M T S

The Michigan Terminal System

Magnetic Tapes

Volume 19

Reference R1019

October 1993

University of Michigan
Information Technology Division
Consulting and Support Services

DISCLAIMER

The MTS manuals are intended to represent the current state of the Michigan Terminal System (MTS), but because the system is constantly being developed, extended, and refined, sections of this volume will become obsolete. The user should refer to the *U-M Computing News*, Computing Center Memos, and future Updates to this volume for the latest information about changes to MTS.

CONTENTS

PREFACE

The software developed by the Information Technology Division staff for the operation of the high-speed IBM 370-compatible computers can be described as a multiprogramming supervisor that handles a number of resident, reentrant programs. Among them is a large subsystem, called MTS (Michigan Terminal System), for command interpretation, execution control, file management, and accounting maintenance. Most users interact with the computer's resources through MTS.

The MTS Volumes are a series of manuals that describe in detail the facilities provided by the Michigan Terminal System. Administrative policies of the Information Technology Division and the physical facilities provided are described in other publications.

The MTS Volumes now in print are listed below. The date indicates the most recent edition of each volume; however, since volumes are periodically updated, users should check the file *CCPUBLICATIONS, or watch for announcements in the *U-M Computing News*, to ensure that their MTS volumes are fully up to date.

Volume 1 *The Michigan Terminal System*, November 1988
Volume 2 *Public File Descriptions*, January 1987
Volume 3 *System Subroutine Descriptions*, April 1981
Volume 4 *Terminals and Networks in MTS*, July 1988
Volume 5 *System Services*, May 1983
Volume 6 *FORTRAN in MTS*, October 1983
Volume 7 *PL/I in MTS*, September 1982
Volume 8 *LISP and SLIP in MTS*, June 1976
Volume 9 *SNOBOL4 in MTS*, September 1975
Volume 10 *BASIC in MTS*, December 1980
Volume 11 *Plot Description System*, August 1978
Volume 12 *PIL/2 in MTS*, December 1974
Volume 13 *The Symbolic Debugging System*, September 1985
Volume 14 *360/370 Assemblers in MTS*, May 1983
Volume 15 *FORMAT and TEXT360*, April 1977
Volume 16 *ALGOL W in MTS*, September 1980
Volume 17 *Integrated Graphics System*, December 1980
Volume 18 *The MTS File Editor*, February 1988
Volume 19 *Magnetic Tapes*, October 1993
Volume 20 *Pascal in MTS*, January 1989
Volume 21 *MTS Command Extensions and Macros*, April 1986
Volume 22 *Utilisp in MTS*, May 1988
Volume 23 *Messaging and Conferencing in MTS*, August 1988

The numerical order of the volumes does not necessarily reflect the chronological order of their appearance; however, in general, the higher the number, the more specialized the volume. Volume 1, for example, introduces the user to MTS and describes in general the MTS operating system, while Volume 10 deals exclusively with BASIC.

The attempt to make each volume complete in itself and reasonably independent of others in the series naturally results in a certain amount of repetition. Public file descriptions, for example, may appear in more than one volume. However, this arrangement permits the user to buy only those volumes that serve his or her immediate needs.

Richard A. Salisbury
General Editor

PREFACE TO VOLUME 19

This edition reflects the changes that have occurred in the magnetic-tape support since October 1989.

The major change has been the addition of cartridge tape tape service. Also, 800 BPI 9-track tapes are no longer supported.

The description of floppy-disk support has been removed from this volume.

MAGNETIC TAPES

INTRODUCTION

This section is intended as a reference guide to the use of magnetic tapes in MTS. However, for current information on the procedures for submitting and retrieving tapes, the reader should consult the *Introduction to Magnetic Tapes*. Throughout this section, the reader will also be referred to *MTS Volume 1: The Michigan Terminal System*. Terms which are not self-evident or completely explained in other volumes of the MTS Manual are underlined and explained the first time they appear. Readers who have no prior experience with magnetic tapes may wish to read *Introduction to Magnetic Tapes*, which covers much of the same material as this section, but on an introductory level.

WHY USE MAGNETIC TAPES?

A *file* is a collection of logically related data stored externally to a program, e.g., on magnetic tape, disk, etc. Examples of files are:

(1) the source statements for a program

(2) the program in executable form after compilation

(3) the input to a program

(4) the output from a program

(5) the data from an experiment

Magnetic tapes are an economical way of storing large, infrequently used files. A cartridge tape, which costs about $6, can potentially hold the equivalent of more than 40,000 pages of disk file space (about 165 million bytes). At current rates, that disk space would cost about $168 per month.

The section "System-Supported Tape Blocking" explains how and why tapes should be blocked. When blocked correctly, magnetic tape files require somewhat less CPU time to read or write than do disk files. However, data stored on tape can only be accessed sequentially, not randomly as is possible with disk files. Thus, in order to read some data which is stored halfway out on a full reel of tape, it is necessary to space the tape to the appropriate point. A spacing operation might take 5 minutes or more of elapsed time, but only a few milliseconds of CPU time. The procedures for accessing data on tape are similar to those for accessing data in sequential disk files.

Before a tape can be used, it must be mounted (placed) on a tape unit by the machine operator. A labor charge is assessed for each tape mount. Each mount is subject to the availability of a tape unit of the required type. During the day, when tape unit usage is highest, it is sometimes difficult for terminal users to get tapes mounted. Batch jobs that require more tape units than are available are automatically rerun when enough tape units are available.

When exchanging programs or data with another computer installation, magnetic tapes are often the most practical medium of exchange. However, since the manner in which tapes are read and written varies from computer to computer, the sender of an exchange tape should provide complete information about the way the tape was written (see Appendix C).

In summary, magnetic tapes are useful for storing large files which are to be accessed sequentially.

MAGNETIC TAPE REELS

A *magnetic tape* is a long strip of 0.5-inch (12.7-millimeter) wide plastic material coated on one side with a magnetizable substance. A 9-track tape is wound on a plastic *reel* not larger than 10.5 inches (267 millimeters) in diameter. Tapes can be purchased in various lengths and on various size reels. Full-length tapes are 2400 feet (732 meters) in length. Shorter tapes are also available, ranging down to a few hundred feet in length.

An IBM 3480-compatible cartridge tape is made up of half-inch tape encased in a plastic cartridge measuring approximately 1 x 4 x 5 inches. This tape is available in standard and extended lengths, varying from 575 feet to 656 feet, all of which will work with MTS. Longer, similar-appearing tapes are generally not 3480-compatible. 3480-compatible cartridges are 18 track, and are recorded at a density of approximately 38,000 bits per inch (BPI). An extended-length cartridge tape can hold up to 40% more than a 2,400-foot reel tape when using large block sizes.

Magnetic tape reels are supplied with a removable, plastic *file-protect ring* (or *write-enabling ring*) which fits in a circular groove in the back of the reel. When the ring is removed, the tape drive is prevented from writing any information on the tape. Cartridge tapes have a thumbwheel in the side of the case; the position of the thumbwheel determines whether the tape can be written. If the user intends to write information on the tape, he or she must explicitly request that the operator insert the ring or position the thumbwheel (see the section "Mounting Magnetic Tapes"). Unless this is done, the tape may only be read.

