSI)
    > MODIFY VITAL LEVEL=(2), OWNER=(JOHN_RODDY)
    > MODIFY VITAL VOLSER=(847519)
    > MODIFY VITAL CREATE=(*), EXPIRE=(06/10/87)
    >
```

The GENERATION and VERSION Options: The GENERATION option specifies an edition (the most current update) of a file. The VERSION option specifies a printing (a copy of the most current edition) of a file. Both options are stored in the Generation and Generation-Version fields of the HDR1, EOF1, and EOV1 labels.

The values you specify for these two options are numbers in the form nnnn (for GENERATION) and nn (for VERSION). The valid values for GENERATION are 0001 through 9999. The valid values for VERSION are 00 through 99. GENERATION defaults to 0001 and VERSION defaults to 00.

The followng example illustrates the subcommand line for the 23rd printing of the 52nd edition of the tape object CLOUDS. It is an IBM labelled tape located on logical device number 4. You are working with the file RAINBOW.

```
    > DECLARE CLOUDS TAPE=(4),LABELS=(IBM),FILEID=(RAINBOW),
    GENERATION=(0052),VERSION=(23)
    >
```

The OFFSET Option: OFFSET specifies the length in characters of any field in a physical tape block which precedes the data portion of that block. This option is for tape objects only. The value is a number from 0 to 99. The default value is 0.

In the following example, you declare the tape object, BLUESKY, located on logical device number 1. You are working with file number 12, and each logical record is 80 characters long. The offset field preceding the data portion of each block is 10 characters long. Note that, in this example, the BFACTOR option is not specified so it defaults to a value of 1. However, because you specify OFFSET=(10), each physical block in this tape file will be 90 characters long.

```
    > DECLARE BLUESKY TAPE=(1),FILENO=(12),LRECL=(80),OFFSET=(10)
    >
```

Note

If you specify LABELS=(ANSI) and LEVEL=(2), the value of OFFSET is stored in the Buffer Offset field of HDR2, EOF2, and EOV2 labels.

The SEQUENCE Option: This option specifies the order of a file within a set of files created simultaneously. This information corresponds to the ANSI file sequence number and the IBM data set sequence number, both of which are found on the HDR1 label.

SEQUENCE is a tape option for labelled tape files. You use SEQUENCE only when reading from these files. The value you specify is a four-digit number, 0001 through 9999. The default value is 0001. In the following example, you declare a tape object, OURS, located on logical device number 2. Each logical record is 100 characters long. The tape contains ANSI standard labels, and you are working with the labelled file with the sequence number 0023.

```
> DECLARE OURS TAPE=(2),LRECL=(100),LABELS=(ANSI),SEQUENCE=(0023)
>
```

The CHARACTERS Option: You use CHARACTERS when reading or writing odd-length physical blocks. All normal data transfers to or from a tape are performed in terms of words. Since a word is composed of two bytes (or characters), only an even number of characters can normally be transferred between memory and a tape. Note that an odd number of words equals an even number of characters (five words = 10 characters). To write a true odd-length physical block, the tape drive must be instructed to transfer only one character per word. To get a one-character-per-word string, a two-character-per-word string is doubled in length and each character from the original string is placed in the right-hand byte of a word. This accounts for the restrictions on the LRECL and BFACTOR options. (For more information, see the descriptions of LRECL and BFACTOR earlier in this chapter.)

For output, when writing a tape file, CHARACTERS causes each block to be exploded from a two-character-per-word string to a one-character-per-word string. For input, when reading a tape file, each physical block is read as a one-character-per-block record and then imploded to a two-character-per-word string. See Figures 7-5 (A) and (B).

Note

If the CHARACTERS and EXCHANGE options are used together on input, CHARACTERS is processed first. On output EXCHANGE is processed first.

CHARACTERS is strictly for tape objects. The value you specify is either 1 or 2. The default value is 2. Note that when you specify a value of 1, the value of LRECL * BFACTOR must be less than or equal to 6143. In the following example, LEDGER is a tape object located on logical device number 4. The logical record length is 37 and the blocking factor is 11. You are working with the 20th file on the tape. Each physical block is to be written as a true odd-length physical block.

```
> DECLARE LEDGER TAPE=(4),LRECL=(37),BFACTOR=(11),CHARACTERS=(1),
FILENO=(20)
>
```

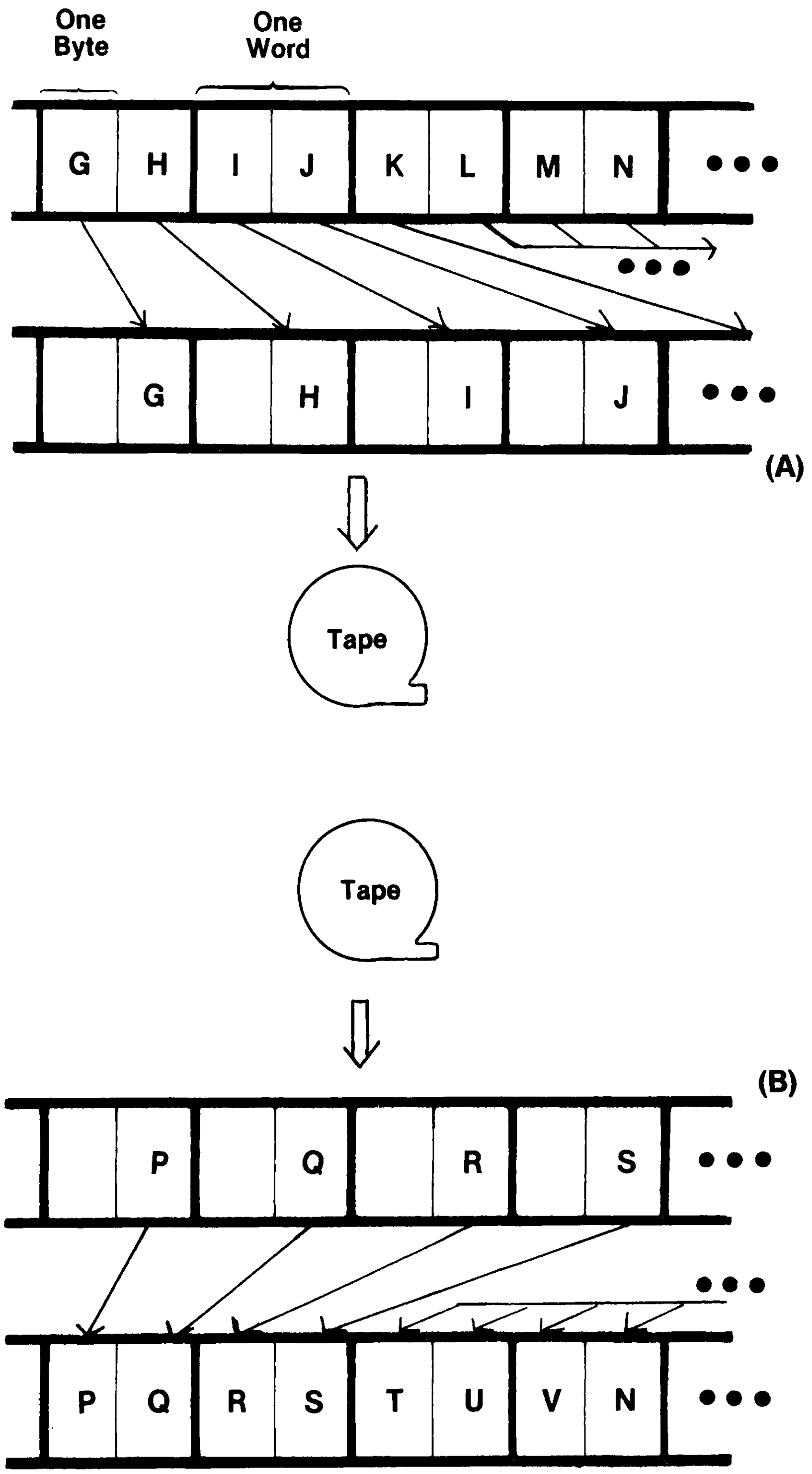
The EXCHANGE Option: When you write a physical block to tape on a Prime system, each high-order byte within a 16-bit word is written preceeding the low-order, or second byte. Certain operating systems of other manufacturers read or write the low-order byte first, and then read or write the high-order byte second. The EXCHANGE option allows you to compensate for this. On output, EXCHANGE causes pairs of bytes in a physical block to be swapped. On input, EXCHANGE causes each byte in a physical block to be swapped with its neighbor before any other processing takes place. See Figures 7-6 (A) and (B).

Note

If the physical block has an odd-length, the last byte is not swapped.

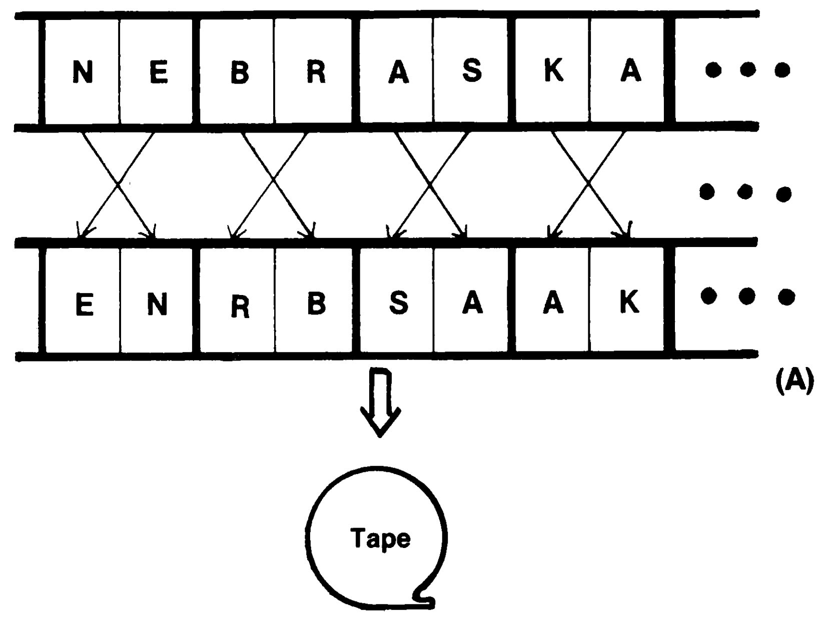
EXCHANGE is a tape option only. The value you specify is either PHYSICAL or NONE. The default value is NO. In the following example, you declare a tape object, JJR, located on logical device number 6. The logical record length is 20 and you are working with the first file on the tape. All bytes in each word of the physical block are to be swapped since you specified PHYSICAL as the value for EXCHANGE.

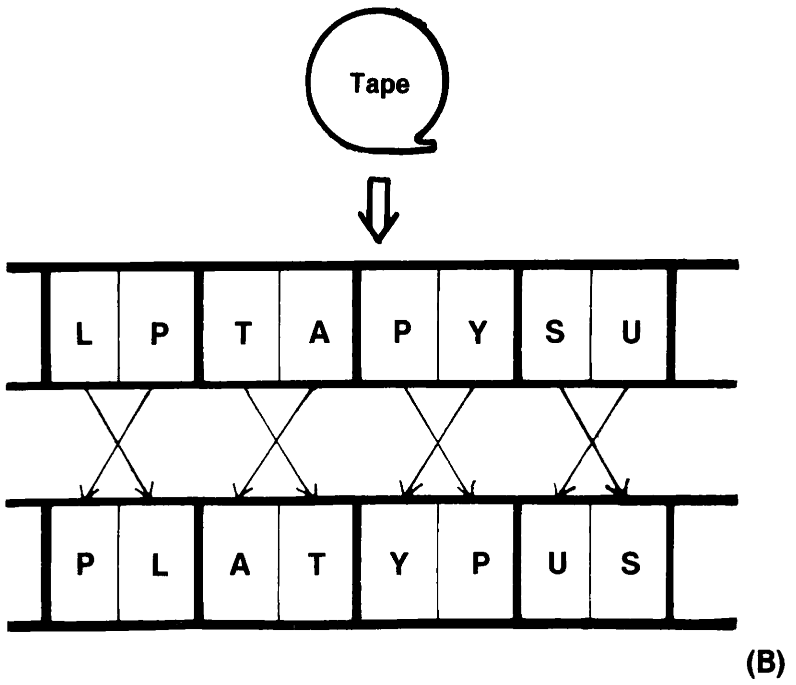
```
> DECLARE JJR TAPE=(6),LRECL=(20),FILENO=(1),EXCHANGE=(PHYSICAL)
>
```



The CHARACTERS Option
(A) Tape Output
(B) Tape Input

Figure 7-5





The EXCHANGE Option
(A) Tape Output
(B) Tape Input

Figure 7-6

The MAXIO Option: This option specifies the maximum number of characters that the tape drive will attempt to read. All tape read operations request that the tape drive read and fill a buffer of a certain size. On some Prime systems, this number varies because of I/O window sizes.

This option is for tape objects only. The value is a number between 2000 and 12,288. The default value is 10,000 characters (5000 words). In the following example, you modify a previously declared tape object, JJR, with the MAXIO option. You specify a value of 9500. This means that the maximum number of characters requested on input is 9500.

```
> MODIFY JJR MAXIO=(9500)
>
```

The SAVE Subcommand

You use SAVE to save a group of options in a PRIMOS global variable. MAGNET saves only those options that you declared or modified to a value other than their default. The subcommand line format is as follows:

```
> SAVE object-name EXTERNAL=(global-variable-name)
```

You must always specify the EXTERNAL option on the SAVE subcommand line. The value of EXTERNAL is a PRIMOS global-variable-name. (For more information, see the description of EXTERNAL earlier in this chapter. For additional information on PRIMOS global variables, refer to The CPL User's Guide.) In the following example, all of the options previously specified for JJR are stored, along with their values, in the global variable name .SUE:

```
> SAVE JJR EXTERNAL=(.SUE)
>
```

The LOAD Subcommand

You use this subcommand to load a user-defined translation table. The internal names of these tables are V, W, X, Y, and Z. MAGNET access these tables when you specify them as edit tokens in a TRANSLATE subcommand. (For more information, see the description of the TRANSLATE subcommand earlier in this chapter, and also Appendix A.) Each user-defined translation table is 512 bytes (256 words) long. The first 256 bytes are used when you are reading from a tape; the last 256 bytes are used when writing to a tape.

You must specify three options on the LOAD subcommand line. They are TYPE, LINES, and DISK. You may specify them in any order. The DISK option identifies the PRIMOS filename or pathname that contains your translation table. LINES and TYPE are described later in this chapter. The LOAD subcommand line format is as follows:

```
                           ⎧ (BIN)  ⎫
                           ⎪ (CHAR) ⎪
> LOAD table-name TYPE=    ⎨ (OCT)  ⎬  , LINES=(n), DISK=(pathname)
                           ⎪ (DEC)  ⎪
                           ⎩ (HEX)  ⎭
```

The TYPE Option: TYPE specifies the interpretation of the characters in the disk file you name as the value for DISK in the LOAD subcommand line. You use TYPE only with the LOAD subcommand, and you must always specify it. There is no default value; you must specify one of the following five values:

- BIN (binary number character strings)
- CHAR (ASCII character strings)
- OCT (octal number character strings)
- DEC (decimal number character strings)
- HEX (hexadecimal number character strings)

All binary digit strings must be eight digits (or characters) long. All octal and decimal digit strings must be three digits (or characters) long. All hexadecimal digit strings must be two digits (or characters) long.

The LINES Option: This option tells MAGNET how many lines to read in the disk file you name as the value for DISK in the LOAD subcommand line. You use LINES only with the LOAD subcommand, and you must always specify it. The value is a number that is a divisor of 512, namely 1, 2, 4, 8, 16, 32, 64, 128, 256, or 512. There is no default value. The number of binary, octal, decimal, or hexadecimal numbers (or characters) that must appear on each line is 512 divided by the value of the LINES option.

Note

For readability, it is not recommended that a small line count (1, 2, 4, 8, or 16) be used, especially if you specify TYPE=(BIN).

Creating Translation Tables: You use a translation table to:

- Convert from one character set to another
- Convert from upper to lowercase and vice versa
- Set or reset certain bits in every byte of a character string

An EBCDIC to ASCII conversion table is an example of a tool you can use to convert from one character set to another. (For more information, see Appendix A.) An example of case conversion can be found in the TRANSLATE function of PL/I:

```
STRING=TRANSLATE(STRING, 'ABCDEFGHIJKLMNOPQRSTUVWXYZ',
   'abcdefghijklmnopqrstuvwxyz');
```

The final function of a translation table, that of setting bits, is found in the industry-compatible ASCII to Prime ASCII translation table. (See Appendix A.)

The syntax you must use for creating your own translation tables is extremely rigid and precise. You have a choice of representing your table in ASCII character strings or binary, octal, decimal, or hexadecimal number character strings.

The ASCII character string representation is useful for case conversion and special symbol translation. With this format, you place the same number (n) of characters on 512 divided by n lines. Every incoming character to be translated actually represents an eight-bit number between 0 and 255. This number is used as an index into the table to find the output character (the translated character). For example, a translation table to convert from lower to uppercase would have the letters A through Z in absolute table positions starting from 0, represented by 225 through 250 decimal.

Each translation table must be 512 bytes long. The first 256 characters form an input table for use when reading from a tape. The second set of 256 characters forms an output table for use when writing to a tape. In the previous case of the lower to uppercase conversion table, it probably makes no sense to have an output table, yet you must include one. A solution is to include just a number of blank lines. Each sequence of characters on every line must begin in column 1.

Note

It is important to keep in mind that blanks are significant. They are treated as characters.

The binary number character string representation is useful when you translate characters that cannot be printed (control characters, for example). Each sequence of binary number character strings must begin in column 1. Each binary number may include the ASCII characters 0 and 1 only. Each number must be exactly eight characters long and must be separated from its neighbor by exactly one space.

The octal, decimal, and hexadecimal number character string representations are exactly the same as the binary representation except:

- For octal numbers, the characters 0 through 7 are allowed.
- For decimal numbers, the octal numbers plus 8 and 9 are allowed.
- For hexadecimal numbers, the decimal numbers plus A through F and a through f are allowed.

These representations are also useful when you translate characters that cannot be printed, and provide a more compact notation than binary. As in the binary representation, all numbers must begin in column 1 and must be separated from their neighbors by only one space. As an example, consider the industry-standard ASCII to Prime ASCII and the Prime ASCII to industry-standard ASCII translation tables in Appendix A. These tables include binary, octal, decimal, and hexadecimal representations. A user table which included all the hexadecimal numbers from both tables in the Appendix would comprise a valid user translation table. All the octal, decimal, or binary numbers would also comprise valid tables.

Suppose that the industry-standard ASCII-Prime ASCII table was found in a disk file, ASC_BIN.TABLE. Also, suppose that this table consists of binary digit strings, eight per line (64 lines). The LOAD subcommand line to load this file (containing the table) into internal table V (edit token V) is:

```
> LOAD V DISK=(ASC_BIN.TABLE), TYPE=(BIN), LINES=(64)
>
```

SAMPLE MAGNET SESSION

The following is a sample MAGNET terminal session. Each step of the session is explained in procedural format following the example.