Magnetic tapes may be classified according to ownership:

(1) *User tapes* are owned by individual users, are stored at the Computing Center, and can be used for the long-term storage of information.

(2) *Pool tapes* are owned by ITD, are available to all users, and are used for the temporary storage of information during the processing of a single job. After the completion of the job, the contents of pool tapes are no longer available.

User tapes must be purchased from a commercial vendor or University Stores. ITD does not sell or rent tapes. Users should purchase tapes that are certified by the manufacturer for use at 6250 *BPI* (bits per inch). In addition, the tapes should be *individually tested* or *certified* by the manufacturer. This is particularly important if large quantities of important data are to be written on the tape.

MAGNETIC TAPE HARDWARE CONCEPTS

Magnetic *tape units* (or *tape drives*) function as both input and output devices, transporting the tape from the supply reel to the takeup reel and reading or writing information as directed by a program. Placing a tape reel onto the unit is called *mounting* (or *loading*) the tape; *dismounting* (or *unloading*) is the opposite. Tapes are mounted and dismounted by the system operator.

A magnetic tape begins with about 10 feet of blank tape (less for a cartridge). This *leader* allows the threading of the tape through the feed mechanism of the tape unit. New tapes are supplied with a *load-point marker* and *end-of-tape marker* to enable the tape unit to sense the beginning and end of the usable portion of the tape. When a tape is mounted, it is positioned at the *load point* as indicated by

the load-point marker. *Rewinding* the tape after it has been read or written moves the tape back to the load point so that more reading or writing can take place from the beginning of the tape. Rewinding a full-length tape (2400 feet) generally takes about one minute of elapsed time. The portion of tape after the end-of-tape marker is referred to as the *end-of-tape area*.

Tracks

The MTS system at the University of Michigan supports 1600 and 6250 BPI 9-track tapes, and 3480-compatible cartridge tapes. Support of 800 BPI 9-track tapes has been discontinued. *9-track drives* read and write 9 parallel *tracks* of data lengthwise down the tape. Each *frame* (one *bit* from each track across the width of the tape) contains 8 bits of information (an 8-bit character) and one *parity* bit. 9-track drives are compatible with IBM System/370 architecture computers, since the main memories of these computers are organized into units of 8-bit bytes and characters are represented in the 8-bit EBCDIC character code.

The following definitions will help to clarify the last paragraph.

(1) EBCDIC is Extended Binary Coded Decimal Interchange Code, in which 8 bits are used to represent each character, thus allowing for 256 different characters. A table of the specific codes is given both in the *EBCDASC program description and the EBCASC system subroutine description in this volume.

(2) A *bit* is a binary digit, i.e., its value is either 0 or 1. The *parity bit* in each frame is an extra bit that is used for error checking. When a tape is written in *odd-parity mode*, the parity bits are set in such a way that each frame has an odd number of 1-bits. In other words, if a character has an even number of 1-bits, the associated parity bit is set to 1. Otherwise, the parity bit is set to 0. When a tape is written in *even-parity mode*, the parity bits are set in such a way that each frame has an even number of 1-bits. When reading a tape, the tape drive will signal a *data check* (indicating a possible error) whenever a frame has the wrong parity. 9-track tape drives *always* operate in odd-parity mode.

Density of 9-track Tapes

Density refers to the spacing between successive frames on a tape, i.e., the number of bits per inch (BPI) along the tape. The number of bits per inch is the same as the number of frames per inch. Data can be recorded on 9-track tapes at 1600 or 6250 BPI.

Tape manufacturers specify the maximum density at which each tape should be used. This indication is usually given in units of BPI, but may also be given in units of magnetic flux changes per inch (FCI). Generally, the maximum usable density is printed on a sticker which is placed on the tape reel by the manufacturer.

Tape units operating at 1600 BPI use the *phase encoding* (PE) technique. This recording technique explicitly records both 0- and 1-bits, i.e., a positive flux reversal indicates a 1-bit, a negative reversal indicates a 0-bit. PE produces, at most, 2 flux reversals per frame in each track. Thus, 3200 FCI certified tape is required for 1600 BPI.

Tape units operating at 6250 BPI use the Group Coded Recording (GCR) technique, in which data is assembled into data groups and translated into expanded codes (which are much less error susceptible) before being recorded on tape. The result of this group assembly and translation is greatly increased

reliability at the expense of extra bytes being recorded on the tape. GCR tapes are actually recorded at 9042 BPI NRZI, but without counting the extra bytes, the effective density is 6250 BPI. Generally, good-condition tape certified for use at 3200 FCI will be usable at 6250 BPI.

Of these two modes, 6250 (GCR) is the most reliable. A tape must be certified for the density at which it is to be used. When exchanging a tape with another computer installation, it is advisable to determine in advance what densities are available at that installation, since there is no industry-wide standard. For tapes to be used locally, 6250 BPI is the recommended density.

The tape units are designed to maintain each tape in only one density at any given time. This is accomplished by recording a "PE identification burst" (a special, non-standard block) at the load point for 1600 BPI tapes or a "GCR identification burst" at the load point for 6250 BPI tapes. Subsequently, the tape units will always recognize the density of the tape and will automatically read and write only at that density. The only time the user need be concerned about density is when rewriting the *first* block, so as to *change* the density (see the description of the MODE keyword in the section "Mounting Magnetic Tapes").

Blocks

On tape, information is arranged in *blocks*. A block is composed of the data transmitted to or from main memory during one I/O operation, i.e., a block is the *physical* unit of data that is read or written by the tape drive. In MTS, a block may be up to 32,767 bytes (characters) in length. When writing a tape, the system normally enforces a hardware recommended minimum of 18 bytes per block, by padding short blocks with blanks. Each pair of blocks is separated by an *interblock gap* (IBG), which is an area of blank (erased) tape approximately 0.32 inches (7.6 mm) long for 6250 BPI tape and 0.6 inches (15.2 mm) long for 1600 BPI tape. An IBG is automatically generated by the tape unit after each block is written. When the tape is being read, the IBG signals the end of the block.

In addition to the parity bit provided in each frame, each NRZI block is checksummed lengthwise by the addition of one or more check bytes at the end of the block. These check bytes are automatically generated by the tape unit on write operations and verified on read operations. The tape unit signals a *data check* error indication if any frame has incorrect parity (*vertical redundancy check*) or if the block check bytes are incorrect (*longitudinal* or *cyclic redundancy check*).

In principle, any number of blocks of data (presumably related in some way) make up a tape file. Actually, the number is restricted by the length of the tape. The tape unit, of course, has no concern about whether the data is logically related, just as MTS allows any data (related or not) to be placed in a disk file. However, the user who writes the data on the tape presumably organizes the tape into files of related data.