```
OK, MAGNET
[MAGNET Rev 18.3]
> DECLARE SYSIN DISK=(DC>INTERIOR>NATL_PARK>YOSEMITE),LRECL=(128)
> MODIFY SYSIN FORMAT=(FIXED)
> /*
> DECLARE SYSOUT1 TAPE=(3),DENSITY=(1600),FILENO=(4),LRECL=(141)
> MODIFY SYSOUT1 BFACTOR=(11),CHARACTERS=(1),EXCHANGE=(YES)
> MODIFY SYSOUT1 FORMAT=(FIXED),BUFFERS=(3),POSTACTION=(UNLOAD)
> TRANSLATE SYSOUT1 (A120,10(F*),O4,3(F),A4)
> /*
> DECLARE SYSOUT2 TAPE=(3),DENSITY=(1600),FILENO=(1),LRECL=(141)
> MODIFY SYSOUT2 BFACTOR=(11),CHARACTERS=(1),EXCHANGE=(YES)
> MODIFY SYSOUT2 FORMAT=(FIXED),VISUAL=(NPCA0001)
> MODIFY SYSOUT2 POSTACTION=(UNLOAD)
> TRANSLATE SYSOUT2 (A120,10(F*),O4,3(F),A4)
> /*
> MOVE SYSIN SYSOUT1
> /*
> DECLARE AUDIT_TAPE_1 TAPE=(0),FILENO=(1),RECORDNO=(1)
> DECLARE AUDIT_TAPE_2 TAPE=(1),FILENO=(1),RECORDNO=(1)
> /*
> COPY AUDIT_TAPE_1  AUDIT_TAPE_2 AMOUNT=(274)
> /*
> POSITION AUDIT_TAPE_1 MODE=(ABSOLUTE)
> POSITION AUDIT_TAPE_2 MODE=(ABSOLUTE)
> /*
> QUIT
OK,
```

Steps of the Session

1. First, declare SYSIN to be a disk object. The pathname of SYSIN is DC>INTERIOR>NATL_PARK>YOSEMITE. By default, each logical record in this file is variable-length (VAR/PRIME). The maximum length of each logical record is 128 characters.

2. With the MODIFY subcommand, you change the format of this disk file to fixed-length records. Each logical record is now defined to be exactly 128 characters.

3. A comment subcommand (/*) is provided for readability.

4. You declare SYSOUT1 to be a tape object, residing on logical device number 3. It has a density of 1600 bpi. Each logical record contains 141 characters. This tape file is the fourth file on the tape.

5. The next two MODIFY subcommands set the blocking factor to 12 and the record format to FIXED. Three I/O buffers are being used. After use, the tape will be unloaded (dismounted). Because 11*141 equals 1551, an odd number, an extra null character would ordinarily be inserted by Prime's hardware. To circumvent this problem, you specify CHARACTERS=(1) for true odd-length records. Also, this tape file will be transferred to a computer which swaps high- and low-order bytes. Thus, you specify EXCHANGE=(YES).

6. The desired translation is 120 industry-compatible ASCII characters, 10 asterisks, 4 binary characters, 3 spaces, and 4 industry-compatible ASCII characters.

7. You declare SYSOUT2. The translation is exactly the same as SYSOUT1, except for two differences. First of all, you do not specify any I/O buffers. Second, you do specify an external visual identification, NPCA0001. This object is declared as an "extension" to SYSOUT1 so that another tape can be ready if the first tape becomes full.

8. The information in the disk object SYSIN is moved to the destination object SYSOUT1. Should end-of-tape occur on SYSOUT1, messages are sent to the operator requesting that the first tape be dismounted and the second tape (with external visual identification NPCA0001) be mounted. The operator receives these messages because the default communications mode is -OPERATOR on the PRIMOS command line.

9. You declare two new objects, AUDIT_TAPE_1 and AUDIT_TAPE_2, as tape objects on logical devices 0 and 1, respectively. You have specified both FILENO and RECORDNO values for these two objects.

10. The first 274 files on AUDIT_TAPE_1 are copied to AUDIT_TAPE_2.

11. Both AUDIT_TAPE_1 and AUDIT_TAPE_2 are positioned according to the values you specified for each object's FILENO and RECORDNO options. They are positioned absolutely, which in this particular example causes the two tapes to be rewound.

12. A QUIT subcommand ends your MAGNET session and returns you to PRIMOS command level. QUIT causes all dynamic space occupied by objects and I/O buffers to be returned to the free memory pool.

MAGNET MESSAGES

There are three types of messages generated by MAGNET that you may see when you work with the subsystem. They are:

- Severity 1 - This identifies a warning. Whatever operation is in progress at the time you receive the warning continues. This means that you remain in MAGNET and sometimes, depending on what happened, you remain within the subcommand.

- Severity 2 - This identifies an uncorrectable error that occurred within a subcommand. You are returned to MAGNET at subcommand level.
- Severity 3 - This identifies a fatal error. You are returned to PRIMOS.

All MAGNET messages are in this format (nnn is the message identification number):

```
*** Message nnn (sev = n): message
```

The following is a listing of all MAGNET messages. A suggested user response has been provided for each message. In addition, a description has been provided for each message that is not self-explanatory.

***Message 001 (sev=3): Invalid PRIMOS command line.

Suggested User Response: Retype a correct PRIMOS command line.

***Message 002 (sev=2): Error while reading subcommand.

Description: This indicates a possible terminal I/O error.

Suggested User Response: Retype the subcommand line. If the problem persists, contact your System Administrator.

***Message 003 (sev=2): Invalid subcommand (ignored).

Description: You typed an invalid subcommand line.

Suggested User Response: Retype the subcommand line. If the problem persists, contact your System Administrator.

***Message 004 (sev=3): A subcommand has aborted.

Description: A MAGNET subcommand was unable to perform the operation you requested. Normally, you receive a message before this one that indicates the specific type of problem.

Suggested User Response: You must reenter MAGNET and try your MAGNET operation(s) again. If the problem persists, contact your System Administrator.

***Message 005 (sev=1): Warning: additional errors may occur.

***Message 006 (sev=2): Invalid character(s) in object-name.

Suggested User Response: Retype the subcommand line using valid characters in your object-name(s).

***Message 007 (sev=2): An object-name is too long or too short.

Suggested User Response: Retype the subcommand line, making sure that any object-name(s) specified are no longer than 32 characters.

***Message 008 (sev=2): EXTERNAL option within global variable is invalid.

Description: The PRIMOS global variable named in the EXTERNAL option contains the word EXTERNAL. Only one level of EXTERNAL is allowed.

Suggested User Response: Correct your global variable so that the word EXTERNAL does not appear.

***Message 009 (sev=2): EXTERNAL variable name not found.

Description: The PRIMOS global variable named in the EXTERNAL option was not found.

Suggested User Response: Make sure that you have the correct global variable file activated. Next, make sure that the particular global variable exists in the file.

***Message 010 (sev=3): Abnormal halt. Contact your system administrator.

Description: An extremely unusual and uncorrectable condition occurred within MAGNET.

Suggested User Response: Save all pertinent information, such as output from COMOUTPUT, DMSTK, and PM commands. Give all of this information to your System Administrator.

***Message 011 (sev=3): Abnormal halt. Some disk files may be open.

Description: An extremely unusual and uncorrectable condition occurred within MAGNET.

Suggested User Response: Save all pertinent information such as output from COMOUTPUT, and DMSTK commands. Give all of this information to your System Administrator. In addition, check for open disk files.

***Message 012 (sev=2): A tape is hardware write - protected.

Suggested User Response: Put a write-enable ring on your tape(s).

***Message 013 (sev=2): A tape is software write - protected.

Suggested User Response: Set the value of the PROTECT option for your tape(s) to NO.

***Message 014 (sev=2): A disk file is write - protected.

Suggested User Response: If you own the directory where the disk file(s) are located, change the protection attributes to enable write (W) access. If you are a non-owner, contact the owner or the System Administrator to have the protection attributes changed.

***Message 015 (sev=2): A disk file is read - protected.

Suggested User Response: If you own the directory where the disk file(s) are located, change the protection attributes to enable read (R) access. If you are a non-owner, contact the owner or the System Administrator to have the protection attributes changed.

***Message 016 (sev=3): A run-away tape condition exists.

Description: A read, forward-space file, or forward-space record operation is executing indefinitely, which eventually causes the tape to be pulled from the supply reel. This only occurs when your tape is blank (read, forward-space record), or when no more filemarkers are found on the tape (forward-space file).

Suggested User Response: Press the terminal BREAK key immediately. Unassign the tape drive(s) and immediately reassign them. This suddenly causes the tape drive(s) to halt, but you still need to reenter MAGNET to continue your magnetic tape operations.

***Message 017 (sev=2): A tape DISMOUNT (UNLOAD) operation has aborted.

Description: Either you or the operator typed REPLY -TAPE ABORT in response to a DISMOUNT request.

Suggested User Response: Depending on the circumstances, this may be an acceptable response.

***Message 018 (sev=2): A tape MOUNT operation has aborted.

Description: Either you or the operator typed REPLY -TAPE ABORT in response to a MOUNT request.

Suggested User Response: Depending on the circumstances, this may be an acceptable response.

***Message 019 (sev=2): An object was not previously declared.

Suggested User Response: Declare the object(s).

***Message 020 (sev=2): An object was already declared.

Suggested User Response: You cannot declare an object more than once. Use the MODIFY subcommand to change option values.

***Message 021 (sev=2): Memory allocation error: delete some objects.

Description: You have used all available memory for objects.

Suggested User Response: Either delete some objects or reduce the value of the BUFFERS option for declared tape objects.

***Message 022 (sev=2): Error while freeing dynamic memory.

Description: An error occurred when either some buffers (BUFFERS option), or the space occupied by a tape or disk object, was returned to the free storage area.

Suggested User Response: If this occurred during a DELETE operation, give a LIST subcommand to make sure that the object was deleted. If the object was not deleted, reissue the DELETE subcommand. If this error occurred during some other operation, try the operation again. If the problem persists, contact your System Administrator.

***Message 023 (sev=2): Uncorrectable tape read error.

Description: A tape READ operation was unsuccessful after 10 attempts.

Suggested User Response: Check for a correct density setting on the tape drive. If the density setting is correct, your tape is either bad or improperly formatted. There may also be a problem with either the tape drive or controller. If the problem persists, contact your System Administrator.

***Message 024 (sev=2): Uncorrectable tape write error.

Description: A tape WRITE operation was unsuccessful after 10 attempts.

Suggested User Response: Either your tape is bad or you are attempting to write records greater than the maximum I/O window size allowed on your system. There may also be a problem with either the tape drive or controller. If the problem persists, contact your System Administrator.

***Message 025 (sev=2): Uncorrectable tape control error.

Description: A tape control operation was unsuccessful.

Suggested User Response: Check for a correct density setting on the tape drive. If the density setting is correct, your tape is either bad or improperly formatted. There may also be a problem with either the tape drive or controller. If the problem persists, contact your System Administrator.

***Message 026 (sev=2): No room is available in the object list.

Description: You are attempting to declare more than 100 objects (the maximum).

Suggested User Response: Delete any objects you no longer need.

***Message 027 (sev=2): Error while setting up default options.

Suggested User Response: Reissue your DECLARE subcommand. If the problem persists, contact your System Administrator.

***Message 028 (sev=2): Error while parsing the option list.

Suggested User Response: Retype your DECLARE or MODIFY subcommand, correcting any errors that exist.

***Message 029 (sev=1): Option conflict (DENSITY & TRACKS?).

Description: A conflict exists among various options.

Suggested User Response: Check your options for compatibility and change option values with the MODIFY subcommand.

***Message 030 (sev=3): Subsystem aborting- Check for open files.

Description: A subcommand just issued has aborted. Normally, you should have received another message prior to this one indicating the specific error. MAGNET cannot recover from this error, and consequently has returned you to PRIMOS command level.

Suggested User Response: Check for open files. Attempt to correct any reported problems. Reenter MAGNET and issue your subcommand again. If the problem persists, contact your System Administrator.

***Message 031 (sev=2): The COPY subcommand will abort.

Suggested User Response: You should have received a message prior to this one indicating the specific error. Correct the problem and reissue the subcommand.

***Message 032 (sev=2): The DECLARE subcommand will abort.

See Message 031.

***Message 033 (sev=2): The DELETE subcommand will abort.

See Message 031.

***Message 034 (sev=2): Tape drive not assigned.

Suggested User Response: Assign all tape drive(s) at PRIMOS command level with the ASSIGN command. Then, reenter MAGNET and reissue the command(s).

***Message 035 (sev=2): The LOAD subcommand will abort.

See Message 031.

***Message 036 (sev=2): The MODIFY subcommand will abort.

See Message 031.

***Message 037 (sev=2): The MOVE subcommand will abort.

See Message 031.

***Message 038 (sev=2): The POSITION subcommand will abort.

See Message 031.

***Message 039 (sev=3): The QUIT subcommand will abort.

See Message 031.

***Message 040 (sev=2): The READ subcommand will abort.

See Message 031.

***Message 041 (sev=2): The SAVE subcommand will abort.

See Message 031.

***Message 042 (sev=2): The TRANSLATE subcommand will abort.

See Message 031.

***Message 043 (sev=2): The WRITE subcommand will abort.

See Message 031.

***Message 044 (sev=2): Error while analyzing the ACCESS option.

Description: The READ or MODIFY subcommand found an error when it analyzed this option.

Suggested User Response: Retype the subcommand line with this option corrected.

***Message 045 (sev=2): Error while analyzing the BFACTOR option.

See Message 044.

***Message 046 (sev=2): Error while analyzing the BUFFERS option.

See Message 044.

***Message 047 (sev=2): Error while analyzing the BYPASS option.

See Message 044.

***Message 048 (sev=2): Error while analyzing the CHARACTERS option.

See Message 044.

***Message 049 (sev=2): Error while analyzing the CREATE option.

See Message 044.

***Message 050 (sev=2): Error while analyzing the DENSITY option.

See Message 044.

***Message 051 (sev=2): Error while analyzing the DISK option.

See Message 044.

***Message 052 (sev=2): Error while analyzing the EXPIRE option.

See Message 044.

***Message 053 (sev=2): Error while analyzing the EXTERNAL option.

See Message 044.

***Message 054 (sev=2): Error while analyzing the FILEID option.

See Message 044.

***Message 055 (sev=2): Error while analyzing the FORMAT option.

See Message 044.

***Message 056 (sev=2): Message 056 has not been implemented at this release.

***Message 057 (sev=2): Error while analyzing the GENERATION option.

See Message 044.

***Message 058 (sev=2): Error while analyzing the LABELS option.

See Message 044.

***Message 059 (sev=2): Error while analyzing the LEVEL option.

See Message 044.

***Message 060 (sev=2): Error while analyzing the LRECL option.

See Message 044.

***Message 061 (sev=2): Error while analyzing the MAXIO option.

See Message 044.

***Message 062 (sev=2): Error while analyzing the NEXTCHAIN option.

See Message 044.

***Message 063 (sev=2): Error while analyzing the OWNER option.

See Message 044.

***Message 064 (sev=2): Error while analyzing the PARITY option.

See Message 044.

***Message 065 (sev=2): Error while analyzing the POSTACTION option.

See Message 044.

***Message 066 (sev=2): Error while analyzing the PREACTION option.

See Message 044.

***Message 067 (sev=2): Error while analyzing the PREVCHAIN option.

See Message 044.

***Message 068 (sev=2): Error while analyzing the PROTECT option.

See Message 044.

***Message 069 (sev=2): Error while analyzing the TAPE option.

See Message 044.

***Message 070 (sev=2): Error while analyzing the TRACKS option.

See Message 044.

***Message 071 (sev=2): Error while analyzing TRANSLATE token(s).

Description: The list of edit tokens in your TRANSLATE subcommand contains one or more errors. For example, you may have typed A$ instead of A4.

Suggested User Response: Retype the TRANSLATE subcommand with all errors corrected.

***Message 072 (sev=2): Error while analyzing the VERSION option.

See Message 044.

***Message 073 (sev=2): Error while analyzing the VISUAL option.

See Message 044.

***Message 074 (sev=2): Error while analyzing the VOLSER option.

See Message 044.

***Message 075 (sev=2): External names must begin with periods.

Suggested User Response: Retype the SAVE, DECLARE, or MODIFY subcommand with a correct PRIMOS global variable name as the value for the EXTERNAL option.

***Message 076 (sev=2): Invalid subcommand or characters.

Suggested User Response: Retype the subcommand line.

***Message 077 (sev=2): An error occurred while saving into a variable.

Description: An error occurred when the SAVE subcommand tried to save an object's options and values.

Suggested User Response: Make sure that your global variable file is activated and not full. Try the SAVE subcommand again. If the problem persists, contact your System Administrator.

***Message 078 (sev=2): No more room exists to declare this object.

Description: No more room is available in the free storage area to declare this object.

Suggested User Response: Delete some objects or modify the values of the BUFFERS option of any previously declared tape objects.

***Message 079 (sev=3): Subsystem aborting- no files are open.

Description: A subcommand just issued has aborted. Normally, you should have received another error message prior to this one indicating the specific error. MAGNET cannot recover from this error, and consequently has returned you to PRIMOS command level.

Suggested User Response: Attempt to correct any reported problems. Reenter MAGNET and try your subcommand again. If the problem persists, contact your System Administrator.

***Message 080 (sev=1): An invalid command line option will be ignored.

Suggested User Response: No corrective action is necessary. However, if you incorrectly typed -SILENT or -USER you should give the QUIT subcommand and reenter MAGNET with the correct MAGNET command line options.

***Message 081 (sev=1): Execution will continue.

***Message 082 (sev=2): Error while analyzing the FILENO option.

See Message 044.

***Message 083 (sev=2): The RENAME subcommand will abort.

See Message 031.

***Message 084 (sev=1): Circular chaining is not permitted.

Description: The values you specified for the PREVCHAIN and/or NEXTCHAIN options are the same as the object-name you are currently specifying in a DECLARE or MODIFY subcommand.

Suggested User Response: Execution continues, but the NEXTCHAIN and/or PREVCHAIN options are not set. (You have to modify them.)

***Message 085 (sev=2): The wrong tape is mounted.

Suggested User Response: Try the entire subcommand operation again with the correct tape mounted.

***Message 086 (sev=1): The ACCESS option is not initialized.

Suggested User Response: Give this option a value if you meant to. Otherwise, ignore this message.

***Message 087 (sev=1): The BFACTOR option is not initialized.

See Message 086.

***Message 088 (sev=1): The BUFFERS option is not initialized.

See Message 086.

***Message 089 (sev=1): The BYPASS option is not initialized.

See Message 086.

***Message 090 (sev=1): The CHARACTERS option is not initialized.

See Message 086.

***Message 091 (sev=1): The CREATE option is not initialized.

See Message 086.

***Message 092 (sev=1): The DENSITY option is not initialized.

See Message 086.

***Message 093 (sev=1): The DISK option is not initialized.

See Message 086.

***Message 094 (sev=1): The EXPIRE option is not initialized.

See Message 086.

***Message 095 (sev=1): The EXTERNAL option is not initialized.

See Message 086.

***Message 096 (sev=1): The FILEID option is not initialized.

See Message 086.

***Message 097 (sev=1): The FORMAT option is not initialized.

See Message 086.

***Message 098 (sev=1): Message 098 has not been implemented at this release.

***Message 099 (sev=1): The GENERATION option is not initialized.

See Message 086.

***Message 100 (sev=1): The LABELS option is not initialized.

See Message 086.

***Message 101 (sev=1): The LEVEL option is not initialized.

See Message 086.

***Message 102 (sev=1): The LRECL option is not initialized.

See Message 086.

***Message 103 (sev=1): The MAXIO option is not initialized.

See Message 086.

***Message 104 (sev=1): The NEXTCHAIN option is not initialized.

See Message 086.

***Message 105 (sev=1): The OWNER option is not initialized.

See Message 086.

***Message 106 (sev=1): The PARITY option is not initialized.

See Message 086.

***Message 107 (sev=1): The POSTACTION option is not initialized.

See Message 086.

***Message 108 (sev=1): The PREACTION option is not initialized.

See Message 086.

***Message 109 (sev=1): The PREVCHAIN option is not initialized.

See Message 086.

***Message 110 (sev=1): The PROTECT option is not initialized.

See Message 086.

***Message 111 (sev=1): The TAPE option is not initialized.

See Message 086.

***Message 112 (sev=1): The TRACKS option is not initialized.

See Message 086.

***Message 113 (sev=1): TRANSLATION has not been specified.

Suggested User Response: Specify a translation (TRANSLATE subcommand), if you meant to. Otherwise, ignore this message.

***Message 114 (sev=1): The VERSION option is not initialized.

See Message 086.

***Message 115 (sev=1): The VISUAL option is not initialized.

See Message 086.

***Message 116 (sev=1): The VOLSER option is not initialized.

See Message 086.

***Message 117 (sev=1): One or more options will be ignored.

Suggested User Response: Use the DISPLAY subcommand to check that all option values are correct. Then, use the MODIFY subcommand to modify option values, if necessary.

***Message 118 (sev=2): Error while analyzing the RECORDNO option.

See Message 044.

***Message 119 (sev=2): Error while analyzing the OFFSET option.

See Message 044.

***Message 120 (sev=1): The OFFSET option is not initialized.

See Message 086.

***Message 121 (sev=2): DISK, TAPE or EXTERNAL must be the first option.

Suggested User Response: Retype the DECLARE subcommand line with DISK, TAPE, or EXTERNAL as the first option specified.

***Message 122 (sev=2): Disk options are invalid for tape objects.

Suggested User Response: Retype the subcommand line with valid tape options.

***Message 123 (sev=2): Tape options are invalid for disk objects.

Suggested User Response: Retype the subcommand line with valid disk options.

***Message 124 (sev=2): Error while analyzing the AMOUNT option.

See Message 044.

***Message 125 (sev=1): The AMOUNT option is not initialized.

See Message 086.

***Message 126 (sev=2): No objects have been declared.

Suggested User Response: You must first declare all objects before using them.

***Message 127 (sev=1): All options following EXTERNAL are ignored.

Suggested User Response: Since all options that follow the EXTERNAL option in your DECLARE or MODIFY subcommand have been ignored, use the DISPLAY subcommand to check that all option values are correct. If necessary, use the MODIFY subcommand to change option values.

***Message 128 (sev=2): Options must be specified.

Description: You typed a DECLARE or MODIFY subcommand without specifying any options.

Suggested User Response: Retype the subcommand line specifying options.

***Message 129 (sev=2): This object has too many options to save.

Description: The object you specified in the SAVE subcommand contains option values which exceed 1024 characters (the maximum allowed for a PRIMOS global variable).

Suggested User Response: Set a few options back to their default values, and try to issue the SAVE subcommand again.

***Message 130 (sev=2): Global variable file bad or uninitialized.

Description: This message usually indicates that you do not have a global variable file activated.

Suggested User Response: Activate your global variable file at PRIMOS level with the DEFINE_GVAR command. If the problem persists, your global variable file contains bad data and must be recreated.

***Message 131 (sev=2): Global variable file is too small or full.

Description: There is not enough room left in your global variable file to save an object with its option values.

Suggested User Response: To make room in your global variable file, use the PRIMOS command DELETE_VAR to delete some variables.

***Message 132 (sev=2): An invalid translation table was specified.

Suggested User Response: Retype the LOAD subcommand, specifying V, W, X, Y, or Z as the user translation table.

***Message 133 (sev=2): Error while analyzing the LINES option.

See Message 044.

***Message 134 (sev=2): Error while analyzing the TYPE option.

See Message 044.

***Message 135 (sev=1): The LINES option is not initialized.

See Message 086.

***Message 136 (sev=1): The TYPE option is not initialized.

See Message 086.

***Message 137 (sev=2): Error while opening a disk file.

Suggested User Response: Do one of the following:

- If you are trying to read from a disk file, check that the file exists and that the protection and/or access control for this file is correctly set.
- If you are trying to write from a disk file, check for proper protection and/or access control. Check that the disk is not full.

***Message 138 (sev=2): Error while reading from a disk file.

Suggested User Response: Check that the protection and/or access control for this disk file is correctly set.

***Message 139 (sev=2): Data conversion error.

Description: Bad character(s) were detected while reading from a disk file into a user translation table.

Suggested User Response: Check your disk file for valid characters. Binary numbers may contain the characters 0 or 1 only. Octal numbers may contain the characters 0 through 7 only. Decimal numbers may contain the characters 0 through 9 only. Hexadecimal numbers may contain the characters 0 through 9 and A through F only. Check for proper spacing of these numbers. Blanks can be considered part of a numeric field and will cause this error to occur.

***Message 140 (sev=2): Error while closing a disk file.

Suggested User Response: Use the PRIMOS CLOSE command to close the disk file.

***Message 141 (sev=2): The DISPLAY subcommand will abort.

See Message 031.

***Message 142 (sev=2): The LIST subcommand will abort.

See Message 031.

***Message 143 (sev=2): A tape drive is not ready and/or online.

Suggested User Response: Make sure that the tape drive is powered-on, a tape has been mounted, and that the online button has been depressed.

***Message 144 (sev=2): File not found.

Suggested User Response: If you are working with a tape object, try rewinding the tape first. If this is a disk object, use the MODIFY subcommand to change the name of the specified disk file.

***Message 145 (sev=2): Specified record not found.

Description: The POSITON subcommand was unable to position to the record number you specified.

Suggested User Response: This may be an acceptable condition. If not, change the value of the RECORDNO option with the MODIFY subcommand and reissue the subcommand.

***Message 146 (sev=2): A positioning sub-operation has aborted.

Description: An error occurred while a tape was being positioned to its correct file and/or record location.

Suggested User Response: Check that the correct tape is mounted and that the correct density is set. You may try rewinding the tape first before you reissue the subcommand. If the problem persists, it usually indicates that your tape is bad.

***Message 147 (sev=2): One or more objects are not of type TAPE.

Suggested User Response: Reissue the subcommand, making sure that at least one of the objects you specified is a tape object. For the COPY subcommand, all objects must be tape objects. For the POSITION subcommand, the one object you specify must be a tape object. For the READ subcommand, the first object must be a tape object. For the WRITE subcommand, the second object must be a tape object.

***Message 148 (sev=2): One or more objects are not of type DISK.

Suggested User Response: Reissue the subcommand, making sure that at least one of the objects you specified is a disk object. For the READ subcommand, the second object you specify must be a disk object. For the WRITE subcommand, the first object you specify must be a disk object.

***Message 149 (sev=2): Error while analyzing the EXCHANGE option.

See Message 044.

***Message 150 (sev=2): Invalid operation for a DISK object.

Suggested User Response: Reissue the COPY or POSITION subcommand, specifying tape objects only.

***Message 151 (sev=2): Invalid operation for a TAPE object.

Suggested User Response: Reissue the READ or WRITE subcommand with disk and tape objects in the correct order.

***Message 152 (sev=2): An I/O error has occurred in the source object.

Description: This message may indicate one of the following problems:

- A tape drive is not assigned.
- A tape drive is not connected.
- An invalid command was sent to the tape drive.
- The source object in the READ, WRITE, or MOVE subcommand you just issued was left open due to a previous error.
- An uncorrectable error occurred while reading a record from the source object.

Suggested User Response: Make sure you have assigned all tape drives you plan to use. Make sure that any tape drives you have assigned are valid. (If your system contains only one tape drive and it is connected to the first controller, that drive can only be identified by 0, 1, 2, or 3.) Make sure that the density switches on all drives are set correctly. Make sure that all disk files are closed before you invoke MAGNET. If the error persists, it may indicate that your tape contains bad data.

***Message 153 (sev=2): An I/O error has occurred in a destination object.

Description: This message may indicate one of the following problems:

- A tape drive is not assigned.
- A tape drive is not connected.
- An invalid command was sent to the tape drive.
- A destination object in the READ, WRITE, or MOVE subcommand you just issued was left open due to a previous error.
- An uncorrectable error occurred while writing a record to a destination object.

Suggested User Response: Make sure you have assigned all tape drives you plan to use. Make sure that any tape drives you have assigned are valid. (If your system contains only one tape drive and it is connected to the first controller, that drive can only be identified by 0, 1, 2, or 3.) Make sure that the density switches on all drives are set correctly. Make sure that all disk files are closed before you invoke MAGNET. If the error persists, it may indicate that a tape is dirty or creased.

***Message 154 (sev=2): Invalid number of files to copy.

Suggested User Response: Reissue the COPY subcommand specifying a valid number of files to copy.

***Message 155 (sev=1): A labelled tape is being used in a COPY operation.

Suggested User Response: This is a warning only and may be an acceptable message. Note that it is unacceptable for the VOL1 label to be copied to any location other than the first record on a tape.

***Message 156 (sev=2): Multiple objects specify the same tape.

Description: In a COPY or MOVE subcommand, two or more objects specify the same tape.

Suggested User Response: Use the MODIFY subcommand to change the value of the TAPE option. Then, reissue the MOVE or COPY subcommand.

***Message 157 (sev=2): Not enough objects have been specified.

Suggested User Response: Reissue the READ, WRITE, MOVE, or COPY subcommand and specify at least two object-names.

***Message 158 (sev=1): Up to 8 object-names may be specified.

Suggested User Response: This is a warning only. You may specify no more than eight object-names on the MOVE or COPY subcommand lines. Any other object-names are ignored.

***Message 159 (sev=2): Same object-name appears more than once.

Suggested User Response: Reissue the READ, WRITE, MOVE, or COPY subcommand so that object-names on the subcommand line appear only once.

***Message 160 (sev=2): Tape objects require at least one buffer.

Description: You have not specified a BUFFERS option for one or more tape objects. Since the default value for this option is 0, the READ, WRITE, or MOVE subcommand could not be executed.

***Message 161 (sev=2): Error while analyzing the SEQUENCE option.

See Message 044.

***Message 162 (sev=2): Error while analyzing the PRINT option.

See Message 044.

***Message 163 (sev=2): Error while chaining tapes.

Description: You had specified a NEXTCHAIN option value for a tape object. However, when it became necessary to change tapes, the next object could not be found.

Suggested User Response: Use the DECLARE subcommand to declare all objects in a chain. Reissue the READ, WRITE, or MOVE subcommand.

8 The MAGSAV and MAGRST Subsystems

INTRODUCTION

MAGSAV and MAGRST are PRIMOS subsystems that move files from any disk including storage modules to a seven- or nine-track magnetic tape and vice versa. The files are moved logically, and may be SAM, DAM, segment directories, UFDs, MFDs, disks, or partitions. Whenever you specify a directory, the directory and all of its components (the subtree) are transferred. Because of their wide capabilities, MAGSAV and MAGRST can be used by both operators and other system users.

THE MAGSAV SUBSYSTEM

MAGSAV is the PRIMOS disk-to-tape backup subsystem. It allows you to copy files and directories from all PRIMOS-supported disks and disk partitions onto seven- or nine-track magnetic tape.

Running MAGSAV

Before invoking MAGSAV under PRIMOS, you must first assign your magnetic tape drive. (See the ASSIGN command in Chapter 4.) To invoke MAGSAV, the command line format is as follows:

```
    MAGSAV [options]
```

There are several MAGSAV command line options you can specify. After you give options (if any) on the command line, MAGSAV responds with a series of questions. The MAGSAV dialog (questions and appropriate user replies) is discussed following the command line options.

Command Line Options: You may specify one or more options in any order on the MAGSAV command line. These options and their functions are:

-7TRK	Specifies seven-track tape format. The default is nine-track.
-LONG	Specifies a 1024-word record size. The default record size is 512-word records.
-VAR	Increases tape record size to 2048 words and writes variable-length records up to 2048 words. This option overrides -LONG and improves the speed of MAGSAV. -VAR is useful for large files, since it decreases the amount of tape used for record headers and IRGs (inter-record gaps). When you specify this option, MAGSAV prints the record size after the REV stamp of the MAGSAV dialog.
-UPDT	Specifies an update. The DUMPED switch in the UFD entry will be set for files and directories that are saved from disk onto tape. If you do not specify this option, the DUMPED switch is not set.
-INC	Specifies an incremental dump. Only files and directories with a reset (=0) DUMPED switch are saved. If you do not specify -INC, all files and directories are saved.

Caution

Do not use this option when backing up the BATCHQ UFD.

-TTY	Takes the tape unit number from your terminal. All other information is taken from the current input stream. You use this option with CPL files and command input files.

Note

When you modify a file, its DUMPED switch is reset (=0). When you use the -UPDT option, the DUMPED switch is set (=1) for each file or directory that is saved. If you use the -INC option, only files that have a 0 DUMPED switch will be saved. In other words, only files that have been modified since the last time MAGSAV was run will be saved.

The MAGSAV Dialog: After you invoke MAGSAV and specify any options on the command line, the MAGSAV dialog begins. MAGSAV requests information from you in the following order. Appropriate user responses are shown.

Request	Response
TAPE UNIT:	Supply the physical or logical tape unit number (0-7). Note When MAGSAV encounters the physical END OF TAPE (EOT), a message appears on your terminal and a new tape unit is requested. The new unit may be the same as the old unit.
ENTER LOGICAL TAPE NUMBER:	Supply 1 for the first logical tape, 2 for the second, and so on. MAGSAV rewinds the tape, then positions it correctly. If you enter the value 0, MAGSAV assumes that your tape is already positioned correctly.
TAPE NAME:	Supply any name, no more than six characters long.
DATE:	Supply the date in the format MM DD YY. PRIMOS checks the date and rejects it if it is not valid. If you reply to this request with a carriage return (CR), the current date is used. (This is the default.)
REV. NO:	Supply any number.
NAME OR COMMAND:	Supply the name of a file or directory you wish to save, or supply an action command. A pathname identifies the specific file or directory you wish to save on tape. If you specify an MFD, you are identifying an entire disk to be saved. (You must be attached to the specified MFD.) If you specify an asterisk (*) you wish to save your current directory. If MAGSAV encounters an access problem during a save operation, then that file is abandoned, and MAGSAV continues with the next file/UFD (if there is one).

Possible action command responses are:

$A ufd [password] [ldisk] — Changes the home UFD. If you do not specify ldisk, only the local disk is searched for the specified UFD. This is the default. $A does not accept pathnames.

$I [filename] n — Prints, at your terminal, an index of files and directories saved from disk to tape. This is the default. If you specify a filename, the index is written into that file. n indicates the number of levels to be included in the index. (The default is two levels.)

$Q — Terminates the logical tape and returns you to PRIMOS.

$R — Terminates the logical tape, rewinds the physical tape, and returns you to PRIMOS.

$UPDT ON — Turns on update and sets the DUMPED switch for all saved files and directories. This action command is the same as the -UPDT command line option.

UPDT OFF — Turns off update and does not set the DUMPED switch for saved files and directories. This is the default.

$INC ON — Turns on incremented dump and saves only those files and directories with a set DUMPED switch. This action command is the same as the -INC command line option.

$INC OFF — Turns off incremented dump and saves files and directories whether or not their DUMPED switch is set. This is the default.

$VALID ON — Checks each entry name to be sure that it conforms to filename rules.

$VALID OFF — Does not check entry names for conformity to filename rules. This is the default.

Sample MAGSAV Session: The following example illustrates a terminal session during which a disk file, DFILE, was saved on tape. If a carriage return (CR) is given in response to the DATE and REV NO prompts as shown below, the system supplies the current date and zero

Rev number. Note that, as in this example, you can supply a logical device number (ldn) as a response to the TAPE UNIT prompt. You can also supply either a pdn or an ldn (if one has been assigned).

```
OK, AS MT1 -ALIAS MT7
Device MT1 Assigned.
OK, STAT DEV

DEVICE  USRNAM  USRNUM  LDEVICE
MT1     ADLEY    50     MT7

OK, MAGSAV
REV. 18.3
TAPE UNIT (9 TRK): 7
ENTER LOGICAL TAPE NUMBER: 0
TAPE NAME: DATATAP
DATE (MM DD YY): (CR)
REV NO:(CR)
NAME OR COMMAND:  DFILE
NAME OR COMMAND:  $Q
OK,
```

THE MAGRST SUBSYSTEM

MAGRST allows you to restore information from a magnetic tape created by MAGSAV on any Prime-supported disk(s). This information is saved from and readily restored into the PRIMOS file system. All restore operations take place in your home UFD. MAGRST can read tapes of any record size, with fixed- or variable-length records (up to 2048 words), making it compatible with MAGSAV.

Running MAGRST

The MAGRST command line format is as follows:

```
MAGRST [option(s)]
```

Type MAGRST after the PRIMOS prompt (OK,). At this point, there are several command line options you can specify. After you give options (if any) on the command line, MAGRST responds with a series of questions. The MAGRST dialog (questions and appropriate user replies) are discussed following the command line options.

Command Line Options: You may specify one or more options in any order on the MAGRST command line. These options and their functions are:

-7TRK	Specifies seven-track format. The default is nine-track.

-TTY	Takes the tape unit number from your terminal. All other information is taken from the current input stream. You use this option with CPL files and command input files.

The MAGRST Dialog: After you invoke MAGRST and specify any options on the command line, the MAGRST dialog begins. MAGRST requests information from you in the following order. Appropriate user responses are shown.

Request	Response
TAPE UNIT:	Supply a physical or logical device number (0-7). If you do not specify the -7TRK option on the MAGRST command line, the default is nine-track. Note When MAGRST encounters the physical END OF TAPE (EOT), a message appears on your terminal and a new tape unit is requested. The new unit may be the same as the old unit.
ENTER LOGICAL TAPE NUMBER:	Supply a logical tape number from 1 to n (1 for the first logical tape, 2 for the second, and so on) if your tape is divided into several logical units. This positions your tape to the specified logical tape. If you enter the value 0, MAGRST assumes that your tape is already positioned correctly. Note A "runaway" tape condition can occur if there is only one logical tape on the currently mounted reel and you supply a number greater than 1 in response to this request. If this happens, MAGRST searches endlessly for the nonexistent logical tape(s) and is not able to read the EOT (end-of-tape) marker. You must unassign your drive to abort the unsuccessful search. MAGRST does not have to search all logical tapes when it restores sequential ones. After MAGRST returns you to PRIMOS, the tape is not rewound. Instead, it is positioned at the location before the beginning of the next

	logical tape in sequence. For sequential logical tapes, run MAGRST again and supply 0 as the response to the LOGICAL TAPE NO: request. Then, the next logical tape is restored without rewinding and reading through the preceding logical tapes.
NAME:	No user response is necessary. MAGRST displays the name of the logical tape you are currently positioned to. This is the name provided during the MAGSAV dialog.
DATE:	No user response is necessary. MAGRST displays the date that the tape was recorded. This is the date provided during the MAGSAV dialog.
REV. NO.	No user response is necessary. MAGRST displays the number provided during MAGSAV.
REEL NO:	No user response is necessary. MAGRST displays the appropriate tape reel number.
READY TO RESTORE:	Supply one of the following options: YES — Restores the entire tape and returns you to PRIMOS. If MAGRST encounters an access problem during a restore operation, then that file is abandoned and MAGRST continues with the next file/UFD (if there is one). NO — Requests a different tape unit and logical tape. (MAGRST does not restore the previously specified tape.) $I [filename] n -- Prints, at your terminal, an index of all files and directories restored. This is the default. If you specify a filename, the index is written into that file. n indicates the number of levels to be included in the index. (The default is two levels.) NW [filename] n — Prints, at your terminal, a tape index, but files and directories are not restored. If you specify a filename, the index is written into that file. n indicates the number of levels to be included in the index. (The default value is 100.) This option is useful if you wish to determine what is on the tape.

$A UFD [password] [ldisk] [key] — Attaches you to the specified UFD. Pathnames are not allowed. Supply a password, if needed. If you do not specify a number for ldisk, MAGRST searches the local disk for the specified UFD. (The STATUS DISK command gives ldisk numbers.) Supply a numerical value for key to attach to a sub-UFD. (A value of 2, for example, will attach you downward two levels.)

PARTIAL — Restores only certain files and directories. Supply pathnames in response to the TREE NAME: request.

TREE NAME: This request prints only if you supply the PARTIAL option. In response to TREE NAME:, supply the pathname of the file or directory you wish to restore. In the pathname, do not include the name of the UFD that the file or directory was saved from. You may specify multiple pathnames for a partial restore. In this case, the TREE NAME: request is repeated after each restoration until you enter a null line (CR), which signals the end of restoration. For a partial or full restore, files with bad records are omitted. The pathnames of these files are printed, along with an error message.

After each file or directory is restored, the message FILE COMPLETE prints at your terminal. The message RESTORE COMPLETE prints when the end of logical tape is reached.

Sample MAGRST Session: The following example illustrates a terminal session during which a file is restored from tape to disk. The file (DFILE) saved in the previous MAGSAV sample session is also used in this example.