Each tape file is followed by a special block called a *tape mark* (or *file mark*), approximately 3.75 inches (95.3 mm) long (including the IBG) for all 9-track densities. The length of a tape mark may vary slightly depending on the density and the manufacturer of the tape unit. Tape marks are generated in such a way that they are distinguishable from all data blocks. Tape marks provide the mechanism for separating files. The tape unit can be commanded to search either forward (*forward-space file* – FSF) or backward (*backspace file* – BSF) for the next tape mark on the tape. This provides a very efficient mechanism (in terms of CPU time) for positioning the tape forward or backward in units of files. However, the tape unit may take several minutes of elapsed time to move the tape the specified number of files.

When writing a tape, the tape drive erases ahead of where it writes. Thus, when writing over an existing block, the tape drive may erase the beginning of the next block. This means that a block in the middle of a tape cannot be replaced reliably. The only way to change a block in the middle of a tape is to rewrite not only the block in question but also all following blocks.

There are two general ways to add new data to a tape:

(1) the new data may be written in a new file, or

(2) the new data may be added to the end of an existing file.

Normally, new data is written at the *logical end* of the tape, i.e., at a point immediately following the existing data. It is possible to write new data at a point in the middle of the tape (thus overwriting existing data), but the last new block (data or tape mark) must then be considered the new logical end of the tape. Beyond this point, any existing blocks will be inaccessible.

An example may help to clarify these concepts. Consider a tape which begins with 250 data blocks, followed by a tape mark, 17 more data blocks, and 6 tape marks. Multiple, consecutive tape marks are frequently, but not always, used to indicate the logical end of a tape, i.e., the end of the last file. Thus, the tape contains 2 files of 250 and 17 data blocks, respectively. Several observations may be made about the tape.

(1) No change can be made to any portion of the first file without eliminating the second file.

(2) To delete the last 14 blocks of the second file, the user would position the tape after block 3 of file 2 and then write a tape mark.

(3) To replace the last 6 blocks of the second file with any number of new blocks, the user would position the tape after block 11 of file 2, write the new blocks, and then write a tape mark.

(4) To add 3 blocks to the second file, the user would position the tape after the first of the group of 6 tape marks, then backspace to before that tape mark, then write 3 blocks followed by a tape mark.

(5) To add a third file to the tape, the user would position the tape after the first of the group of 6 tape marks, then write the new file followed by a tape mark.

Block Length

The approximate length (in inches) of a tape block can be computed by dividing the number of bytes (or characters) of data in the block by the density (in BPI) and adding the IBG length (0.65 for 1600 BPI, and 0.32 for 6250 BPI).

For a 12000-byte, 1600-BPI tape block:

Length = 12000/1600 + .65 = 8.15 inches

For a 12000-byte 6250 BPI tape block:

Length = 12000/6250 + .32 = 2.24 inches

SYSTEM-SUPPORTED TAPE BLOCKING

In almost every application, both the amount of system CPU time and the length of tape required can be greatly reduced if long blocks are utilized. Since most programs write relatively short records, the system normally performs a *blocking* procedure, combining many short records in a large system-provided buffer and writing out the buffer (a long block) each time it is full. When reading a tape, the system performs a *deblocking* procedure, reading a long block from the tape into a system-provided buffer and then subdividing it into many short records. This blocking/deblocking is done automatically, according to blocking parameters specified by the user.

The following table shows how much tape and how many blocks would be required to write 10,000 100-byte records at 1600 BPI, using 4 different block sizes.

Block Size (bytes)	*Total Length (feet)*	*Number of Blocks*
100	594	10000
4000	66	250
10000	58	100
30000	54	34

With a block size of 100 bytes, the tape would be about 594 feet long and have 10,000 blocks, i.e., 10,000 I/O operations would be required to write or read the tape. With a block size of 30000 bytes (each block containing 300 100-byte records), the tape length and the number of blocks would be drastically reduced.

When block size is increased, the number of blocks and the number of interblock gaps (IBGs) is decreased. Thus, the fraction of tape occupied by data is increased relative to the fraction used by IBGs.

For example, on an unblocked, 6250 BPI tape containing 80-byte records, the data occupies less than 4% of every foot of tape, while the IBGs take over 96%. If the records are blocked into 8000-byte blocks (100 lines per block), the data takes nearly 80%, while the IBGs take only about 20%. As a result of the blocking, the length of tape required for a given amount of data is reduced by a factor of 21 and the number of physical I/O operations is reduced by a factor of 100.

In general, block sizes on the order of 8000 to 16000 bytes are recommended for writing tapes. Blocks of this length will provide reasonable efficiency (i.e., a substantial reduction in the number of blocks and therefore the number of I/O operations) and at the same time conserve the amount of tape required.

Blocking Parameters

For each currently mounted tape, the system maintains a set of parameters which collectively describe the current status of the tape. Many of these parameters can be specified by the user when the tape is mounted (see the section “Mounting Magnetic Tapes” below). The parameters can subsequently be changed by control commands or programs (see the section “Control Commands” below). Three of the parameters (format, block size, and logical record length) are related to the use of

blocking. They are referred to collectively as the *blocking parameters*.

Block Size

The *block size* parameter (also called the *SIZE* parameter) sets an upper bound on the length of any block read or written. Any block which is longer than SIZE will be truncated.

For example, when a tape with 4000-byte blocks is read using a SIZE of 3000, only the *first* 3000 bytes of each block will be read, the remainder being ignored. Similarly, if an attempt is made to write a block of length greater than 3000, only the first 3000 bytes of the block will actually be written.

The SIZE parameter must have a value between 18 (normally) and 32767. Because tape drives cannot reliably read very short blocks, the system will not normally write very short blocks. The system does allow the minimum block size to be set by the user, but 18 is the recommended minimum and the default. When writing, any block which is less than 18 bytes long will be padded to 18 bytes with trailing blanks. The SIZE parameter cannot be changed in the middle of a blocked file.

Records

A *record* is defined as the basic unit of information which can be exchanged between a program and the MTS I/O subroutines (READ, WRITE, SCARDS, et al.). For a terminal, a record is an input or output line. For a line file, a record is a line. The $COPY command, for example, copies records. A record is sometimes referred to as a *logical record*.

Blocking

Blocking is defined as the process of combining one or more records into a block before writing. *Deblocking*, the reverse, is the process of subdividing a block into records after reading it. The purpose of blocking is to increase the efficiency of I/O operations, i.e., to reduce the CPU time and real time required and to reduce the length of tape wasted in interblock gaps (IBGs). By blocking a tape, it is possible to make the blocks as long as necessary even though the records themselves are short. On an *unblocked tape*, records and blocks are equivalent, i.e., no blocking or deblocking is performed.

Tape blocking can be enabled or disabled by the user, thus allowing any user or system utility program to use the standard system blocking facilities described here, to do its own blocking and deblocking, or to do no blocking at all. Some programs (e.g., *SORT, FORTRAN programs which use the unformatted READ and WRITE statements, and PL/I programs) provide built-in support for blocking because they are able to increase performance by making fewer calls on the system I/O routines (one call per block, instead of one call per record). PL/I users should read Appendix H for more specific information on PL/I blocking. When using such programs, it is very important to make sure that either the program or the system is doing the blocking, but *not both*.

There are two basic ways in which tapes can be blocked and deblocked using the system tape support. With *fixed-length blocking*, all records in a given tape file are forced to be of the same length. When such a tape file is written, the system will pad short records with trailing blanks to the specified length and truncate long records to the specified length, thus forcing all records in the file to be of exactly the same length. This is a very efficient blocking scheme (in terms of CPU time) and lends itself particularly well to data whose records are all of the same length.

Variable-length blocking is used for records which are not all of the same length. In this case, the specified record length is used only as an upper bound (for truncation) and short records are *not*

padded.

For example, if the data in a file consists of variable-length printer output lines, then the use of variable-length blocking is appropriate. If fixed-length blocking were employed for such data, a record length equal to the length of the longest possible line would be required in order to eliminate the possibility of truncation. The disadvantage of fixed-length blocking would be that any shorter line would be padded with trailing blanks, thus making all records, short or long, take the maximum amount of space. With variable-length blocking, the short records will not be padded. However, an overhead of approximately 4 extra bytes per record is required to delimit the records within the block. Despite this overhead, if the records vary considerably in length, there will be a net saving of space on the tape.

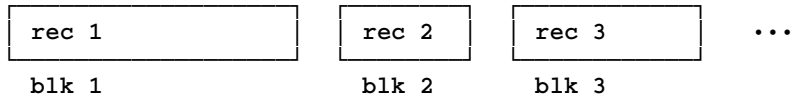
Record Length

The *logical record length* (LRECL) parameter specifies the record length for fixed-length blocking and an upper bound on the record length for variable-length blocking. The LRECL parameter must have a value between 1 and 32767, and cannot be changed in the middle of a blocked file.

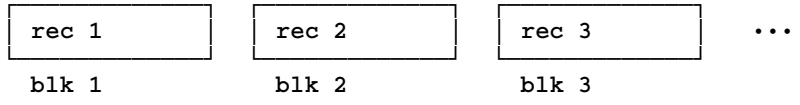
Formats

The *format* parameter determines the manner in which a tape file is blocked and deblocked. The following formats are available:

U Format U (undefined length) is equivalent to no blocking at all. Each record is a block, and therefore the logical record length (LRECL) is irrelevant. The block size (SIZE) is an upper bound on block lengths, i.e., blocks longer than SIZE will be truncated, but blocks shorter than SIZE will not be padded.

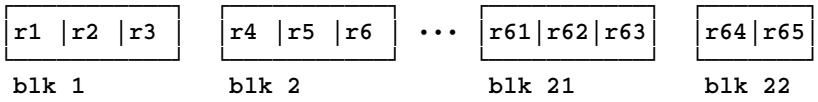


F Format F (fixed-length) is also for unblocked records. This format is similar to format U, except that the system will truncate or pad each record so that its length is exactly equal to the logical record length (LRECL). With this format, the block size (SIZE) must be at least as great as LRECL. As an example of the use of format F, suppose that unblocked 80-byte lines are to be stored on a tape, but that the lines are stored in an MTS line file and that trailing blanks have been trimmed off. In this case, writing the tape with format F and a SIZE and LRECL of 80 will cause any "short" lines to be padded with blanks.

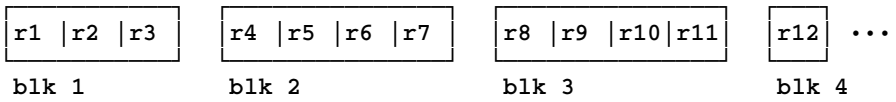


FBS Format FBS (fixed-length, blocked, standard) is for fixed-length blocking, i.e., the blocking of fixed-length records into fixed-length blocks. With this format, each record is truncated or padded to the logical record length (LRECL). The block size (SIZE) is an upper bound on block lengths, and must be at least as great as LRECL. When writing a format FBS file, the system will combine as many records as possible into each block without allowing the block length to exceed SIZE. *Only* the last block to be written can

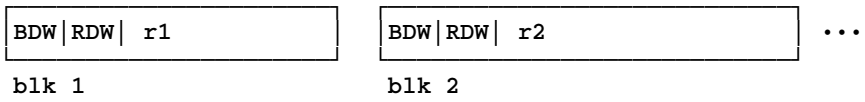
be shorter than the others, and then only if there are not enough records to fill it completely. When reading, the system will read blocks from the tape, then deblock successive records of length LRECL. If the last record of a block is shorter than LRECL, the system will pad the record with enough trailing blanks to form a full-length record. When reading, SIZE must be at least as great as the longest block (so that no block is truncated).



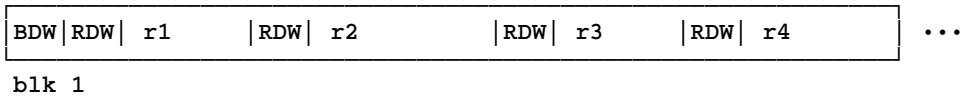
FB Format FB (fixed-length, blocked) is the same as format FBS, except that the IBM definition of format FB allows short blocks other than at the end of a file. When writing a format FB file, MTS will make *all* blocks the maximum length except possibly the last one to be written. When reading a format FB or FBS file, MTS will accept short blocks in the middle of the file.



V Format V (variable-length) is for unblocked, variable-length records. This format is similar to format U, except that each format V block begins with 8 extra control bytes: a 4-byte *block descriptor word* (BDW) containing the length of the block, followed by a 4-byte *record descriptor word* (RDW) containing the length of the record. With this format, the logical record length (LRECL) is an upper bound on record lengths, i.e., records longer than LRECL are truncated, but records shorter than LRECL are not padded. When writing, the block size (SIZE) must be at least 8 greater than LRECL to allow for the block and record descriptor words. When reading, SIZE must be at least as great as the longest block (so that no block is truncated). Format V offers one minor advantage over format U: when reading a format V file, the system will check that the block length as determined by the tape unit agrees with the length as recorded in the block descriptor word.

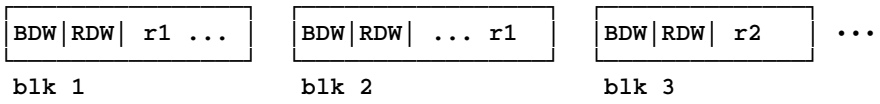


VB Format VB (variable-length, blocked) is for variable-length blocking, i.e., the blocking of variable-length records into variable-length blocks. At the beginning of each block, there is a 4-byte block descriptor word containing the length of the block, and at the beginning of each record within the block, there is a 4-byte record descriptor word containing the length of the record. Thus, format VB has an overhead of 4 bytes per block, plus 4 bytes per record. With this format, the logical record length (LRECL) is an upper bound on record lengths, i.e., records longer than LRECL are truncated, but records shorter than LRECL are not padded. The block size (SIZE) is an upper bound on block lengths. When writing a format VB file, the system will combine as many records as possible into each block without allowing the block length to exceed SIZE. When writing, SIZE must be at least 8 greater than LRECL, to allow for the block and record descriptor words. When reading, SIZE must be at least as great as the longest block (so that no block is truncated).