```
OK, MAGRST
REV. 18.3
YOU ARE NOT ATTACHED TO AN MFD
TAPE UNIT (9 TRK):0
ENTER LOGICAL TAPE NUMBER: 1
NAME: DATATAP
DATE(MM DD YY): 09-02-81
REV NO:       0
REEL NO:      1
READY TO RESTORE: PARTIAL
TREE NAME: DFILE
TREE NAME:(CR)
*** STARTING RESTORE ***
*** END LOGICAL TAPE ***
*** RESTORE COMPLETE ***
OK,
```

MAGSAV AND MAGRST MESSAGES

Messages you get from MAGSAV and MAGRST are either informational or error-related. The messages for both subsystems appear here as a group. They are listed alphabetically and identified as being generated by MAGSAV or MAGRST (or both). An explanation is also provided for each message.

-
  ```
  ***** DISK IS FULL *****
  use `delete' command to delete unnecessary files.
  Type `s' to continue.  (MAGRST)
  ```

MAGRST: The partition you are restoring to is full and MAGRST cannot continue until you delete some files.

-
  ```
  *****WARNING — REEL NUMBER NOT IN SEQUENCE
  CONTINUE WITH THIS REEL (Y OR N)?
  ```

MAGRST: You mounted a tape during MAGRST that is not the next reel. You may continue by typing Y, or allow mounting a new tape by typing N.

-
  ```
  BAD ATTACH ON:
  OMITTING TREE-PATH:
  ```

MAGRST: The subsystem was unable to attach to a UFD while restoring it. Subordinate files and UFDs cannot be restored.

-
  ```
  BAD ATTACH:
  NEXT COMMAND MUST BE AN ATTACH
  ```

MAGSAV and MAGRST: An attempt to illegally use $A. MAGSAV/MAGRST will not continue until a correct $A is performed.

- BAD FILETYPE FROM TAPE:

MAGRST: A treename block containing an illegal file/UFD type has been read from tape. The treename and associated data blocks will be ignored.

- BAD LABEL RECORD ID

MAGRST: A new reel with a name different from the previous reel has been loaded and read. MAGRST allows a new tape to be loaded.

- COULDN'T READ FIRST RECORD

MAGRST: The first record on your tape could not be read. Try a new tape.

- COULDN'T READ UFD RECORD,

MAGRST: A UFD record on your tape could not be read.

- DUE TO BAD TAPE ADDITIONAL TAPE MARKS WERE ADDED TO THE END LOGICAL TAPE. DO NOT APPEND OTHER MAGSAV LOGICAL TAPES TO THIS REEL.

MAGSAV: A tape error occurred while a tape mark was being written.

- EMPTY DISK!

MAGSAV: The subsystem is operating on an empty disk. Command dialog will allow you to attach somewhere else.

- ERROR CREATING FILE.

MAGRST: The subsystem could not create the requested file. The file is ignored.

- ERROR OPENING FILE.

MAGRST: The subsystem could not open the requested file. The file is ignored.

- FILE ERROR IN SEGDIR.:

MAGRST: An error was found in SEGDIR. It is ignored.

- FILE IN USE

MAGSAV and MAGRST: A file already in use by other users is either being saved or restored. It is ignored.

- FILE PROTECTED

MAGSAV and MAGRST: A file cannot be saved or restored because it is protected. The file is ignored.

- FILETYPE MISMATCH

MAGRST: The subsystem is attempting to restore an object from tape which has the same name, but a different type, as an object on the disk. The file is ignored.

- INVALID COMMAND

MAGSAV and MAGRST: You typed a command that the subsystem could not recognize.

- INVALID RECID:

MAGRST: An invalid label was encountered in the first tape reel. The subsystem will abort.

- MAGSAV UNABLE TO CONTINUE

MAGSAV: The subsystem cannot find a UFD in a pathname. MAGSAV prints the PRIMOS error message plus this message and exits.

- MT IS OFFLINE OR NOT READY

MAGSAV and MAGRST: The tape drive you requested is offline or not ready. Fix the drive and continue with your save or restore operation.

- NO END LOGICAL TAPE RECORD!!

MAGRST: The subsystem encountered the end of a physical tape, but it did not encounter an end-logical-tape record or an end-of-reel record. It assumes an end-of-logical tape.

- PRE-REV. 12 TAPE

MAGRST: The subsystem is restoring a pre-Rev 12 tape.

- PROB OPENING CURRENT UFD: (error information)
  ATTACH AND TYPE `S'

MAGSAV and MAGRST: The UFD you are currently attached to is causing a problem. Try to attach again.

- PROB READING TAPE LABEL

MAGRST: The subsystem cannot read the tape label. .It may not be a MAGSAV tape. Try another tape.

- PROB WRITING FILE.

MAGRST: The subsystem cannot write the file. It will be omitted.

- RECOVERED MT IO ERRORS

MAGSAV and MAGRST: This specifies the number of magnetic tape errors during a save or restore operation, which were overcome. This is useful for checking the condition of the tapes you are using.

- RESTORE ABORTED ON:

MAGRST: The subsystem aborted due to an unrecoverable tape read error.

- RESTORING PROTECTED UFD:
  OWNER PROT SET TO

MAGRST: The subsystem is restoring a UFD that was previously protected and shows the new protection. The UFD will be restored.

- RUN OUT OF UNITS
  ALTER NUMBER OF UNITS
  treename
  TYPE 'S' TO CONTINUE

MAGSAV: The subsystem ran out of units on which to open UFDs. The specified file will be lost.

- SEGDIR PROBLEM.

MAGSAV and MAGRST: An error occurred in SEGDIR. The file will be omitted.

- TAPE NOT AT LOAD POINT

MAGRST: Your tape is not positioned to the beginning of tape.

- TAPEOF — MT STATUS ERROR

MAGSAV: The subsystem encountered an error while writing a tape marker.

- 'TOO MANY LEVELS' treename

MAGSAV: The subsystem can only save up to 18 levels on PRIMOS, 13 on PRIMOS II. If more levels are attempted, MAGSAV prints this message, ignores the file, returns to the previous level, and saves the files at that level while continuing back up the tree in the same manner.

Note

A PRIMOS system configured for 16 file units/user can save only 13 levels.

- TOO MANY NAMES -- STARTING RESTORE

MAGRST: The subsystem has filled its name table for a partial restore. The restore will still begin.

- UNABLE TO WRITE LABEL. MOUNT NEW TAPE.

MAGSAV: The subsystem cannot write a label on this tape. Obtain another tape.

- UNEXPECTED EOF ENCOUNTERED ASSUMING END OF REEL

MAGRST: The subsystem encountered the end of the tape reel. You can mount a new reel.

- WARNING-SAVING FILE OR SUB-UFD CALLED MFD

MAGSAV: This is only a warning. Your save operation will continue successfully.

- YOU ARE NOT ATTACHED TO AN MFD

MAGSAV: This is only a warning. Your save operation will continue successfully.

9 The PHYSAV and PHYRST Subsystems

INTRODUCTION

PHYSAV and PHYRST are PRIMOS subsystems that create a disk image backup on magnetic tape and restore that image back to disk from tape. Both subsystems support the following hardware:

- Nine-track magnetic tape
- 40MB, 80MB, and 300MB storage module disks
- 32MB, 64MB, and 96MB cartridge module disks

Although PHYSAV and PHYRST are usually used solely by the system operator, both subsystems can be used by all other users as well.

THE PHYSAV SUBSYSTEM

PHYSAV copies the contents of one or more assigned disk partitions to magnetic tape in physical track order. You cannot restore a single file with PHYSAV. No attention is paid to logical file structure. The smallest unit you can restore is a partition (as defined to the PHYSAV subsystem). PHYSAV runs in V-mode under PRIMOS only.

All tracks (of all partitions) on one disk cylinder are written to tape before the disk heads are moved to the next cylinder. This minimizes disk read time. One disk track is written as two magnetic tape blocks of 5212 and 4170 words. After PHYSAV successfully writes the header of each tape section, it informs you with a message identifying the reel, the logical tape, and the corresponding section. A new section begins with 1 for the beginning of a logical tape, and is incremented by 1 each time a new physical magnetic tape reel is started.

Note

You can save a partition using the record availability table (RAT). In this case, all records on a track with records in use by the file system are saved. For more information on using the RAT, see the System Administrator's Guide.

Running PHYSAV

PHYSAV allows you to save assigned partitions on magnetic tape. You can save one or more partitions in one logical tape. A physical reel can contain one or more logical tapes, and a logical tape can span several physical reels. All partitions to be saved in one logical tape must be on the same controller and unit. You can, at PRIMOS level, save one or more partitions that include the command partition. For detailed information, see the System Administrator's Guide.

The PHYSAV command line format is as follows:

```
PHYSAV [-UNMOD]
```

Note

Use the -UNMOD command line option only if you are using one of the following old controllers: wire wrap disk controller boards without ECR 3748, and etched boards without ECRs 3062 and 3342. Using -UNMOD with these boards prevents system hangs due to incorrect recovery from DMX overruns.

The PHYSAV Dialog: After you invoke the subsystem, PHYSAV responds with a series of questions. PHYSAV requests information from you in the following order. Appropriate user responses are shown.

Request	Response
UNIT NO:	Supply the physical tape unit number (0-7), or you may type QUIT. (Reenter the subsystem by typing REN.)
LOGICAL TAPE:	Specify 1 for the first logical tape, 2 for the second, and so on. Note There is no check for the previous existence of logical tape 1; the tape will be written from BOT.
COMMENT	Supply a comment up to 80 characters long.
PHYS.DEV.NO:	Specify the physical device number of the partition to be saved (40460, for example).
USE THE RAT (YES/NO)?	YES — Saves only tracks with records in use by the file system. NO — Saves all records of all tracks. Note If you declare a split partition by its true physical device number, then the paging portion will not be saved (even if you do not specify the RAT option). However, if it is part of several partitions declared as one, the entire partition is saved.
40MB DISK (YES/NO)?	PHYSAV asks this question only if there is not enough information to distinguish a 40MB disk from an 80MB or 300MB disk.
PARAMETERS OK (YES/NO)?	YES — Begins the save operation. NO — Exits from the subsystem. You may reenter by typing REN.

Badspots on a Disk

You can avoid encountering known badspots on a disk (those which have been entered in a partition BADSPT file) by declaring a partition with its own physical device number. You cannot, however, avoid badspots on a partition that is one of a set of partitions declared as a whole. But disk read errors will be reported and the records will not be written. However, as they do not form part of the file system, information will not be lost. Files with records falling on badspots that have appeared after the partition was made will be truncated. For more information on badspots, see the System Administrator's Guide.

Reentering PHYSAV

If you exit from PHYSAV for any reason, you may continue from the exit point by typing S. Restart facilities are also available at different points in the PHYSAV dialog by typing REN. You may do this anytime up to the final NO answer to the question:

```
WRITE NEXT LOG.TAPE (YES/NO)?
```

PHYSAV asks this question after it delivers a message announcing a completed save operation. With a NO response, you exit from the subsystem completely. If you respond with YES, you reenter the subsystem.

If you reenter PHYSAV either this way or with the REN command, the subsystem does not start from the beginning. Rather, it begins from the latest, most convenient point. For example:

- If you have not assigned the magnetic tape unit, you can type QUIT or CNTRL P in response to the UNIT NO: question, assign the magnetic tape unit, then reenter the subsystem at this point by typing REN.
- If you have not assigned one of the partitions to be saved, the subsystem exits. You may then assign the partition and restart the subsystem from this point by typing REN. You will have to reenter the last physical device number, but PHYSAV will have remembered partition numbers you already entered.
- If you exit from the subsystem once the save operation has begun, you may continue from the exit point by typing S, or restart the latest section (logical tape or current reel, whichever is the most recent) by typing REN.

Sample PHYSAV Session

The following example illustrates a terminal session using PHYSAV:

```
OK, PHYSAV
REV 18.3
DATE : SEP 09, 1981    TIME : 08.41
UNIT NO: 0
LOGICAL TAPE: 1
**COMMENT**
This is a save on a Prime machine.
PHYS.DEV.NO: 30462
USE THE RAT (YES/NO)? YES
PHYS.DEV.NO: 40462
USE THE RAT (YES/NO)? YES
PHYS.DEV.NO: (CR)

DISK                         HEAD OFFSET, #HEADS,  SAVING
030462   NEWSYS                  6          2     RECORDS USED
040462   OLDSYS                  8          2     RECORDS USED