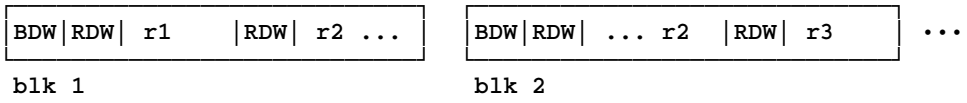


A variation on formats V and VB permits records to be spread between several blocks. This is referred to as *spanning*. It is thus possible to have records which are longer than the blocks used to block them.

VS Format VS (variable-length, spanned) takes each output record, truncated to the logical record length (LRECL), and continues it as necessary from block to block. The maximum length of each block is the block size (SIZE). With format VS, LRECL can be greater than SIZE, since a record may extend over several blocks. The portion of a record appearing in one block is called a *segment*. With format VS, there is only one segment per block. In each record descriptor word (or, more precisely, each segment descriptor word), two bits are used to indicate if the segment is the first, intermediate, or last segment of a record. When reading a format VS file, SIZE must be at least as great as the longest block (so that no block is truncated).



VBS Format VBS (variable-length, blocked, spanned) is identical to format VS except that a given block may contain segments for more than one record.



When reading a tape file, the system treats formats F, FB, and FBS identically and formats V, VB, VS, and VBS identically.

The following formats are used only with ANSI-labeled tapes.

D Format D (decimal) is for unblocked, variable-length records. This format is similar to V format except that the record-descriptor word containing the record length is a 4-character decimal number in ASCII. The maximum record length is restricted to 9995 bytes.

DB Format DB (decimal, blocked) is for blocked, variable-length records. This format is similar to VB format except that the record-descriptor word containing the record length is a 4-character decimal number in ASCII. The maximum record length is restricted to 9995 bytes.

SYSTEM-SUPPORTED LABELED TAPES

Magnetic tapes can be either labeled or unlabeled. A labeled tape contains additional blocks (called labels) which are generated and used by the operating system to record useful information about the tape, e.g., how it is blocked. An unlabeled tape contains only those blocks which are explicitly written by the user.

By default, labeled tapes are processed using the MTS tape-labeling scheme which is a subset of the OS/VS tape-labeling scheme. Other tape-labeling schemes that may be used in MTS are the VLO tape-labeling scheme and the ANSI tape-labeling scheme. These alternative schemes are requested by specifying the LBLTYPE keyword on the mount request. Also, MTS can read labeled tapes produced by the IBM DOS/TOS operating systems (but it cannot write tapes to be used on these systems).

The MTS tape-labeling scheme is compatible with that used in the IBM OS/VS operating systems. The format for the MTS tape-labeling scheme is given in Appendix D of this section. The format of the OS/VS tape-labeling scheme is given in the IBM publication, *MVS/ESA Magnetic Tape Labels and File Structure Administration* (SC26-4511).

The VLO (Volume Label Only) tape-labeling scheme is similar to the MTS tape-labeling scheme except that only the volume label is present. The individual files on the tape are unlabeled. The VLO labeling scheme is useful for tapes containing large numbers of small files where the presence of separate file labels would consume too much space on the tape. The format for the VLO tape-labeling scheme is given in Appendix E to this section. When reading a VLO tape, the LBLTYPE=VLO mount parameter must be specified each time the tape is mounted, otherwise the tape will appear to be empty.

The ANSI (American National Standard Institute) tape-labeling scheme is used by several non-IBM computer vendors as their standard for labeled magnetic tapes. Most IBM and IBM-compatible systems use the IBM-labeling scheme which is compatible with the MTS-labeling scheme. The ANSI-labeling scheme will be useful mainly for the exchanging of tapes with computer installations that do not support the IBM standard, but do support the ANSI standard. The format for the ANSI tape-labeling scheme is given in Appendix F to this section.

The extent of the ANSI support includes the capability of reading and writing magnetic tapes with ANSI labels, blocking and deblocking of tape records according to the formats U, F, V, and D, and automatic translation of characters from ASCII to EBCDIC on input and from EBCDIC to ASCII on output using the standard MTS translation tables.

Advantages of Labeled Tapes

The primary advantages of labeled tapes are security, convenience, and reliability (through increased error checking). Labels are processed automatically by the operating system. Thus, labeled tapes require very little additional effort on the part of the user.

The first block on a labeled tape is called a *volume label*. This block is written at the time the tape is initialized, and is automatically read by the system every time the tape is mounted. The volume label contains, among other things, the *volume name* of the tape (referred to as the *volume serial number* in IBM publications), which is a string of 1 to 6 characters not containing commas or embedded blanks. This volume name is checked each time the tape is mounted. Before allowing the tape to be mounted, the system will compare the volume name given in the tape catalog (or in the mount request) with the volume name recorded on the tape. If these do not agree, the tape will not be mounted. This procedure helps to protect the tape against accidental or unauthorized use.

With labeled tapes, each file (referred to as a *data set* in IBM publications) contains not only data blocks, but also several additional blocks called *file labels*. These are automatically processed by the operating system and are used to store useful information about the file. Among other things, the file labels contain the *file name* (or *data set name*), which is a string of from 1 to 17 characters which may not contain commas or embedded blanks and may not both begin and end with asterisks. The user has

the option of specifying a file name when the file is written or allowing MTS to assign a default name of the form

Vvvvvvv.Fnnnn

where "vvvvvv" is the tape volume name and "nnnn" is the four-digit file number. For example, if a labeled tape has the volume name DT30A, then the ninth file will have the default name VDT30A.F0009.

File names can be used to position the tape. This method of positioning is more convenient and positive than positioning by file number (as is required for unlabeled tapes).

Labeled tapes allow additional error checking by the system, using the additional information associated with each file. The security provided by this checking is nearly always worth the small amount of space required on the tape and the small amount of CPU time required to check or generate the label information at the beginning and end of each file. The checking or generating of this label information is generally transparent to the user.

In addition to the file name, a complete set of *blocking parameters* (format, block size, and logical record length) is stored in the labels for each file. This information is generated by the system when the file is written and is automatically retrieved and used to deblock the file when it is read. However, it is always possible for the user to override the blocking parameters contained in the labels (see the section "Control Commands" below). Note: labeled tapes generated by the IBM DOS and TOS systems do not contain blocking information. Therefore, the blocking parameters must be supplied by the user, just as for unlabeled tapes.

An *expiration date* can be specified for each file on a labeled tape. This is also stored in the file labels. Prior to the expiration date, the file cannot be modified or replaced. Only if a file has "expired" may it be changed. If there are other files following the one being replaced or modified, they are assumed to expire on the same date, i.e., their expiration dates are *not* checked and they will be inaccessible after the modification of the current file. In some cases, the user may wish to disable expiration-date checking so that existing files can be changed (see the DTCHK command description in the section "Control Commands" below).

For record-keeping purposes, the system also saves the current date, userID, and batch receipt number (if a batch job) as part of the label information as each file is written.

Disadvantages of Labeled Tapes

The primary disadvantage of labeled tapes is that they may be difficult to use in other, non-IBM-compatible systems. Each system expects tape labels to be in a specific format. The IBM OS/VS labeling scheme, although used widely on IBM systems, is by no means an industry-wide standard. Thus, it is always advisable to find out in advance what impact the presence of labels may have on the use of a tape at some other installation. ANSI labeled tapes are acceptable in MTS and at most other non-MTS installations.

Even though the IBM OS/VS labeling scheme is compatible with that used in MTS, additional restrictions may apply to volume and file names. If a tape written in MTS is to be processed by an OS/VS system, it is advisable to find out in advance whether a particular name is acceptable to that system. Conversely, the MTS labeling scheme does not support a few of the options available in the OS/VS labeling scheme. In particular, MTS does not allow a file to be continued on multiple volumes

(reels) of tape, and does not allow user-defined file labels (in addition to the system-defined labels).

A second and minor disadvantage of labeled tapes is processing inefficiency: labels take additional space on the tape (about 1 foot per file) and label processing requires additional CPU time. However, both of these factors are generally so small as to be insignificant.

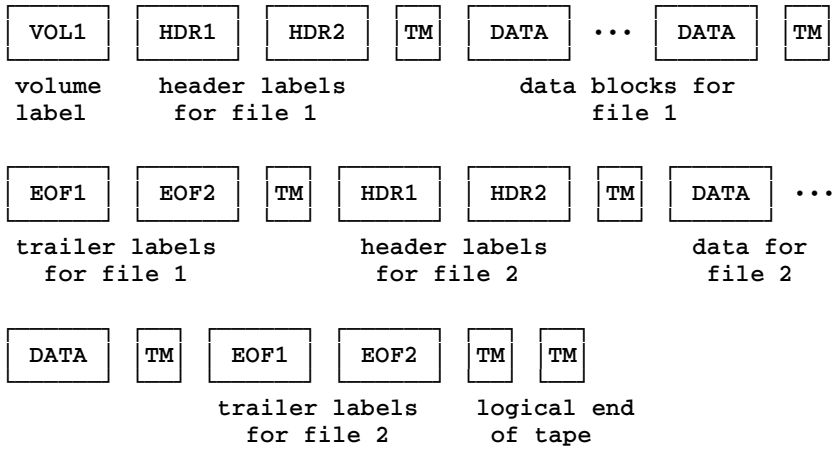
Volume Label

By definition, in MTS a *labeled tape* is any tape which begins with a *volume label*. This is an 80-byte block which has "VOL1" as the first 4 characters. The volume label contains the 6-character volume name (beginning with byte 5), and a 10-character *owner identification* (beginning with byte 42). The volume label is automatically processed by the system so that its presence is transparent to the user.

File Labels

Each file on a labeled tape has six extra blocks: two 80-byte *file header labels* (HDR1 and HDR2) and a tape mark at the beginning and two *file trailer labels* (EOF1 and EOF2) and a tape mark at the end. The labels contain information which describes the file. For details on label formats, see Appendix D. During normal operations, the labels are processed automatically by the system. When the user positions, reads, or writes the tape, the presence of the labels and extra tape marks may be disregarded. Each file appears to consist of zero or more data blocks followed by a single tape mark.

The diagram below illustrates a labeled tape with two files. The box labeled VOL1 represents the volume label. The HDR1 and HDR2 boxes are the header labels for each file, while the EOF1 and EOF2 boxes are the trailer labels. Each TM box represents a tape mark.



The last tape mark indicates that there are no more files on this labeled tape. If there were a third file, its header labels would appear in place of the last tape mark. The data portion of each file consists of zero or more data blocks. It is these data blocks which are read or written in response to program requests and user commands. The labels are implicitly read or written by the system and are transparent to the user. The equivalent unlabeled tape would have only the data blocks for file 1, a tape mark, the data blocks for file 2, and a tape mark.

Blocking Considerations

A complete set of blocking parameters (format, block size, and logical record length) is stored in the labels for each file. When the file is entered (from either end), the system will normally change the current blocking parameters to correspond to the blocking parameters in the file labels. Thus, when reading the file, the system will automatically deblock the file according to the blocking parameters in the file labels. (The user need not specify the parameters.) In some cases, however, the user may want to deblock the file according to a different set of parameters. These may be specified by giving the appropriate control command (see the section "Control Commands" below).

When an existing file on a labeled tape is replaced by a new file, the system will use the blocking parameters from the labels of the existing file, unless the user explicitly specifies new parameters (see the section "Control Commands" below). When an existing file on a labeled tape is modified or appended (not totally replaced), the system will also use the blocking parameters from the labels of the existing file. In this case, the user may not specify new parameters, since the parameters may not change in the middle of the file. If a new file is added to the *end* of a labeled tape, the system will use the blocking parameters from the labels of the *preceding* file, unless the user explicitly specifies new parameters.

Density Considerations

Each labeled tape must be of uniform density. For a 9-track labeled tape, the density can be changed only if the tape is reinitialized (by the INIT mount keyword parameter or the INIT control command) or relabeled (by the *LABEL program or the TAPEINIT subroutine).

System pool tapes are labeled and can be reinitialized, but not relabeled. The density of a pool tape can be specified when the tape is mounted. If not specified, pool tapes will be 6250 BPI.

MOUNTING MAGNETIC TAPES

The $MOUNT command may be used to issue a mount request to the system operator. There is also the MOUNT system subroutine which may be used by a program to issue a mount request.

A *mount request* consists of all of the information needed to mount a single tape. The general form of a mount request is:

```
tapename [*pdn* [keywords]]
```

One or more requests of this form can be entered either directly on the $MOUNT command separated by semicolons, or as separate input lines (read from *SOURCE*) terminated by an end-of-file:

```
$MOUNT request

$MOUNT request; request; ...

$MOUNT
request
request
  ...
$ENDFILE
```

When a request is too long for one input line, it can be continued on the next line by ending the first line with the current MTS command-line continuation character (initially a minus sign).

If the system detects a syntactic error in a mount request from a terminal user, it will prompt the user to either enter a replacement for the faulty parameter or reenter the *entire request*. (Note that a request begins with a tape name, not with $MOUNT).

When the mount requests have been determined to be free of syntactic errors, they are displayed on the operator console. The operator will then check whether enough drives are available, and if possible, proceed to mount the tapes. (If several tapes are to be mounted simultaneously, it is best to give all of the mount requests in a single $MOUNT command. Otherwise, the operator may mount some of the tapes before enough drives are available to mount all of them.) The operator will inform the system if there were any problems with the tapes (e.g., inconsistent identifying parameters). The system will then give the user a message indicating the outcome of the mount operations.