PARAMETERS OK (YES/NO)? YES
SAVE COMPLETE
WRITE NEXT LOG.TAPE (YES/NO)? NO
OK,
```

PHYSAV Messages

Messages you receive from PHYSAV are either informational or error-related. All PHYSAV messages are listed here alphabetically. An explanation is provided for each message.

- BAD COMMAND LINE PARAMETER

PHYSAV has encountered a parameter other than -UNMOD. Retype a correct command line.

- BAD HEADER OR TAPE

An error occurred while PHYSAV attempted to position to a logical tape greater than 1. There are three possibilities:

- The tape has been started in an incorrect position (not immediately before a header label).
- The tape was written using a subsystem other than PHYSAV.
- The tape header is unreadable.

You should rewind and try again, or restart the save on a new reel from logical tape 1.

- BAD KEY (rtnnam)

This indicates some type of subsystem error. See your System Administrator.

- BAD RAT OR NOT FILE SYSTEM PARTITION

This is a warning message. If you are attempting to save the partition with the RAT, then PHYSAV will reject the partition entry. Otherwise, the subsystem continues.

- CANNOT ERASE DATA (HW STATUS=xxxxxx)

If a problem is encountered while a block, PHYSAV attempts to rewrite the bad block up to 20 times, erasing three more inches of tape each time while attempting to get over a bad section of the tape. If there are problems with erasing the tape, the latest section (from current reel or logical tape, whichever is most recent) must be rewritten.

For an explanation of bit settings represented in the message by (HW STATUS=xxxxxx), see Table 9-1, Magnetic Tape Controller Hardware Status.

- CANNOT ERASE HEADER (HW STATUS=xxxxxx)

If a problem is encountered while writing a magnetic tape header, PHYSAV attempts to rewrite it up to 20 times, each time erasing three more inches of tape to try to get over a possible bad section on the tape. If there are problems with erasing the tape, the latest section (from current reel or logical tape, whichever is most recent) must be rewritten.

For an explanation of bit settings, see Table 9-1.

Table 9-1
Magnetic Tape Controller Hardware Status

100000	Parity error
040000	Runaway tape
020000	CRC error
010000	LRC error
004000	Low DMX range
002000	Permanent error
001000	Read-after-write error
000400	File marker detected
000200	Ready
000100	On-line
000040	End-of-tape detected
000020	Rewinding
000010	Beginning-of-tape (at load point)
000004	Tape is write-protected
000002	DMX overrun
000001	Rewind complete

- CANNOT ERASE TRAILER (HW STATUS=xxxxxx)

If a problem is encountered while writing trailer labels, PHYSAV attempts to rewrite the labels up to 20 times, each time erasing three more inches of tape to try to get over a possible bad section on the tape. If there are problems with erasing the tape, then the latest section (from current reel or logical tape, whichever is most recent) must be rewritten.

For an explanation of bit settings, see Table 9-1.

- CANNOT WRITE HEADER (HW STATUS=xxxxxx)

PHYSAV cannot write the first header of a logical tape. You have the option of either trying again on the same reel (in which case the subsystem will wind the tape to the correct position) or mounting a new reel.

For an explanation of bit settings, see Table 9-1.

- CANNOT WRITE TRAILER LABELS (HW STATUS=xxxxxx)

If PHYSAV cannot successfully rewrite trailer labels, the latest magnetic tape section (from current reel or logical tape, whichever is most recent) must be rewritten.

For an explanation of bit settings, see Table 9-1.

- DIMENSION ERROR (DSKADD)

The array used to hold the disk RAT must be increased in size (parameter MAXRAT). This should only happen if total disk sizes are increased. See your System Administrator.

- DISK ERROR WHILE ACCESSING RAT

PHYSAV encountered a read error while trying to access the disk RAT. You should use FIXRAT on the partition before you save it.

- DISK READ ERROR, PARTITION aaaaaa, RECORD bbbbbb

PHYSAV encountered an error while reading a disk record. (See also the supervisor terminal listing.) The disk record will not be saved on tape. If this is a known badspot, then this record does not form part of the file system. If it is a new badspot, you should use FIXRAT on the partition after you restore it. The file whose record falls on this badspot will then be truncated.

- DMX OVERRUN

DMX overrun occurred during the magnetic tape write. The magnetic tape controller is at the wrong backplane priority and must be higher than the disk controller.

- END OF INFORMATION
  LAST LOG.TAPE IS x
  WRITE LOG. TAPE x+1 (YES/NO)?

PHYSAV prints this message while positioning to a logical tape. PHYSAV detected the end of information before the requested logical tape was reached. You can either write the next logical tape in sequence, or be prompted again for unit number and logical tape number.

- END OF REEL, MOUNT NEW REEL

PHYSAV detected the end-of-physical-tape marker. The subsystem has correctly written trailer labels and it is ready to continue on the next physical reel.

- HARDWARE ERROR (HW STATUS=xxxxxx)

An unrecoverable problem has occurred. You must first fix the problem before attempting to reenter the subsystem (using REN to restart from the beginning of the latest reel or logical tape).

For an explanation of bit settings, see Table 9-1.

- MORE THAN 10 PARTITIONS

At present, the largest disk that can be added to the system is a 300MB disk, which can contain a maximum of 10 partitions.

- MORE THAN y RECOVERED ERRORS

If PHYSAV detects more than y recovered errors for one physical magnetic tape reel, the tape is considered suspect. By default, y is set to 50. (This can be changed with the parameter MAXERR and program recompilation.) PHYSAV asks you if you wish to rewrite the current section, although the tape is readable and need not be rewritten.

- NO. OF PARTITIONS SO FAR = x

This message occurs when reentering PHYSAV using the REN command. It indicates the number of partitions for which details have already been accepted by PHYSAV.

- NOT AT B-O-T, REWIND (YES/NO)?

PHYSAV expects a magnetic tape to be positioned at B-O-T each time you select a unit. Unless you are sure that the tape is correctly positioned, you should rewind it.

- NOT SAME CONTROLLER OR UNIT

All partitions to be saved in one logical tape must be on the same disk (connected to the same controller and on the same unit).

- NOT STORAGE MODULE TYPE

The only disks that can be saved using PHYSAV are:

  - 40MB, 80MB, and 300MB storage modules
  - 32MB, 64MB, and 96MB cartridge modules

- PARTITION ALREADY SPECIFIED

The partition has already been defined to PHYSAV.

- POSITIONING TO LOG.TAPE x

The message occurs when positioning to a logical tape greater than 1.

- RAT INCONSISTENT WITH PARTITION SIZE

If you are saving a partition using the RAT, it must be a true partition. Therefore, you cannot save multiple partitions declared as one whole partition using the RAT.

- RE-ENTER LAST PHYS.DEV.NO.

This message occurs when reentering PHYSAV (using REN), and indicates:

- You have not assigned a partition to be saved.
- You did not specify any partitions to be saved.
- You answered NO to the question PARAMETERS OK (YES/NO)?, but wish to reenter the subsystem to save a different combination of partitions.
- You have exited from PHYSAV.

PHYSAV remembers any partitions previously entered.

- RE-ENTER UNIT NO.

This message occurs when reentering PHYSAV (using REN). You have exited from PHYSAV before the subsystem accepted the magnetic tape unit number.

- RESTARTING CURRENT REEL

PHYSAV encountered a problem while writing a continuation reel. You can either try the same physical reel again on the same or a different unit, or mount a new reel.

- RESTARTING LOG.TAPE

PHYSAV encountered a problem while writing the first section of a logical tape. You can either restart on the same reel (in which case the subsystem will wind the tape to the correct position) or mount a new reel.

- REWINDING

This message occurs when a selected magnetic tape unit is rewinding.

- REWRITE LOG.TAPE (YES/NO)?
  or REWRITE CURRENT REEL (YES/NO)?

Depending on the nature of the problem PHYSAV has encountered, you respond with either YES or NO:

- YES — restarts the latest section.
- NO — attempts a rewrite; lets the subsystem attempt recovery by backspacing to the magnetic tape block giving trouble.

- TAPE IS WRITE PROTECTED

A write-enable ring is not on the current tape reel.

- THIS IS THE BEG. OF LOG.TAPE n

The present reel starts with a logical tape number greater than the one requested.

- THIS IS A CONTINUATION OF LOG.TAPE x

PHYSAV detected a problem while trying to position to a logical tape greater than 1. The present reel is a continuation of the logical tape requested or of a logical tape with number greater than the one requested.

- UNIT NOT READY OR OFFLINE

You should put the unit on line or, if already on line, try another unit. This message also occurs if a magnetic tape runs off the end of reel (the EOT marker has not been detected or there is no EOT marker) or the tape has broken. In this case, you should rewind the tape and rewrite the latest section (from the start of logical tape or reel, whichever is the most recent) on a new magnetic tape reel or unit.

- UNRECOVERED ERROR (HW STATUS=xxxxxx)

PHYSAV attempts to rewrite a magnetic tape block up to 20 times, erasing three more inches of tape each time in an attempt to get over a local badspot. If PHYSAV is unsuccessful after 20 attempts, this error occurs. You must rewrite the current section.

For an explanation of bit settings, see Table 9-1.

- USE NEW REEL (YES/NO)?

PHYSAV encountered a problem while writing the first section of a logical tape greater than 1. You can either try to restart using the same reel (PHYSAV winds the tape to the correct position) or restart at the start of a new reel.

- WARNING, LOG TAPE ALREADY EXISTS
  OVERWRITE EXISTING LOG. TAPE x (YES/NO)?

This is a warning message. PHYSAV attempted to position to the start of a logical tape. If you reply YES, PHYSAV overwrites the logical tape and cannot access any subsequent logical tapes previously overwritten. If you reply NO, the tape is rewound and you are prompted for a unit number and a logical tape number.

- WARNING - MULTIPLE PARTITIONS
  CANNOT AVOID POSSIBLE BADSPOTS

This is a warning message. If you have declared several partitions as one, then PHYSAV cannot access a BADSPT file on each partition. If there are no badspots, there will be no problem. If there are known badspots, then DISK READ ERRORS will be reported when PHYSAV tries to read bad records. In either case, the saved partition will still be complete.

- WRITING LOG. TAPE 1

PHYSAV is about to write logical tape 1. This message is intended as a warning that the tape is to be overwritten.

THE PHYRST SUBSYSTEM

PHYRST restores one or more partitions saved with PHYSAV to assigned partitions of the same size. It is not necessary to restore all saved partitions in one run. PHYRST runs in R-mode under either PRIMOS II or PRIMOS.

A verify facility is available to verify the readability of a magnetic tape. The tape is read (as for the restore operation), but the information is not written to disk. The verify facility does not verify that all the data has been written to the tape. It only verifies that all tape blocks are readable and in sequence.

Running PHYRST

PHYRST allows you to restore assigned partitions from a magnetic tape written with PHYSAV. Partitions being restored do not have to be on the same controller or unit. However, a saved partition can only be restored to another partition of equal size. You do not have to restore all partitions saved on your tape. You can restore only one partition if you wish. You can also, at PRIMOS level, restore a command partition. For detailed information, see the System Administrator's Guide.

The PHYRST command line format is as follows:

```
PHYRST [-UNMOD]
```

Note

The -UNMOD command line option functions the same for PHYRST as it does for PHYSAV.

The PHYRST Dialog: After you invoke the subsystem, PHYRST responds with a series of questions. PHYRST requests information from you in the following order. Appropriate user responses are shown.

Request	Response
UNIT NO:	Supply the physical tape unit number (0-7), or you may type QUIT. (Reenter the subsystem by typing S 1000.)
LOGICAL TAPE:	Specify 1 for the first logical tape, 2 for the second, and so on.

CORRECT TAPE (YES/NO)?	YES — Continues PHYRST. NO — Prompts you again for a unit number. After the requested logical tape is found, PHYRST prints out the details of the saved information.
RESTORE ALL PARTITIONS TO ORIGINAL POSITIONS (YES/NO)?	YES — Restores all partitions to their original positions. NO — Asks if each partition is to be restored as it is saved.
RESTORE PARTITION xxxxxx (YES/NO)?	NO — Prompts with the next partition saved. YES — Generates the next dialog question.
AS PARTITION:	Supply either a physical device number or type a (CR): pdn — The physical device number to which the partition is to be restored. (CR) — Causes the partition to be restored to its original position.
PARAMETERS OK (YES/NO)?	NO — Exits you from PHYRST. (Type S 1000 to reenter.) YES — Begins the restore operation.

Restoring Partitions Stored with Badspots

If you saved a partition with known badspots (those which have been entered in a partition BADSPT file), the BADSPT file will be saved on tape. If you then restore the partition to a partition with the same logical position, the records remain marked as bad in the BADSPT file. The file system is never able to access these records unless the entire partition is remade.

If, however, you restore the partition to a different logical position and perform a FIXRAT, FIXRAT remembers that entries in the BADSPT file refer to tracks on surfaces with different head offsets. Records are returned to the file system pool and the BADSPT file remains with the original entries.

You can also restore a partition to another partition with different badspots. For additional information on badspots, see the System Administrator's Guide.

Reentering PHYRST

PHYRST runs in R-mode. Consequently, you cannot reenter it with the REN command as you can with PHYSAV. If you exit or QUIT from PHYRST for any reason (a partition is not assigned, for example), you can reenter by typing S 1000. PHYRST restarts from the latest most convenient point. For example, if a partition is not assigned, PHYRST restarts from the beginning of the section that specifies partitions to be restored.

If you type QUIT during a restore operation, PHYRST restarts from the beginning of the latest section (logical tape or current reel, whichever was most recently started). Typing S always continues the subsystem.

Sample PHYRST Session

The following example illustrates a terminal session using PHYRST:

```
OK, PHYRST
REV 18.3
UNIT NO: 0
LOGICAL TAPE: 1

REEL:   1 LOG.TAPE:    1 SECTION:    1

DATE:  SEP 09, 1981 AT 14:22
This is a save on a Prime machine.
PARTITIONS SAVED
030462  NEWSYS
040462  OLDSYS

CORRECT TAPE (YES/NO)? YES
RESTORE OR VERIFY (RE/VE)? RE
RESTORE ALL PARTITIONS TO ORIGINAL POSITIONS (YES/NO)? NO
RESTORE PARTITION 030462 (YES/NO)? NO
RESTORE PARTITION 040462 (YES/NO)? YES
AS PARTITION: 100462

DISK                             TO BE RESTORED AS
040462  OLDSYS                             100462
```

```
PARAMETERS OK (YES/NO)? YES
RESTORE COMPLETE
RESTORE/VERIFY NEXT LOG.TAPE (YES/NO)? NO
OK,
```

PHYRST Messages

Messages you receive from PHYRST are either informational or error-related. All PHYRST messages are listed here alphabetically. An explanation is provided for each message.

- CORRECT TAPE (YES/NO)?

This message is generated after the magnetic tape has been positioned to the correct logical tape. Details of the save will have been produced before you receive this message.

- DISK WRITE ERROR, PARTITION aaaaaa, RECORD bbbbbb

Either PHYRST has encountered a new badspot, or the address of the record to be written was incorrectly read from the tape.

You should check the record address to see if it lies within the partition being restored, type QUIT, and retry the current reel by typing S 1000. Use FIXRAT on the partition when the restore is complete.

- ERROR, EOF READ

PHYRST should never encounter an EOF if it is reading the trailer/header labels first. Information may have been lost here. Restart PHYRST to restore any subsequent reels that belong to the same logical tape. Use FIXRAT on the partitions after the restore is complete. You could also try a different magnetic tape unit for the same reel.

- HARDWARE ERROR (HW STATUS=xxxxxx)

This message indicates an unrecoverable error. It is followed by the message:

```
RESTART CURRENT REEL (YES/NO)?
                or
RESTART LOG.TAPE (YES/NO)?
```

If you reply NO, PHYRST continues by treating the error as recoverable. In other words, it backspaces five blocks, repositions to the block causing the problem, and attempts to read the block up to 20 times.

If this is unsuccessful, PHYRST prints:

```
UNRECOVERED ERROR (HWSTAT=xxxxxx), BLOCK z

RESTART CURRENT REEL (YES/NO)?
                or
RESTART LOG.TAPE (YES/NO)?
```

If you reply NO, PHYRST tries to read up to 20 successive blocks in an attempt to find a good block from which to continue. For an explanation of bit settings, see Table 9-1. If still unsuccessful, PHYRST prints:

```
MORE THAN 20 SUCCESSIVE ERRORS

RESTART CURRENT REEL (YES/NO)?
                or
RESTART LOG.TAPE (YES/NO)?
```

If you reply NO, PHYRST tries again to read up to 20 successive blocks.

- MORE THAN 20 SUCCESSIVE ERRORS

PHYRST detected an unrecovered error, continued, and discovered 20 successive bad blocks. You can either try the next 20 blocks or restart the current section on the same or a different magnetic tape unit.

- NO PARTITIONS TO RESTORE

This is an informational message. You answered NO when PHYRST asked you if it should restore each of the partitions saved on tape.

- NOT SAME REEL

PHYRST has encountered problems, and is consequently attempting to restart the latest section. However, the tape that it found mounted on the new unit is not the original one.

- RESTART LOG.TAPE (YES/NO)?
  or RESTART CURRENT REEL (YES/NO)?

PHYRST has encountered problems with the current section. See the description for HARDWARE ERROR.

- RESTARTING LOG.TAPE
  or RESTARTING CURRENT REEL

One of the following has happened:

- An unrecovered error occurred.
- You requested that PHYRST restart the current section.
- You exited and typed S 1000.

- RESTORE COMPLETE
  RESTORE/VERIFY NEXT LOGICAL TAPE (YES/NO)?

All of the partitions that you requested to be restored from one logical tape have been restored. You can go on to the next logical tape by typing YES. Otherwise, PHYRST exits.

- THIS REEL DOES NOT FOLLOW LAST
  CONTINUE (YES/NO)?

A continuation reel has been mounted during a normal restore operation which either is out of sequence or is not part of the current set of save tapes.

- UNEQUAL NO. OF HEADS

You can only restore a partition to one of equal size.

- UNRECOVERED BLOCK SEQUENCE ERROR, BLOCK x

PHYRST has read a block whose sequence number does not follow that of the previous one. An incomplete dump without trailer labels may exist on the tape, and information written previously has now been read.

- x UNRECOVERED MT ERROR(S)

```
*************
RUN FIXRAT
*************
```

PHYRST has detected one or more unrecovered magnetic tape errors. You should run FIXRAT to see if any vital information has been lost. For example, if the entire partition has been saved, tracks may have been lost that were not in use by the file system.

- VERIFY COMPLETE
  RESTORE/VERIFY NEXT LOGICAL TAPE (YES/NO)?

This is an informational message informing you that the logical tape has been verified.

Appendixes

A
MAGNET Edit Tokens, Character Set Tables, and Translation Tables

TRANSLATION EDIT TOKENS

Edit tokens are part of the MAGNET TRANSLATE subcommand line. Each edit token specifies how you want particular fields of a logical record translated. All MAGNET edit tokens are described in detail in the following paragraphs. (For more information on the TRANSLATE subcommand and edit tokens, see Chapter 7.)

Edit Token A

All characters specified with edit token A translate to or from industry-standard ASCII. When you are reading from a tape, this means translating from industry-standard ASCII to Prime ASCII. When you are writing to a tape, this means translating from Prime ASCII to industry-standard ASCII. For example, if you have 80-character industry-standard ASCII logical records on an input tape, you could specify the following:

```
    (A80)
```

Note that you may also specify this translation as (80(A1)), but this format processes much more slowly.

Edit Token B

This edit token translates seven-track BCDIC code to Prime ASCII, and vice versa. For example, if you have declared a tape object named BANKST consisting of 120-character logical records, your TRANSLATE subcommand line may be:

```
> TRANSLATE BANKST (B120)
>
```

This causes each of the 120 characters in a logical record to be translated from seven-track BCDIC code to Prime ASCII if you are using BANKST as an input object. Again, note that you can also use the form (120(B1)), but it processes much more slowly.

Edit Token C

This is the column position edit token. You use it to move to a field in a logical record. It causes MAGNET's internal pointer to position to a different location. For example, assuming input from a tape, the string:

```
(A80,C10,A10)
```

causes the following to occur:

- Character columns 1 through 80 are translated from industry-standard ASCII to Prime ASCII.
- You are repositioned to column 10.
- Character columns 10 through 19 are translated from industry-standard ASCII to Prime ASCII.

In this example, output from the translation process is 90 characters even though the length of the input logical record is only 80 characters. Note that columns 10 through 19 are translated twice and will appear twice in the output. However, in the following example, less characters will appear on output:

```
(A40,C71,A10)
```

Here, character columns 41 through 70 are not translated and are effectively deleted from the output.

Edit Token D

This is the deletion edit token. It allows you to delete characters. For example, the string:

```
(A40,D30,A10)
```

performs the same task as the previous example. In other words, (A40,C71,A10) is equivalent to (A40,D30,A10).

Edit Token E

This edit token translates EBCDIC code to Prime ASCII, and vice versa. For example, the string:

```
(E100)
```

causes 100 characters to be translated from EBCDIC to Prime ASCII or vice versa.

Edit Token F

This edit token allows you to specify fill characters. These characters may be any symbols except the following:

- Comma ","
- Left and right parentheses "(" and ")"
- Slash "/"
- Apostrophe "'"

Note that alpha characters are always translated to uppercase. For example, the string:

```
(A10,10(F*),C20,A20,10(FX))
```

causes the following to occur:

- 10 characters are translated using A format.
- 10 asterisks (*) are inserted into the output record.
- You are positioned to column 20.
- 20 characters (columns 20 through 39) are translated using A format.

- 10 X's are inserted into the output record.

The output record contains 50 characters. If you specify this edit token by itself, without any fill character, a blank is inserted. The string:

```
(20(F))
```

causes 20 blanks to be inserted.

Edit Tokens G through N

These eight edit tokens specify seven-track packing and unpacking codes. (For more information, see Chapter 1.) Edit tokens G through N and their corresponding packing codes are as follows:

Edit Token	Packing Code
G	2424
H	4242
I	2466
J	4266
K	6246
L	6426
M	6624
N	6642

The repetition factor you specify for these edit tokens is a byte count. Thus, if you wish to pack, on input, three binary numbers in K format, you specify (K9). You use the number 9 because each binary number on input occupies three bytes. On output, each binary number occupies only two bytes or one 16-bit word. Likewise, G and H format binary numbers occupy four bytes on input from a tape and occupy two bytes on output to disk. When you are writing to a tape, binary numbers are expanded from two bytes to four bytes for G and H formats, and to three bytes for I, J, K, L, M, and N formats. When you are determining logical record lengths for tape and disk objects, you must take into account seven-track binary number expansion and compression.

In the following example, 20 binary numbers are translated using H format:

```
(H80)
```

Edit Token O

You use this edit token to specify no translation in your TRANSLATE subcommand line. For example, the subcommand line:

```
> TRANSLATE ACCOUNTS (O100)
>
```

causes each logical record in ACCOUNTS to be transferred through the translation mechanism without any translation occurring (assuming that each logical record is 100 characters long).

Edit Token P

This edit token enforces Prime ASCII input and output. In other words, the high-order bit of each byte is set on, if it is not already. For example, the subcommand line:

```
> TRANSLATE ACCOUNTS (P100)
>
```

causes the high-order bit of each byte of each logical record to be set on.

Edit Tokens Q through U

These edit tokens do not specify anything at this time. They are reserved for future use.

Edit Tokens V through Z

These five edit tokens specify user-defined translation codes. (For more information, see Chapter 7.) You use these edit tokens the same way you use edit tokens A, B, E, O, and P. One character of input is translated according to the table you loaded with a LOAD subcommand and becomes one character of output.

Repetition Factor *

If you wish to specify translation for the remaining characters of a logical record without specifying an absolute number, you may use the repetiton factor *. For example, the subcommand line:

```
> TRANSLATE WUMPUS (A*)
>
```

causes all the characters in each logical record of WUMPUS to be translated according to A format. The subcommand line:

```
> TRANSLATE GRINCH (E29,O10,3(F),E*)
>
```

causes the following to occur:

- 29 characters are translated according to E format.
- 10 characters or bytes are not translated.
- Three blanks are inserted.
- All remaining characters are translated according to E format.

CHARACTER SET TABLES

These character set tables are from the following sources:

- ASCII - American National Standards Institute standard X3.4-1977.
- BCD - Coded Character Sets, History and Development, by C.E. Mackenzie, 1980, Addison-Wesley Publishing, page 68.
- EBCDIC - IBM System/370 Reference Summary, Third Edition, February 1974 (Publication No. GX20-1850-2).

Industry-standard ASCII

^ => Control key depressed

8-Bit Octal Code	Char		Octal Code When in Left Byte	8-Bit Octal Code	Char	Octal Code When in Left Byte
000	NUL	^@	0000	040	Sp	0200
001	SOH	^A	0004	041	!	0204
002	STX	^B	0010	042	"	0210
003	ETX	^C	0014	043	#	0214
004	EOT	^D	0020	044	$	0220
005	ENQ	^E	0024	045	%	0224
006	ACK	^F	0030	046	&	0230
007	BEL	^G	0034	047	'	0234
010	BS	^H	0040	050	(	0240
011	HT	^I	0044	051	)	0244
012	LF	^J	0050	052	*	0250
013	VT	^K	0054	053	+	0254
014	FF	^L	0060	054	,	0260
015	CR	^M	0064	055	-	0264
016	SO	^N	0070	056	.	0270
017	SI	^O	0074	057	/	0274
020	DLE	^P	0100	060	0	0300
021	DC1	^Q	0104	061	1	0304
022	DC2	^R	0110	062	2	0310
023	DC3	^S	0114	063	3	0314
024	DC4	^T	0120	064	4	0320
025	NAK	^U	0124	065	5	0324
026	SYN	^V	0130	066	6	0330
027	ETB	^W	0134	067	7	0334
030	CAN	^X	0140	070	8	0340
031	EM	^Y	0144	071	9	0344
032	SUB	^Z	0150	072	:	0350
033	ESC	^[	0154	073	;	0354
034	FS	^\	0160	074	<	0360
035	GS	^]	0164	075	=	0364
036	RS	^^	0170	076	>	0370
037	US	^_	0174	077	?	0374

8-Bit Octal Code	Char	Octal Code When in Left Byte	8-Bit Octal Code	Char	Octal Code When in Left Byte
100	@	0400	140	`	0600
101	A	0404	141	a	0604
102	B	0410	142	b	0610
103	C	0414	143	c	0614
104	D	0420	144	d	0620
105	E	0424	145	e	0624
106	F	0430	146	f	0630
107	G	0434	147	g	0634
110	H	0440	150	h	0640
111	I	0444	151	i	0644
112	J	0450	152	j	0650
113	K	0454	153	k	0654
114	L	0460	154	l	0660
115	M	0464	155	m	0664
116	N	0470	156	n	0670
117	O	0474	157	o	0674
120	P	0500	160	p	0700
121	Q	0504	161	q	0704
122	R	0510	162	r	0710
123	S	0504	163	s	0714
124	T	0520	164	t	0720
125	U	0524	165	u	0724
126	V	0530	166	v	0730
127	W	0534	167	w	0734
130	X	0540	170	x	0740
131	Y	0544	171	y	0744
132	Z	0550	172	z	0750
133	[	0554	173	{	0754
334	\	0560	174	|	0760
135	]	0564	175	}	0764
136	^	0570	176	~	0770
137	_	0574	177	DEL	0774

Prime ASCII

F => Valid filename character
R => Reserved command line character
^ => Control key depressed

8-Bit Octal Code	Char		Octal Code When in Left Byte	8-Bit Octal Code	Char	Octal Code When in Left Byte	
200	NUL	^@	1000	240	Sp	1200	
201	SOH	^A	1004	241	!	1204	R
202	STX	^B	1010	242	"	1210	R
203	ETX	^C	1014	243	#	1214	F
204	EOT	^D	1020	244	$	1220	F
205	ENQ	^E	1024	245	%	1224	R
206	ACK	^F	1030	246	&	1230	FR
207	BEL	^G	1034	247	'	1234	R
210	BS	^H	1040	250	(	1240	R
211	HT	^I	1044	251	)	1244	R
212	LF	^J	1050	252	*	1250	F
213	VT	^K	1054	253	+	1254	
214	FF	^L	1060	254	,	1260	R
215	CR	^M	1064	255	-	1264	FR
216	SO	^N	1070	256	.	1270	F
217	SI	^O	1074	257	/	1274	F
220	DLE	^P	1100	260	0	1300	F
221	DC1	^Q	1104	261	1	1304	F
222	DC2	^R	1110	262	2	1310	F
223	DC3	^S	1114	263	3	1314	F
224	DC4	^T	1120	264	4	1320	F
225	NAK	^U	1124	265	5	1324	F
226	SYN	^V	1130	266	6	1330	F
227	ETB	^W	1134	267	7	1334	F
230	CAN	^X	1140	270	8	1340	F
231	EM	^Y	1144	271	9	1344	F
232	SUB	^Z	1150	272	:	1350	R
233	ESC	^[	1154	273	;	1354	R
234	FS	^\	1160	274	<	1360	
235	GS	^]	1164	275	=	1364	R
236	RS	^^	1170	276	>	1370	
237	US	^_	1174	277	?	1374	

8-Bit Octal Code	Char	Octal Code When in Left Byte	8-Bit Octal Code	Char	Octal Code When in Left Byte
300	@	1400 R	340	`	1600 R
301	A	1404 F	341	a	1604
302	B	1410 F	342	b	1610
303	C	1414 F	343	c	1614
304	D	1420 F	344	d	1620
305	E	1424 F	345	e	1624
306	F	1430 F	346	f	1630
307	G	1434 F	347	g	1634
310	H	1440 F	350	h	1640
311	I	1444 F	351	i	1644
312	J	1450 F	352	j	1650
313	K	1454 F	353	k	1654
314	L	1460 F	354	l	1660
315	M	1464 F	355	m	1664
316	N	1470 F	356	n	1670
317	O	1474 F	357	o	1674
320	P	1500 F	360	p	1700
321	Q	1504 F	361	q	1704
322	R	1510 F	362	r	1710
323	S	1514 F	363	s	1714
324	T	1520 F	364	t	1720
325	U	1524 F	365	u	1724
326	V	1530 F	366	v	1730
327	W	1534 F	367	w	1734
330	X	1540 F	370	x	1740
331	Y	1544 F	371	y	1744
332	Z	1550 F	372	z	1750
333	[	1554 R	373	{	1754 R
334	\	1560 R	374	|	1760
335	]	1564 R	375	}	1764 R
336	^	1570 R	376	~	1770 R
337	_	1574 F	377	DEL	1774

EBCDIC

Decimal	Octal	Hex.	Char.	Decimal	Octal	Hex.	Char.
000	000	00	NUL	032	040	20	DS
001	001	01	SOH	033	041	21	SOS
002	002	02	STX	034	042	22	FS
003	003	03	ETX	035	043	23	
004	004	04	PF	036	044	24	BYP
005	005	05	HT	037	045	25	LF
006	006	06	LC	038	046	26	ETB
007	007	07	DEL	039	047	27	ESC
008	010	08		040	050	28	
009	011	09		041	051	29	
010	012	0A	SMM	042	052	2A	SM
011	013	0B	VT	043	053	2B	CU2
012	014	0C	FF	044	054	2C	
013	015	0D	CR	045	055	2D	ENQ
014	016	0E	SO	046	056	2E	ACK
015	017	0F	SI	047	057	2F	BEL
016	020	10	DLE	048	060	30	
017	021	11	DC1	049	061	31	
018	022	12	DC2	050	062	32	SYN
019	023	13	TM	051	063	33	
020	024	14	RES	052	064	34	PN
021	025	15	NL	053	065	35	RS
022	026	16	BS	054	066	36	UC
023	027	17	IL	055	067	37	EOT
024	030	18	CAN	056	070	38	
025	031	19	EM	057	071	39	
026	032	1A	CC	058	072	3A	
027	033	1B	CU1	059	073	3B	CU3
028	034	1C	IFS	060	074	3C	DC4
029	035	1D	IGS	061	075	3D	NAK
030	036	1E	IRS	062	076	3E	
031	037	1F	IUS	063	077	3F	SUB

Decimal	Octal	Hex.	Char.	Decimal	Octal	Hex.	Char.
064	100	40	SP	096	140	60	-
065	101	41		097	141	61	/
066	102	42		098	142	62	
067	103	43		099	143	63	
068	104	44		100	144	64	
069	105	45		101	145	65	
070	106	46		102	146	66	
071	107	47		103	147	67	
072	110	48		104	150	68	
073	111	49		105	151	69	
074	112	4A		106	152	6A	
075	113	4B	.	107	153	6B	,
076	114	4C	<	108	154	6C	%
077	115	4D	(	109	155	6D	_
078	116	4E	+	110	156	6E	>
079	117	4F		111	157	6F	?
080	120	50		112	160	70	
081	121	51		113	161	71	
082	122	52		114	162	72	
083	123	53		115	163	73	
084	124	54		116	164	74	
085	125	55		117	165	75	
086	126	56		118	166	76	
087	127	57		119	167	77	
088	130	58		120	170	78	
089	131	59		121	171	79	`
090	132	5A	!	122	172	7A	:
091	133	5B	$	123	173	7B	#
092	134	5C	*	124	174	7C	@
093	135	5D	)	125	175	7D	'
094	136	5E	;	126	176	7E	=
095	137	5F		127	177	7F	"

Decimal	Octal	Hex.	Char.	Decimal	Octal	Hex.	Char.
128	200	80		160	240	A0	
129	201	81	a	161	241	A1	~
130	202	82	b	162	242	A2	s
131	203	83	c	163	243	A3	t
132	204	84	d	164	244	A4	u
133	205	85	e	165	245	A5	v
134	206	86	f	166	246	A6	w
135	207	87	g	167	247	A7	x
136	210	88	h	168	250	A8	y
137	211	89	i	169	251	A9	z
138	212	8A		170	252	AA	
139	213	8B		171	253	AB	
140	214	8C		172	254	AC	
141	215	8D		173	255	AD	
142	216	8E		174	256	AE	
143	217	8F		175	257	AF	
144	220	90		176	260	B0	
145	221	91	j	177	261	B1	
146	222	92	k	178	262	B2	
147	223	93	l	179	263	B3	
148	224	94	m	180	264	B4	
149	225	95	n	181	265	B5	
150	226	96	o	182	266	B6	
151	227	97	p	183	267	B7	
152	230	98	q	184	270	B8	
153	231	99	r	185	271	B9	
154	232	9A		186	272	BA	
155	233	9B		187	273	BB	
156	234	9C		188	274	BC	
157	235	9D		189	275	BD	
158	236	9E		190	276	BE	
159	237	9F		191	277	BF	

Decimal	Octal	Hex.	Char.	Decimal	Octal	Hex.	Char.
192	300	C0	{	224	340	E0	\
193	301	C1	A	225	341	E1	
194	302	C2	B	226	342	E2	S
195	303	C3	C	227	343	E3	T
196	304	C4	D	228	344	E4	U
197	305	C5	E	229	345	E5	V
198	306	C6	F	230	346	E6	W
199	307	C7	G	231	347	E7	X
200	310	C8	H	232	350	E8	Y
201	311	C9	I	233	351	E9	Z
202	312	CA		234	352	EA	
203	313	CB		235	353	EB	
204	314	CC		236	354	EC	
205	315	CD		237	355	ED	
206	316	CE		238	356	EE	
207	317	CF		239	357	EF	
208	320	D0	}	240	360	F0	0
209	321	D1	J	241	361	F1	1
210	322	D2	K	242	362	F2	2
211	323	D3	L	243	363	F3	3
212	324	D4	M	244	364	F4	4
213	325	D5	N	245	365	F5	5
214	326	D6	O	246	366	F6	6
215	327	D7	P	247	367	F7	7
216	330	D8	Q	248	370	F8	8
217	331	D9	R	249	371	F9	9
218	332	DA		250	392	FA	
219	333	DB		251	373	FB	
220	334	DC		252	374	FC	
221	335	DD		253	375	FD	
222	336	DE		254	376	FE	
223	337	DF		255	377	FF	

BCD

Decimal	Octal	Hex.	Char.
000	000	00	blank
001	001	01	1
002	002	02	2
003	003	03	3
004	004	04	4
005	005	05	5
006	006	06	6
007	007	07	7
008	010	08	8
009	011	09	9
010	012	0A	0
011	013	0B	#
012	014	0C	@
013	015	0D	/
014	016	0E	S
015	017	0F	T
016	020	10	U
017	021	11	V
018	022	12	W
019	023	13	X
020	024	14	Y
021	025	15	Z
022	026	16	,
023	027	17	)
024	030	18	-
025	031	19	J
026	032	1A	K
027	033	1B	L
028	034	1C	M
029	035	1D	N
030	036	1E	O
031	037	1F	P
032	040	20	Q
033	041	21	R
034	042	22	$
035	043	23	*
036	044	24	&
037	045	25	A
038	046	26	B
039	047	27	C
040	050	28	D
041	051	29	E
042	052	2A	F
043	053	2B	G
044	054	2C	H
045	055	2D	I
046	056	2E	.
047	057	2F	(

TRANSLATION TABLES

BCD to Prime ASCII

Binary	Octal	Decimal	Hex.	Characters
10100000 10110001	120261	41361	A191	space 1
10110010 10110011	131263	45747	B2B3	2 3
10110100 10110101	132265	46261	B4B5	4 5
10110110 10110111	133267	46775	B6B7	6 7
10111000 10111001	134271	47289	B8B9	8 9
10110000 10100011	130243	45219	60A3	0 #
11000000 10101111	140257	49327	C0AF	@ /
11010011 11010100	151724	54228	D3D4	S T
11010101 11010110	152726	54742	D5D6	U V
11010111 11011000	153730	55256	D7D8	W X
11011001 11011010	154732	55770	D9DA	Y Z
10101100 10101001	126251	44201	ACA9	,)
10101101 11001010	126712	44490	ADCA	- J
11001011 11001100	145714	52172	CBCC	K L
11001101 11001110	146716	52686	CDCE	M N
11001111 11010000	147720	53200	CFD0	O P
11010001 11010010	150722	53714	D1D2	Q R
10100100 10101010	122252	42154	A4AA	$ *
10101011 11000001	125701	43969	ABC1	+ A
11000010 11000011	141303	49859	C2C3	B C
11000100 11000101	142305	50373	C4C5	D E
11000110 11000111	143307	50887	C6C7	F G
11001000 11001001	144311	51401	C8C9	H I
10101110 10101000	127250	44712	AEA8	. (
10101001 10100101	124645	43429	A9A5	) %
00000000 00000000	000000	00000	0000	(No lowercase or control characters. This line repeats 104 additional times.)

Prime ASCII to BCD

Binary	Octal	Decimal	Hex.	Characters
00000000 00000000	000000	00000	0000	(No lowercase or control characters. This line repeats 15 additional times.)
00000000 00001100	000014	00012	000C	spc !
00001100 00001011	006013	03083	0C0D	" #
00101011 00001100	025414	11020	260C	$ %
00001100 00001100	006014	03084	0C0C	? '
00111100 00011100	016074	07228	1C3C	( )
00101100 00110000	026060	11312	2C30	* +
00011011 00100000	015440	06944	1B20	, -
00111011 00010001	035421	15121	3B11	. /
00001010 00000001	005001	02561	0A01	0 1
00000010 00000011	001003	00515	0203	2 3
00000100 00000101	002005	01029	0405	4 5
00000110 00000111	003007	01543	0607	6 7
00001000 00001001	004011	02057	0809	8 9
00001100 00001100	006014	03084	0C0C	: ;
00001100 00001100	006014	03084	0C0C	< =
00001100 00001100	006014	03084	0C0C	> ?
00000000 00110001	000061	00049	0031	@ A
00110010 00110011	031063	12851	3233	B C
00110100 00110101	032065	13365	3435	D E
00110110 00110111	033067	13879	3637	F G
00111000 00111001	034071	14393	3839	H I
00100001 00100010	020442	08482	2122	J K
00100011 00100100	021444	08996	2324	L M
00100101 00100110	022446	09510	2526	N O
00100111 00101000	023450	10024	2728	P Q
00101001 00010010	024422	10514	2912	R S
00010011 00010100	011424	04884	1314	T U
00010101 00010110	012426	05398	1516	V W
00010111 00011000	013430	05912	1718	X Y
00011001 00001100	014414	06412	190C	Z]
00001100 00001100	006014	03084	0C0C	\ [
00001100 00001100	006014	03084	0C0C	^ _
00000000 00000000	000000	00000	0000	(No lowercase or control characters. This line repeats 31 additional times.)

00000000	00000000	000000	00000	0000	(No lowercase or control characters. This line repeats 15 additional times.)	
00000000	00001100	000014	00012	000C	spc	!
00001100	00001011	006013	03083	0C0D	"	#
00101011	00001100	025414	11020	260C	$	%
00001100	00001100	006014	03084	0C0C	?	'
00111100	00011100	016074	07228	1C3C	(	)
00101100	00110000	026060	11312	2C30	*	+
00011011	00100000	015440	06944	1B20	,	-
00111011	00010001	035421	15121	3B11	.	/
00001010	00000001	005001	02561	0A01	0	1
00000010	00000011	001003	00515	0203	2	3
00000100	00000101	002005	01029	0405	4	5
00000110	00000111	003007	01543	0607	6	7
00001000	00001001	004011	02057	0809	8	9
00001100	00001100	006014	03084	0C0C	:	;
00001100	00001100	006014	03084	0C0C	<	=
00001100	00001100	006014	03084	0C0C	>	?
00000000	00110001	000061	00049	0031	@	A
00110010	00110011	031063	12851	3233	B	C
00110100	00110101	032065	13365	3435	D	E
00110110	00110111	033067	13879	3637	F	G
00111000	00111001	034071	14393	3839	H	I
00100001	00100010	020442	08482	2122	J	K
00100011	00100100	021444	08996	2324	L	M
00100101	00100110	022446	09510	2526	N	O
00100111	00101000	023450	10024	2728	P	Q
00101001	00010010	024422	10514	2912	R	S
00010011	00010100	011424	04884	1314	T	U
00010101	00010110	012426	05398	1516	V	W
00010111	00011000	013430	05912	1718	X	Y
00011001	00001100	014414	06412	190C	Z ·	]
00001100	00001100	006014	03084	0C0C	\	[
00001100	00001100	006014	03084	0C0C	^	_
00000000	00000000	000000	00000	0000	(No lowercase or control characters. This line repeats 31 additional times.)	

Prime ASCII to EBCDIC

Binary	Octal	Decimal	Hex.	Characters	
00000000 00000001	000001	00001	0001	NUL	SOH
00000010 00000011	001003	00515	0203	STX	ETX
00110111 00101101	033455	14125	372D	EOT	ENQ
00101110 00101111	027057	11823	2E2F	ACK	BEL
00010110 00000101	013005	05637	1605	BS	HT
00100101 00001011	022413	09483	250B	LF	VT
00001100 00001101	006015	03085	0C0D	FF	CR
00001110 00001111	007017	03599	0E0F	SO	SI
00010000 00010001	010021	04113	1011	DLE	DC1
00010010 00010011	011023	04627	1213	DC2	TM
00111100 00111101	036075	15421	3C3D	DC4	NAK
00110010 00100110	031046	12838	3226	SYN	ETB
00011000 00011001	014031	06169	1819	CAN	EM
00111111 00100111	037447	16167	3F27	SUB	ESC
00100010 01111100	021174	08828	227C	FS	@
00110101 01111100	032574	13692	357C	RS	@
01000000 01011010	040132	16474	405A	SP	!
01111111 01111011	077573	32635	7F7B	"	#
01011011 01101100	055554	23404	5B6C	$	%
01010000 01111101	050175	20605	507D	&	'
01001101 01011101	046535	19805	4D5D	(	)
01011100 01001110	056116	23630	5C4E	*	+
01101011 01100000	065540	27488	6B60	,	-
01001011 01100001	045541	19297	4B61	.	/
11110000 11110001	170361	61681	F0F1	0	1
11110010 11110011	171363	62195	F2F3	2	3
11110100 11110101	172365	62709	F4F5	4	5
11110110 11110111	173367	63223	F6F7	6	7
11111000 11111001	174371	63737	F8F9	8	9
01111010 01011110	075136	31326	7A5E	:	;
01001100 01111110	046176	19582	4C7E	<	=
01101110 01101111	067157	28271	6E6F	>	?
01111100 11000001	076301	31937	7CC1	@	A
11000010 11000011	141303	49859	C2C3	B	C
11000100 11000101	142305	50373	C4C5	D	E
11000110 11000111	143307	50887	C6C7	F	G
11001000 11001001	144311	51401	C8C9	H	I
11010001 11010010	150722	53714	D1D2	J	K
11010011 11010100	151724	54228	D3D4	L	M
11010101 11010110	152726	54742	D5D6	N	O
11010111 11011000	153730	55256	D7D8	P	Q
11011001 11100010	154742	55778	D9E2	R	S
11100011 11100100	161744	58340	E3E4	T	U
11100101 11100110	162746	58854	E5E6	V	W
11100111 11101000	163750	59368	E7E8	X	Y
11101001 01111100	164574	59772	E97C	Z	@
11100000 01111100	160174	57468	E07C	\	@
01011111 01101101	057555	24429	5F6D		_
01111001 10000001	074601	31105	7981	`	a

10000010	10000011	101203	33411	8283	b	c
10000100	10000101	102205	33925	8485	d	e
10000110	10000111	103207	34439	8687	f	g
10001000	10001001	104211	34953	8889	h	i
10010001	10010010	110622	37266	9192	j	k
10010011	10010100	111624	37780	9394	l	m
10010101	10010110	112626	38294	9596	n	o
10010111	10011000	113630	38808	9798	p	q
10011001	10100010	114642	39330	99A2	r	s
10100011	10100100	121644	41892	A3A4	t	u
10100101	10100110	122646	42406	A5A6	v	w
10100111	10101000	123650	42920	A7A8	x	y
10101001	11000000	124700	43456	A9C0	z	{
01101010	11010000	065320	27344	6AD0	|	}
10100001	00000111	120407	41223	A107	~	del
00000000	00000001	000001	00001	0001	NUL	SOH
00000010	00000011	001003	00515	0203	STX	ETX
00110111	00101101	033455	14125	372D	EOT	ENQ
00101110	00101111	027057	11823	2E2F	ACK	BEL
00010110	00000101	013005	05637	1605	BS	HT
00100101	00001011	022413	09483	250B	LF	VT
00001100	00001101	006015	03085	0C0D	FF	CR
00001110	00001111	007017	03599	0E0F	SO	SI
00010000	00010001	010021	04113	1011	DLE	DC1
00010010	00010011	011023	04627	1213	DC2	TM
00111100	00111101	036075	15421	3C3D	DC4	NAK
00110010	00100110	031046	12838	3226	SYN	ETB
00011000	00011001	014031	06169	1819	CAN	EM
00111111	00100111	037447	16167	3F27	SUB	ESC
00100010	01111100	021174	08828	227C	FS	@
00110101	01111100	032574	13692	357C	RS	@
01000000	01011010	040132	16474	405A	SP	!
01111111	01111011	077573	32635	7F7B	"	#
01011011	01101100	055554	23404	5B6C	$	%
01010000	01111101	050175	20605	507D	&	'
01001101	01011101	046535	19805	4D5D	(	)
01011100	01001110	056116	23630	5C4E	*	+
01101011	01100000	065540	27488	6B60	,	-
01001011	01100001	045541	19297	4B61	.	/
11110000	11110001	170361	61681	F0F1	0	1
11110010	11110011	171363	62195	F2F3	2	3
11110100	11110101	172365	62709	F4F5	4	5
11110110	11110111	173367	63223	F6F7	6	7
11111000	11111001	174371	63737	F8F9	8	9
01111010	01011110	075136	31326	7A5E	:	;
01001100	01111110	046176	19582	4C7E	<	=
01101110	01101111	067157	28271	6E6F	>	?
01111100	11000001	076301	31937	7CC1	@	A
11000010	11000011	141303	49859	C2C3	B	C
11000100	11000101	142305	50373	C4C5	D	E
11000110	11000111	143307	50887	C6C7	F	G
11001000	11001001	144311	51401	C8C9	H	I
11010001	11010010	150722	53714	D1D2	J	K

11010011	11010100	151724	54228	D3D4	L	M
11010101	11010110	152726	54742	D5D6	N	O
11010111	11011000	153730	55256	D7D8	P	Q
11011001	11100010	154742	55778	D9E2	R	S
11100011	11100100	161744	58340	E3E4	T	U
11100101	11100110	162746	58854	E5E6	V	W