If all of the tape drives are busy, the user will be prompted whether to wait for an available drive or to cancel the mount request (see the description below of the WAIT keyword for further details).

The following sections describe the individual parameters in a mount request.

Tape Name

Each user tape has a *tape name*. This name, unless it has been changed, corresponds to the receipt number assigned by the program *TAPESUBMIT when the tape was submitted for use (see *Introduction to Magnetic Tapes*). The tape is further identified by an external *tape ID* (a string of 1 to 10 identifying characters which the user selects when the tape is submitted). Both the tape name and the tape ID are written on a tape label which is affixed to the tape reel. The tape name subsequently may be changed by the user by running the program *TAPES (see Appendix G).

When mounting a user tape, the user must specify the tape name. When mounting a pool tape, the word POOL is used in place of the tape name.

Pseudodevice Name

The user may specify a *pseudodevice name* (PDN) for each tape to be mounted. This PDN will be used as the FDname of the tape, so that the tape can be referenced by programs or commands. A tape PDN must begin and end with asterisks, and may have from one to fourteen other characters, generally letters and digits (the same set of characters that are valid for file names). Furthermore, a tape PDN may not conflict with predefined MTS PDNs such as *SOURCE* and *SINK* (see the paragraphs on pseudodevice names in the section “Files and Devices” in *MTS Volume 1: The Michigan Terminal System*).

The pseudodevice name may be omitted from the mount request if only a single tape is being mounted. In this case, the default pseudodevice name from the tape catalog will be used; unless changed, this defaults to *T*.

Volume Label

If the tape is a labeleled tape, a volume label must be supplied. By default, the volume label is taken from the tapes database. However, as an added security measure when writing to a tape, the user may supply the volume label on the MOUNT command, in which case it must agree with the

volume label recorded on the tape.

Mount Keywords

There are many different keyword parameters which may be entered as part of each mount request. Successive keyword parameters must be separated by blanks or commas but may be entered in any order. However, the keyword parameters must appear after the PDN. The following table summarizes the available keyword parameters. When parameters are omitted from a mount request, the default values are assumed.

Keyword Parameter	*Function*
BLKPFX=n	Specify block-prefix length for ANSI labeled tapes (0≤n≤99); defaults to 0 for initialized tapes
{BLOCKING | BLK}={ON | OFF}	Enable or disable blocking; defaults to ON
CC={A | M}	Specify nonblank control character for next new file written
{DSNAME | DSN | NAME}=name	Specify file name to be used for next new file written
DTCHK={ON | OFF}	Enable or disable expiration date checking; defaults to ON
EXPDT[={mm-dd-yy | mm/dd/yy}]	Specify expiration date for new files
{FORMAT | FMT | RECFM}=fmt[([size][,lrecl])] where "fmt" is {U | F | FB | FBS | V | VB | VBS | D | DB}	Specify blocking format and, optionally, block size and/or logical record length; defaults to U(32767) for unlabeled tapes
INIT={YES | NO}	Initialize labeled tape; defaults to NO
LBLTYPE={MTS | OS/VS | VLO | ANSI}	Specify label type; defaults to MTS
LP={ON | OFF}	Enable or disable label processing; defaults ON for labeled tapes, OFF otherwise
LRECL=n	Specify logical record length (1 ≤ n ≤ 32767); defaults to 32767 for unlabeled tapes
MINSIZE=n	Specify minimum block size (1 ≤ n ≤ 100)
MODE=mode	Specify tape mode; defaults to the current mode
POSN={* | *n* | *EOT* | name}	Position tape to beginning of current file, *n*th file, end-of-tape, or data set name

QUIT={YES | NO}	Control termination of batch job if mount fails; defaults to YES
RETRY=n	Specify retry count for read errors ($0 \le n \le 15$); defaults to 10
RING={IN | OUT}	Specify placement of file-protect ring; defaults to OUT
{SIZE | BLKSIZE}=n	Specify maximum block size ($18 \le n \le 32767$); defaults to 32767 for unlabeled tapes
TIMER={ON | OFF | n}	Enable, disable, or specify elapsed-time interval (in minutes) for inactive tape warning message; defaults to ON, n=15 minutes
TRANSLATE={MTS(parity) | MTS | IBM | NONE}	Specify translation scheme for ANSI labeled or ASCII unlabeled tapes; parity is EVEN, ODD, ZERO, or ONE; defaults to MTS(ZERO)
{VOLUME | VOL | LABEL | VOLSER}={name | 'name'}	Specify volume name of labeled tape
WAIT={YES | NO | PROMPT}	Control terminal-user waiting for queuing of mount requests when no tape drives available; the default is PROMPT
WRITE={YES | NO}	Specify placement of file-protect ring; defaults to NO

Blocking Keywords (BLOCKING, FORMAT, LRECL, SIZE, MINSIZE, and BLKPFX)

The BLOCKING keyword parameter may be used to enable or disable the system tape-blocking facility. By default, blocking is enabled (ON). If blocking is disabled (OFF), the system will not perform any blocking of input or output records regardless of what format is specified, i.e., it will operate as though the blocking format were set to U. This is useful when reading or writing a blocked, labeled tape where the blocking is to be done by a program and not by the system tape-blocking facility. For example, if the program is writing data which *it* has blocked into format FB blocks and if blocking has been disabled, then the system will write each block directly onto the tape without performing any blocking function.

When writing a new file on a labeled tape with BLOCKING set OFF, the system will store the current blocking parameters in the file labels even though it will not use them to do any blocking. Thus, even if blocking is to be done by a program, the user should specify the correct blocking parameters before the first block in the file is written.

The FORMAT, LRECL, and SIZE keywords may be used to set the current blocking parameters when a tape is mounted. The FORMAT keyword may also be used to set the SIZE and/or LRECL parameters. For an unlabeled tape, the default blocking parameters are U(32767), i.e., format U with a block size of 32767. For a labeled tape that has no existing files (i.e., a pool tape or a tape that has been freshly initialized), the default parameters are VBS(16384,32767). For a labeled tape that has existing files, the default parameters are obtained from the labels of the file to which the tape is positioned during the mount operation. This will be either the first file on the tape or some file

determined by the POSN keyword in the mount request. If the tape is positioned to its logical end, however, the default parameters are obtained from the labels of the preceding file. After the tape has been mounted, the current blocking parameters may be changed by giving the appropriate control commands (see the section "Control Commands").

For an unlabeled tape, the current blocking parameters will remain the same until explicitly changed by a control command or program. However, for a labeled tape (except one that was generated by an IBM DOS or TOS system), whenever a file is entered, the current blocking parameters are changed to correspond to the blocking parameters in the file labels. Thus, the blocking parameters specified in the mount request will apply only to the file to which the tape was positioned during the mount operation. If the tape is repositioned to some other file, the current blocking parameters will generally be changed.