11100111	11101000	163750	59368	E7E8	X	Y
11101001	01111100	164574	59772	E97C	Z	@
11100000	01111100	160174	57468	E07C	\	@
01011111	01101101	057555	24429	5F6D		_
01111001	10000001	074601	31105	7981	`	a
10000010	10000011	101203	33411	8283	b	c
10000100	10000101	102205	33925	8485	d	e
10000110	10000111	103207	34439	8687	f	g
10001000	10001001	104211	34953	8889	h	i
10010001	10010010	110622	37266	9192	j	k
10010011	10010100	111624	37780	9394	l	m
10010101	10010110	112626	38294	9596	n	o
10010111	10011000	113630	38808	9798	p	q
10011001	10100010	114642	39330	99A2	r	s
10100011	10100100	121644	41892	A3A4	t	u
10100101	10100110	122646	42406	A5A6	v	w
10100111	10101000	123650	42920	A7A8	x	y
10101001	11000000	124700	43456	A9C0	z	{
01101010	11010000	065320	27344	6AD0	|	}
10100001	00000111	120407	41223	A107	~	del

EBCDIC to Prime ASCII

Binary	Octal	Decimal	Hex.	Characters	
10100000 10000001	120201	41089	A081	space	SOH
10000010 10000011	101203	33411	8283	STX	ETX
10111111 10001001	137611	49033	BF89	?	HT
10111111 10010000	137620	49040	BF90	?	RCP
10111111 10111111	137677	49087	BFBF	?	?
10111111 10001011	137613	49035	BF8B	?	VT
10001100 10001101	106215	35981	8C8D	FF	CR
10001110 10001111	107217	36495	8E8F	RRS	BRS
10010000 10010001	110221	37009	9091	RCP	RHT
10010010 10010011	111223	37523	9293	HLF	RVT
10111111 10111111	137677	49087	BFBF	?	?
10001000 10111111	104277	35007	88BF	BS	?
10011000 10011001	114231	39065	9899	CAN	EM
10111111 10111111	137677	49087	BFBF	?	?
10111111 10111111	137677	49087	BFBF	?	?
10111111 10111111	137677	49087	BFBF	?	?
10111111 10111111	137677	49087	BFBF	?	?
10011100 10111111	116277	40127	9CBF	FS	?
10111111 10001010	137612	49034	BF8A	?	NL
10010111 10011011	113633	38811	979B	ETB	ESC
10111111 10111111	137677	49087	BFBF	?	?
10111111 10111111	137677	49087	BFBF	?	?
10111111 10000101	137605	49029	BF85	?	ENQ
10000110 10000111	103207	34439	8687	ACK	BEL
10000000 10000000	100200	32896	8080	NUL	NUL
10010110 10000000	113200	38528	9680	SYN	NUL
10111111 10011110	137636	49054	BF9E	?	RS
10111111 10000100	137604	49028	BF84	?	EOT
10111111 10111111	137677	49087	BFBF	?	?
10111111 10111111	137677	49087	BFBF	?	?
10010100 10010101	112225	38037	9495	HLR	NAK
10000000 10011010	100232	32922	809A	NUL	SUB
10100000 10111111	120277	41151	A0BF	SP	?
10111111 10111111	137677	49087	BFBF	?	?
10111111 10111111	137677	49087	BFBF	?	?
10111111 10111111	137677	49087	BFBF	?	?
10111111 10111111	137677	49087	BFBF	?	?
10111111 10101110	137656	49070	BFAE	?	.
10111100 10101000	136250	48296	BCA8	<	(
10101011 11111100	125774	44028	ABFC	+	|
10100110 10111111	123277	42687	A6BF	&	?
10111111 10111111	137677	49087	BFBF	?	?
10111111 10111111	137677	49087	BFBF	?	?
10111111 10111111	137677	49087	BFBF	?	?
10111111 10111111	137677	49087	BFBF	?	?
10100001 10100100	120644	41380	A1A4	!	$
10101010 10101001	125251	43689	AAA9	*	)
10111011 10111111	135677	48063	BBBF	;	?
10101101 10101111	126657	44463	ADAF	-	/

10111111	10111111	137677	49087	BFBF	?	?
10111111	10111111	137677	49087	BFBF	?	?
10111111	10111111	137677	49087	BFBF	?	?
10111111	10111111	137677	49087	BFBF	?	?
10111111	10101100	137654	49068	BFAC	?	,
10100101	11011111	122737	42463	A5DF	%	_
10111110	10111111	137277	48831	BEBF	>	'?'
10111111	10111111	137677	49087	BFBF	?	?
10111111	10111111	137677	49087	BFBF	?	?
10111111	10111111	137677	49087	BFBF	?	?
10111111	10111111	137677	49087	BFBF	?	?
10111111	11100000	137740	49120	BFE0	?	SP
10111010	10100011	135243	47779	BAA3	:	#
11000000	10100111	140247	49319	C0A7	@	'
10111101	10100010	136642	48546	BDA2	=	"
10111111	11100001	137741	49121	BFE1	?	a
11100010	11100011	161343	58083	E2E3	b	c
11100100	11100101	162345	58597	E4E5	d	e
11100110	11100111	163347	59111	E6E7	f	g
11101000	11101001	164351	59625	E8E9	h	i
10111111	10111111	137677	49087	BFBF	?	?
10111111	10111111	137677	49087	BFBF	?	?
10111111	10111111	137677	49087	BFBF	?	?
10111111	11101010	137752	49130	BFEA	?	j
11101011	11101100	165754	60396	EBEC	k	l
11101101	11101110	166756	60910	EDEE	m	n
11101111	11110000	167760	61424	EFF0	o	p
11110001	11110010	170762	61938	F1F2	q	r
10111111	10111111	137677	49087	BFBF	?	?
10111111	10111111	137677	49087	BFBF	?	?
10111111	10111111	137677	49087	BFBF	?	?
10111111	11111110	137776	49150	BFFE	?	~
11110011	11110100	171764	62452	F3F4	s	t
11110101	11110110	172766	62966	F5F6	u	v
11110111	11111000	173770	63480	F7F8	w	x
11111001	11111010	174772	63994	F9FA	y	z
10111111	10111111	137677	49087	BFBF	?	?
10111111	10111111	137677	49087	BFBF	?	?
10111111	10111111	137677	49087	BFBF	?	?
10111111	10111111	137677	49087	BFBF	?	?
10111111	10111111	137677	49087	BFBF	?	?
10111111	10111111	137677	49087	BFBF	?	?
10111111	10111111	137677	49087	BFBF	?	?
10111111	10111111	137677	49087	BFBF	?	?
10111111	10111111	137677	49087	BFBF	?	?
10111111	10111111	137677	49087	BFBF	?	?
10111111	10111111	137677	49087	BFBF	?	?
11111011	11000001	175701	64449	FBC1	{	A
11000010	11000011	141303	49859	C2C3	B	C
11000100	11000101	142305	50373	C4C5	D	E
11000110	11000111	143307	50887	C6C7	F	G
11001000	11001001	144311	51401	C8C9	H	I
10111111	10111111	137677	49087	BFBF	?	?

10111111	10111111	137677	49087	BFBF	?	?
10111111	10111111	137677	49087	BFBF	?	?
11111011	11001010	175712	64458	FBCA	{	J
11001011	11001100	145714	52172	CBCC	K	L
11001101	11001110	146716	52686	CDCE	M	N
11001111	11010000	147720	53200	CFD0	O	P
11010001	11010010	150722	53714	D1D2	Q	R
10111111	10111111	137677	49087	BFBF	?	?
10111111	10111111	137677	49087	BFBF	?	?
10111111	10111111	137677	49087	BFBF	?	?
11011100	10111111	156277	56511	DCBF	\	?
11010011	11010100	151724	54228	D3D4	S	T
11010101	11010110	152726	54742	D5D6	U	V
11010111	11011000	153730	55256	D7D8	W	X
11011001	11011010	154732	55770	D9DA	Y	Z
10111111	10111111	137677	49087	BFBF	?	?
10111111	10111111	137677	49087	BFBF	?	?
10111111	10111111	137677	49087	BFBF	?	?
10110000	10110001	130261	45233	B0B1	0	1
10110010	10110011	131263	45747	B2B3	2	3
10110100	10110101	132265	46261	B4B5	4	5
10110110	10110111	133267	46775	B6B7	6	7
10111000	10111001	134271	47289	B8B9	8	9
11111100	10111111	176277	64703	FCBF	|	?
10111111	10111111	137677	49087	BFBF	?	?
10111111	10111111	137677	49087	BFBF	?	?

Industry-standard ASCII to Prime ASCII

Binary		Octal	Decimal	Hex.	Characters	
10000000	10000001	100201	32897	8081	NUL	SOH
10000010	10000011	101203	33411	8283	STX	ETX
10000100	10000101	102205	33925	8485	EOT	ENQ
10000110	10000111	103207	34439	8687	ACK	BEL
10001000	10001001	104211	34953	8889	BS	HT
10001010	10001011	105213	35467	8A8B	LF	VT
10001100	10001101	106215	35981	8C8D	FF	CR
10001110	10001111	107217	36495	8E8F	SO	SI
10010000	10010001	110221	37009	9091	DLE	DC1
10010010	10010011	111223	37523	9293	DC2	DC3
10010100	10010101	112225	38037	9495	DC4	NAK
10010110	10010111	113227	38551	9697	SYN	ETB
10011000	10011001	114231	39065	9899	CAN	EM
10011010	10011011	115233	39579	9A9B	SUB	ESC
10011100	10011101	116235	40093	9C9D	FS	GS
10011110	10011111	117237	40607	9E9F	RS	US
10100000	10100001	120241	41121	A0A1	sp	!
10100010	10100011	121243	41635	A2A3	"	#
10100100	10100101	122245	42149	A4A5	$	%
10100110	10100111	123247	42663	A6A7	&	'
10101000	10101001	124251	43177	A8A9	(	)
10101010	10101011	125253	43691	AAAB	*	+
10101100	10101101	126255	44205	ACAD	,	-
10101110	10101111	127257	44719	AEAF	.	/
10110000	10110001	130261	45233	B0B1	0	1
10110010	10110011	131263	45747	B2B3	2	3
10110100	10110101	132265	46261	B4B5	4	5
10110110	10110111	133267	46775	B6B7	6	7
10111000	10111001	134271	47289	B8B9	8	9
10111010	10111011	135273	47803	BABB	:	;
10111100	10111101	136275	48317	BCBD	<	=
10111110	10111111	137277	48831	BEBF	>	?
11000000	11000001	140301	49345	C0C1	@	A
11000010	11000011	141303	49859	C2C3	B	C
11000100	11000101	142305	50373	C4C5	D	E
11000110	11000111	143307	50887	C6C7	F	G
11001000	11001001	144311	51401	C8C9	H	I
11001010	11001011	145313	51915	CACB	J	K
11001100	11001101	146315	52429	CCCD	L	M
11001110	11001111	147317	52943	CECF	N	O
11010000	11010001	150321	53457	D0D1	P	Q
11010010	11010011	151323	53971	D2D3	R	S
11010100	11010101	152325	54485	D4D5	T	U
11010110	11010111	153327	54999	D6D7	V	W
11011000	11011001	154331	55513	D8D9	X	Y
11011010	11011011	155333	56027	DADB	Z	[
11011100	11011101	156335	56541	DCDD	\	]
11011110	11011111	157337	57055	DEDF	^	_
11100000	11100001	160341	57569	E0E1	`	a

11100010	11100011	161343	58083	E2E3	b	c
11100100	11100101	162345	58597	E4E5	d	e
11100110	11100111	163347	59111	E6E7	f	g
11101000	11101001	164351	59625	E8E9	h	i
11101010	11101011	165353	60139	EAEB	j	k
11101100	11101101	166355	60653	ECED	l	m
11101110	11101111	167357	61167	EEEF	n	o
11110000	11110001	170361	61681	F0F1	p	q
11110010	11110011	171363	62195	F2F3	r	s
11110100	11110101	172365	62709	F4F5	t	u
11110110	11110111	173367	63223	F6F7	v	w
11111000	11111001	174371	63737	F8F9	x	y
11111010	11111011	175373	64251	FAFB	z	{
11111100	11111101	176375	64765	FCFD	|	}
11111110	11111111	177377	65279	FEFF	~	DEL
10000000	10000001	100201	32897	8081	NUL	SOH
10000010	10000011	101203	33411	8283	STX	ETX
10000100	10000101	102205	33925	8485	EOT	ENQ
10000110	10000111	103207	34439	8687	ACK	BEL
10001000	10001001	104211	34953	8889	BS	HT
10001010	10001011	105213	35467	8A8B	LF	VT
10001100	10001101	106215	35981	8C8D	FF	CR
10001110	10001111	107217	36495	8E8F	SO	SI
10010000	10010001	110221	37009	9091	DLE	DC1
10010010	10010011	111223	37523	9293	DC2	DC3
10010100	10010101	112225	38037	9495	DC4	NAK
10010110	10010111	113227	38551	9697	SYN	ETB
10011000	10011001	114231	39065	9899	CAN	EM
10011010	10011011	115233	39579	9A9B	SUB	ESC
10011100	10011101	116235	40093	9C9D	FS	GS
10011110	10011111	117237	40607	9E9F	RS	US
10100000	10100001	120241	41121	A0A1	sp	!
10100010	10100011	121243	41635	A2A3	"	#
10100100	10100101	122245	42149	A4A5	$	%
10100110	10100111	123247	42663	A6A7	&	'
10101000	10101001	124251	43177	A8A9	(	)
10101010	10101011	125253	43691	AAAB	*	+
10101100	10101101	126255	44205	ACAD	,	-
10101110	10101111	127257	44719	AEAF	.	/
10110000	10110001	130261	45233	B0B1	0	1
10110010	10110011	131263	45747	B2B3	2	3
10110100	10110101	132265	46261	B4B5	4	5
10110110	10110111	133267	46775	B6B7	6	7
10111000	10111001	134271	47289	B8B9	8	9
10111010	10111011	135273	47803	BABB	:	;
10111100	10111101	136275	48317	BCBD	<	=
10111110	10111111	137277	48831	BEBF	>	?
11000000	11000001	140301	49345	C0C1	@	A
11000010	11000011	141303	49859	C2C3	B	C
11000100	11000101	142305	50373	C4C5	D	E
11000110	11000111	143307	50887	C6C7	F	G
11001000	11001001	144311	51401	C8C9	H	I
11001010	11001011	145313	51915	CACB	J	K

11001100	11001101	146315	52429	CCCD	L	M
11001110	11001111	147317	52943	CECF	N	O
11010000	11010001	150321	53457	D0D1	P	Q
11010010	11010011	151323	53971	D2D3	R	S
11010100	11010101	152325	54485	D4D5	T	U
11010110	11010111	153327	54999	D6D7	V	W
11011000	11011001	154331	55513	D8D9	X	Y
11011010	11011011	155333	56027	DADB	Z	[
11011100	11011101	156335	56541	DCDD	\	]
11011110	11011111	157337	57055	DEDF	^	_
11100000	11100001	160341	57569	E0E1	`	a
11100010	11100011	161343	58083	E2E3	b	c
11100100	11100101	162345	58597	E4E5	d	e
11100110	11100111	163347	59111	E6E7	f	g
11101000	11101001	164351	59625	E8E9	h	i
11101010	11101011	165353	60139	EAEB	j	k
11101100	11101101	166355	60653	ECED	l	m
11101110	11101111	167357	61167	EEEF	n	o
11110000	11110001	170361	61681	F0F1	p	q
11110010	11110011	171363	62195	F2F3	r	s
11110100	11110101	172365	62709	F4F5	t	u
11110110	11110111	173367	63223	F6F7	v	w
11111000	11111001	174371	63737	F8F9	x	y
11111010	11111011	175373	64251	FAFB	z	{
11111100	11111101	176375	64765	FCFD	|	}
11111110	11111111	177377	65279	FEFF	~	DEL

Prime ASCII to Industry-standard ASCII

Binary		Octal	Decimal	Hex.	Characters	
00000000	00000001	000001	00001	0001	NUL	SOH
00000010	00000011	001003	00515	0203	STX	ETX
00000100	00000101	002005	01029	0405	EOT	ENQ
00000110	00000111	003007	01543	0607	ACK	BEL
00001000	00001001	004011	02057	0809	BS	HT
00001010	00001011	005013	02571	0A0B	LF	VT
00001100	00001101	006015	03085	0C0D	FF	CR
00001110	00001111	007017	03599	0E0F	SO	SI
00010000	00010001	010021	04113	1011	DLE	DC1
00010010	00010011	011023	04627	1213	DC2	DC3
00010100	00010101	012025	05141	1415	DC4	NAK
00010110	00010111	013027	05655	1617	SYN	ETB
00011000	00011001	014031	06169	1819	CAN	EM
00011010	00011011	015033	06683	1A1B	SUB	ESC
00011100	00011101	016035	07197	1C1D	FS	GS
00011110	00011111	017037	07711	1E1F	RS	US
00100000	00100001	020041	08225	2021	sp	!
00100010	00100011	021043	08739	2223	"	#
00100100	00100101	022045	09253	2425	$	%
00100110	00100111	023047	09767	2627	&	'
00101000	00101001	024051	10281	2829	(	)
00101010	00101011	025053	10795	2A2B	*	+
00101100	00101101	026055	11309	2C2D	,	-
00101110	00101111	027057	11823	2E2F	.	/
00110000	00110001	030061	12337	3031	0	1
00110010	00110011	031063	12851	3233	2	3
00110100	00110101	032065	13365	3435	4	5
00110110	00110111	033067	13879	3637	6	7
00111000	00111001	034071	14393	3839	8	9
00111010	00111011	035073	14907	3A3B	:	;
00111100	00111101	036075	15421	3C3D	<	=
00111110	00111111	037077	15935	3E3F	>	?
01000000	01000001	040101	16449	4041	@	A
01000010	01000011	041103	16963	4243	B	C
01000100	01000101	042105	17477	4445	D	E
01000110	01000111	043107	17991	4647	F	G
01001000	01001001	044111	18505	4849	H	I
01001010	01001011	045113	19019	4A4B	J	K
01001100	01001101	046115	19533	4C4D	L	M
01001110	01001111	047117	20047	4E4F	N	O
01010000	01010001	050121	20561	5051	P	Q
01010010	01010011	051123	21075	5253	R	S
01010100	01010101	052125	21589	5455	T	U
01010110	01010111	053127	22103	5657	V	W
01011000	01011001	054131	22617	5859	X	Y
01011010	01011011	055133	23131	5A5B	Z	[
01011100	01011101	056135	23645	5C5D	\	]
01011110	01011111	057137	24159	5E5F	^	_
01100000	01100001	060141	24673	6061	`	a

01100010	01100011	061143	25187	6263	b	c
01100100	01100101	062145	25701	6465	d	e
01100110	01100111	063147	26215	6667	f	g
01101000	01101001	064151	26729	6869	h	i
01101010	01101011	065153	27243	6A6B	j	k
01101100	01101101	066155	27757	6C6D	l	m
01101110	01101111	067157	28271	6E6F	n	o
01110000	01110001	070161	28785	7071	p	q
01110010	01110011	071163	29299	7273	r	s
01110100	01110101	072165	29813	7475	t	u
01110110	01110111	073167	30327	7677	v	w
01111000	01111001	074171	30841	7879	x	y
01111010	01111011	075173	31355	7A7B	z	{
01111100	01111101	076175	31869	7C7D	|	}
01111110	01111111	077177	32383	7E7F	~	DEL
00000000	00000001	000001	00001	0001	NUL	SOH
00000010	00000011	001003	00515	0203	STX	ETX
00000100	00000101	002005	01029	0405	EOT	ENQ
00000110	00000111	003007	01543	0607	ACK	BEL
00001000	00001001	004011	02057	0809	BS	HT
00001010	00001011	005013	02571	0A0B	LF	VT
00001100	00001101	006015	03085	0C0D	FF	CR
00001110	00001111	007017	03599	0E0F	SO	SI
00010000	00010001	010021	04113	1011	DLE	DC1
00010010	00010011	011023	04627	1213	DC2	DC3
00010100	00010101	012025	05141	1415	DC4	NAK
00010110	00010111	013027	05655	1617	SYN	ETB
00011000	00011001	014031	06169	1819	CAN	EM
00011010	00011011	015033	06683	1A1B	SUB	ESC
00011100	00011101	016035	07197	1C1D	FS	GS
00011110	00011111	017037	07711	1E1F	RS	US
00100000	00100001	020041	08225	2021	sp	!
00100010	00100011	021043	08739	2223	"	#
00100100	00100101	022045	09253	2425	$	%
00100110	00100111	023047	09767	2627	&	'
00101000	00101001	024051	10281	2829	(	)
00101010	00101011	025053	10795	2A2B	*	+
00101100	00101101	026055	11309	2C2D	,	-
00101110	00101111	027057	11823	2E2F	.	/
00110000	00110001	030061	12337	3031	0	1
00110010	00110011	031063	12851	3233	2	3
00110100	00110101	032065	13365	3435	4	5
00110110	00110111	033067	13879	3637	6	7
00111000	00111001	034071	14393	3839	8	9
00111010	00111011	035073	14907	3A3B	:	;
00111100	00111101	036075	15421	3C3D	<	=
00111110	00111111	037077	15935	3E3F	>	?
01000000	01000001	040101	16449	4041	@	A
01000010	01000011	041103	16963	4243	B	C
01000100	01000101	042105	17477	4445	D	E
01000110	01000111	043107	17991	4647	F	G
01001000	01001001	044111	18505	4849	H	I
01001010	01001011	045113	19019	4A4B	J	K

01001100	01001101	046115	19533	4C4D	L	M
01001110	01001111	047117	20047	4E4F	N	O
01010000	01010001	050121	20561	5051	P	Q
01010010	01010011	051123	21075	5253	R	S
01010100	01010101	052125	21589	5455	T	U
01010110	01010111	053127	22103	5657	V	W
01011000	01011001	054131	22617	5859	X	Y
01011010	01011011	055133	23131	5A5B	Z	[
01011100	01011101	056135	23645	5C5D	\	]
01011110	01011111	057137	24159	5E5F	^	_
01100000	01100001	060141	24673	6061	`	a
01100010	01100011	061143	25187	6263	b	c
01100100	01100101	062145	25701	6465	d	e
01100110	01100111	063147	26215	6667	f	g
01101000	01101001	064151	26729	6869	h	i
01101010	01101011	065153	27243	6A6B	j	k
01101100	01101101	066155	27757	6C6D	l	m
01101110	01101111	067157	28271	6E6F	n	o
01110000	01110001	070161	28785	7071	p	q
01110010	01110011	071163	29299	7273	r	s
01110100	01110101	072165	29813	7475	t	u
01110110	01110111	073167	30327	7677	v	w
01111000	01111001	074171	30841	7879	x	y
01111010	01111011	075173	31355	7A7B	z	{
01111100	01111101	076175	31869	7C7D	|	}
01111110	01111111	077177	32383	7E7F	~	DEL

B
Command Line Options and Batch Jobs

While invoking MAGNET, there are several command line options you can specify. The -SILENT option disables the printing of Severity 1 messages. (See Chapter 7 for detailed information on MAGNET messages.) The -USER and -OPERATOR (or -OPR) options specify where MOUNT and DISMOUNT messages are printed. -USER causes these messages to be printed on your terminal. -OPERATOR (or -OPR) causes them to be printed on the operator's console.

All MOUNT and DISMOUNT messages require a reply from either you or the operator. At the operator's console, the reply takes the form of the REPLY command. (See Chapter 5.) The operator, however, should not use the following form of the REPLY command:

```
    REPLY -usrnum -TAPE pdn
```

Acceptable forms for the operator are:

```
    REPLY -usrnum -TAPE GO

            or

    REPLY -usrnum -TAPE ABORT
```

After you receive a MOUNT or DISMOUNT message and a prompt (>) at your terminal, your reply should be one of the following:

```
> REPLY -TAPE GO

            or

> REPLY -TAPE ABORT
```

Any other response causes the MOUNT or DISMOUNT message to be repeated.

The default communication mode is -OPERATOR. Directing messages and replies to the operator's console rather than your input command stream allows you to build prepackaged magnetic tape jobs that you can run under BATCH. These jobs can be run at any time, with tape mount and dismount instructions submitted to an operator prior to execution.

C MAGNET Additional Features and Notes

PRE-REV 18.3 MAGNET

The pre-Rev 18.3 functionality of MAGNET is still supported. The four subcommands that still accept interactive dialog are:

Subcommand	Function
POSITION	Positions the tape to a file/record.
READ	Reads a file from tape to disk.
WRITE	Writes a file to tape from disk.
COPY	Copies a file from one tape to another.

The POSITION Subcommand

The POSITION subcommand positions the magnetic tape to a specific file and record number. POSITION has two modes, absolute and relative. An absolute position causes the tape to be rewound before any spacing occurs. A relative position allows the tape to be moved forward or backward from the current position.

When you give the POSITION subcommand, MAGNET requests a magnetic tape unit number. The response is an integer in the range 0 to 7, optionally followed by a /7 or /9 to indicate a seven- or nine-track transport. The default is nine-track. MAGNET then asks whether this is an absolute or relative position. Finally, the file and record numbers are requested. If you specify an absolute position, the file and record numbers are absolute (starting with 1), and must be positive. In relative mode, the file number represents the number of files to space forward or backward and may be positive or negative. The record number is the number of records to space forward or backward and must be greater than or equal to 0. An example of a typical POSITION operation is as follows:

```
    OK, MAGNET
    [MAGNET, rev. 18.3]
    > POSITION
    MTU # = 0
    Relative or absolute? ABSOLUTE
    File # = 23
    Record # = 271
    >
```

The READ Subcommand

The READ subcommand allows a file to be read from a magnetic tape and written to disk. It also provides optional unblocking and EBCDIC or BCD translation.

The READ subcommand requests the unit number, which you must enter in the format described under the POSITION subcommand. You then enter the magnetic tape file number, which must be a positive integer. If this number is greater than 0, the tape is rewound and positioned to the specified file number. If the file number is 0, the tape is not rewound.

The next series of questions in the READ subcommand dialog relates to the format of the data on the magnetic tape file to be read. The logical record size is the number of bytes in each logical record. The blocking factor is the number of logical records (line images) contained in one physical tape record. MAGNET then asks you for the type of translation, and you provide one of the following responses:

ASCII	Specifies no translation between tape and disk. MAGNET writes the data to the disk file in Prime ASCII.
EBCDIC	Instructs MAGNET to translate the data on the tape from EBCDIC to Prime ASCII before writing it to the disk file.
BCD	Instructs MAGNET to translate the data from BCD (six-bit) to Prime ASCII before writing it to the disk file. This option is only useful for seven-track tapes.

BINARY	Instructs MAGNET to write the data verbatim to a binary disk file. The record size is the specified logical record size. Packing or unpacking only occurs if you are using a seven-track tape.

If you specify either EBCDIC or BCD translation, MAGNET asks if the entire record is to be translated or only selected fields. This permits bypassing translation of binary fields in EBCDIC records (output, for example, by COBOL). For partial record translation, MAGNET requests starting and ending column numbers of the data to be translated. This input is terminated with a null line (CR only). Finally, MAGNET requests the disk output filename. The following is an example of a typical READ procedure:

```
OK, MAGNET
[MAGNET, rev. 18.3]
> READ
MTU # = 0
File # = 1
Logical record length = 80
Blocking factor = 10
ASCII, EBCDIC, BCD or BINARY? EBCDIC
Full or partial record translation? PARTIAL
Enter pairs of starting/ending column numbers, one pair per line.
Separate each column number with a comma. Terminate entry with a
null line (carriage-return) only.
10,25
35,50
Disk file: FILE.INPUT
>
```

The WRITE Subcommand

The WRITE subcommand is similar to the READ subcommand, except that the file on disk is written to the magnetic tape. WRITE also provides facilities for blocking and character translation.

The WRITE subcommand first asks for the magnetic tape unit and file numbers, and then for information on how the data format is to appear on the tape. As for the READ subcommand, you must specify the logical record size and the blocking factor. You must also provide the type of translation, if any. You specify one of the following options:

ASCII	Specifies no translation.
EBCDIC	Specifies EBCDIC translation onto tape.
BCD	Specifies BCD translation onto tape.
BINARY	Specifies a binary file.

If you specify EBCDIC or BCD translation, MAGNET asks whether the entire record or just certain fields are to be translated. (See the READ subcommand description for more information.) Finally, MAGNET requests the name of the input disk file. An example of a typical WRITE procedure is as follows:

```
OK, MAGNET
[MAGNET, rev. 18.3]
> READ
MTU # = 0
File # = 1
Logical record length = 60
Blocking factor = 20
ASCII, EBCDIC, BCD or BINARY? ASCII
Disk file: SUEFILE
>
```

The COPY Subcommand

The COPY subcommand copies a file (or files) from one magnetic tape to another. No character translation is provided. The magnetic tape transports may be either seven- or nine-track.

MAGNET requests the FROM and TO tape units, starting file numbers, and the number of files to copy. To copy an entire tape, you must enter a large number. (EOT is detected and causes the copy operation to halt.) The following is an example of a typical COPY procedure:

```
OK, MAGNET
[MAGNET, rev. 18.3]
> COPY

*** "FROM" tape information: ***

MTU # = 2
Starting file # = 5

*** "TO" tape information: ***

MTU # = 6
Starting file # = 9

 # of files to copy = 27
Print record sizes? NO
>
```

Notes on Pre-Rev 18.3 Features

- All interactive commands now return to MAGNET subcommand level, not to PRIMOS level as in the pre-Rev 18.3 software. You must use the QUIT subcommand to return to PRIMOS command level.
- Pre-Rev 18.3 MAGNET printed messages indicating completion of READ, WRITE, COPY, or POSITION operations. New MAGNET does not print these messages.
- When EOT is detected, the new MAGNET REPLY mechanism is now used. In addition, all MOUNT and DISMOUNT messages are directed to the operator's console unless you explicitly use the -USER command line option.
- MAGNET now retries all I/O operations up to 10 times in case of error. Prior to Rev 18.3, MAGNET indicated that an error had occurred and whether or not it was a recoverable error. New MAGNET does not supply you with this information. In addition, all error messages have changed. (See Chapter 7.)
- It is recommended that you modify any existing command or CPL files to take advantage of the new MAGNET subcommands. The old (pre-Rev 18.3) subcommand forms are provided for compatibility only.

ADDITIONAL NOTES

- User labels are not yet supported.
- In the new forms of the READ, WRITE, and MOVE subcommands, the default translation code used is Prime ASCII to industry standard ASCII for output, and industry standard ASCII to Prime ASCII for input. The default translation is Prime ASCII to Prime ASCII for both input and output when you use the old forms of the READ and WRITE subcommands.
- Physical record lengths (LRECL * BFACTOR) should be at least 20 characters.
- ACCESS codes are recognized, but are not processed or verified.
- Pre-Rev 18.3 MAGNET did not write a filemarker after EOT was detected. This functionality was nonstandard. New MAGNET does write this filemarker on output, and expects to read it on input. Therefore, some tapes may be incompatible between the old and new versions of MAGNET.

- To read information from a tape verbatim, specify:

  FORMAT=(VAR/PRIME) for your source tape;
  FORMAT=(FIXED) for your destination disk file;
  (O*) translation for your source tape;
  LRECL=(the-same-large-number) for both source and destination;
  and use the MOVE subcommand.

- Use ASSIGN commands with -TPID options to get the first tapes mounted prior to invoking MAGNET.

- DISMOUNT messages repeat until a tape drive is offline and not ready. MOUNT messages repeat until a tape drive is both online and ready.

- Use the LABEL command to initialize a tape (see Chapter 6) prior to using labelled tape features within MAGNET.

- For MAGNET messages 152 and 153 (see Chapter 7), check for tape drives not assigned, powered-down, not ready, or offline.

- The best value to specify for the BUFFERS option is 3. A larger number will probably not produce any increase in speed. In fact, speed may decrease due to the larger amounts of storage accessed.

- Do not specify a BUFFERS option for the second or later objects in a chain. (For more information, see the descriptions of the PREVCHAIN and NEXTCHAIN options in Chapter 7.) Buffers that belong to the first object are reused for later objects.

- Tape objects should be reused for multiple tape files. In addition to opening files on tapes, MAGNET requires that the tapes themselves be opened. If you use several objects specifying different files on the same tape, the tape is rewound each time you specify a different object within a READ, WRITE, or MOVE subcommand. It is better, therefore, to use the MODIFY subcommand to change filenames or numbers prior to using different data transferral subcommands.

- All subcommands may be abbreviated to the first letter(s) of the subcommand. Thus, POSITION could be abbreviated to P, PO, POS, etc. LIST may be abbreviated to L, but LOAD may not be abbreviated shorter than LO. DECLARE may be abbreviated to DEC only. (An allowed alternative abbreviation for DECLARE is DCL.) DELETE may be abbreviated to D. DISPLAY may be abbreviated to DI. MOVE may be abbreviated to MOV, while MODIFY may only be abbreviated to MOD. Finally, READ may be abbreviated to R, while RENAME can only be abbreviated to REN. You cannot abbreviate options or option values. The one exception to this rule, however, is the response to the "ABSOLUTE OR RELATIVE?" question within the old form (pre-Rev 18.3) of the POSITION subcommand. In this case, you may respond with A or R.

Index

Index