The MINSIZE keyword specifies the minimum length for blocks to be written. Any block with fewer bytes than MINSIZE will be padded with blanks. The default MINSIZE is 18 bytes. This parameter may also be changed by giving the MINSIZE control command (see the section "Control Commands" below). The MINSIZE keyword has no effect when reading a tape.

The BLKPFX keyword specifies the length of the block-prefix field (BDW field) that begins variable-length blocks on ANSI labeled tapes. This defaults to the value specified in the ANSI HDR2 label on input, and on output, to the value from the label field of the previous file. If the block-prefix length is not given in a label and not specified by the BLKPFX command, MTS will use a value of zero. If the block-prefix length is four, MTS will interpret the first four characters as a 2-byte binary number for format V, and as a 4-character ASCII decimal number for format D. On output, MTS will generate such a block descriptor if BLKPFX=4.

Label Processing Keywords (NAME, DTCHK, EXPDT, INIT, LP, VOLUME, CC, and LBLTYPE)

When mounting a tape, the system checks whether the tape is labeled or not. If it is labeled, the volume label must be supplied either implicitly from the tapes database or explicitly by specifying the VOLUME keyword parameter in the mount request and in which case the volume name given must agree with the volume name recorded on the tape. If the tape is not labeled and the VOLUME keyword is given, the equals sign "=" must be followed by a blank or be the last character of the mount request.

If the tape unit is unable to read the first block without getting an unrecoverable read error, the system cannot verify whether or not a volume label is present. In this situation, the system will print an error message indicating that the first block cannot be read, and the tape will *not* be mounted. If a tape is totally blank, there will be no first block at all and the tape cannot be mounted. In general, a blank tape must be initialized before it can be mounted. The procedures for initializing a tape are described in *Introduction to Magnetic Tapes*, Tutorial T7002. If the beginning of the tape has been physically damaged, the first block may be unreadable.

The LP (label processing) keyword parameter may be used to enable or disable label processing. If the tape is labeled, LP defaults ON (enabled); if not, it defaults OFF (disabled). When label processing is enabled, the system expects to find the appropriate labels for each existing file and will generate the appropriate labels whenever a new file is written. When disabled, the system assumes nothing about the tape structure, i.e., all blocks are treated as data blocks on input and no labels are generated on output. (Note: even if the LP=OFF keyword parameter is given in a mount request for a labeled tape, the volume name will still be checked.) Label processing may also be enabled or disabled after the tape has been mounted (see the section "Control Commands" below).

Normally, when an attempt is made to modify or replace an existing file on a labeled tape, MTS will check that the file has expired (i.e., that the expiration date has been reached) before allowing the modification to take place. If other files follow the one being replaced or modified, they are assumed to expire on the same date, i.e., their expiration dates are *not* checked. When a labeled tape is mounted, the user may specify (DTCHK=OFF) that expiration dates are not to be checked at all. The default is DTCHK=ON. Expiration-date checking may also be enabled or disabled after the tape has been mounted (see the section "Control Commands" below).

The expiration date for all *new* files to be written on a labeled tape may be specified as EXPDT=mm-dd-yy. If no expiration date is given and the tape has no existing files, the system will use the date 01-00-00 for all new files (i.e., the files will already have expired). If no expiration date is given and the tape does have existing files, then when an existing file is replaced, the system will use the expiration date from the existing file; when a new file is added to the logical end of the tape, the system will use the expiration date from the preceding file. The expiration date for new files may also be specified after the tape has been mounted (see the section "Control Commands" below).

The INIT=YES keyword parameter may be given for any labeled tape which is mounted with RING=IN or WRITE=YES. When given, this parameter causes *all* files currently on the tape to be logically deleted, i.e., the tape is reinitialized. If the tape is 9-track and the MODE keyword is also given, the tape may be reinitialized at a different density. Otherwise, the tape will be reinitialized at the same density. In either case, the result is an "empty" labeled tape, equivalent to a tape which has been freshly labeled. Needless to say, the default for this parameter is NO (except for pool tapes, which are always initialized).

The NAME keyword parameter allows the user to specify (in advance) a name for the next new file (i.e., the next complete file) to be written on a labeled tape. File names may also be specified after the tape has been mounted (see the section "Control Commands") or be allowed to default to a name consisting of the letter V, followed by the 1 to 6 character volume name, followed by a period and the letter F, followed by the four-digit file number.

The CC keyword may be used to set the control-character field in the HDR2 and EOF2/EOV2 labels. The control-character field is used at OS/VS installations, but not at MTS installations, to indicate the type of carriage-control characters used. The control-character values are:

A	uses ASCII control characters
M	uses machine control characters
blank	uses no control characters

MTS normally writes a blank in this field (byte 37) unless the CC keyword is given before the first record of the file is written. This keyword affects only the *next* new file written.

The LBLTYPE keyword specifies the type of tape-labeling scheme to be used. LBLTYPE=MTS specifies that the MTS tape-labeling scheme is to be used and is the default. OS/VS is a synonym for MTS. LBLTYPE=VLO specifies that the tape contains VLO (Volume Label Only) labels. LBLTYPE=ANSI specifies that the tape contains ANSI labels. For ANSI tapes, the LBLTYPE keyword is optional since MTS will automatically recognize ANSI labeled tapes if the VOLUME keyword is given.

MODE Keyword

When the tape is mounted, the unit will automatically recognize the current density of the tape. Thus, when mounting a tape, the MODE need not be specified unless the user wishes to *change* the

density of the tape. To change the density, the user must specify the *new* density (MODE). For a labeled tape, the density change is accomplished when the tape is reinitialized (by using the INIT keyword on the mount request or the INIT control command) or relabeled (by the *LABEL program or the TAPEINIT subroutine). For an unlabeled tape, the density change is accomplished when the first block is rewritten.

POSN Keyword

The POSN (position) keyword parameter may be used to initially position a tape:

(1) to the n*th* file,

(2) to the logical end of the tape (if labeled), or

(3) to a file with a particular name (if labeled).

For unlabeled tapes, only form (1) is permissible. For labeled tapes, if the POSN keyword is given in a mount request, then any blocking keyword parameters also given in the mount request will override the blocking parameters in the file labels.

If no POSN keyword is given in a mount request, the tape will be left in position at the load point (at the beginning of the first file). After a tape has been mounted, it may be positioned as many times as required (see the section "Control Commands").

Unsuccessful Mounts (QUIT)

An error in a batch mount request will normally result in the job being signed off. However, the QUIT=NO keyword parameter may be used to inhibit this automatic signoff. A mount request can fail for several reasons: because the user has given a faulty tape name or tape ID, because the volume name was incorrect, because the POSN keyword was specified improperly, etc. In any case, an error message will be printed and the batch job will be signed off unless QUIT=NO has been given. Interactive users are *never* signed off by a mount error, i.e., the QUIT keyword is ignored.

RETRY Keyword

The RETRY keyword parameter may be used to specify the number of times the system will retry a read operation following a data check (see Appendix B). The actual number of retries is 4 times the value specified. The default value is 10 (40 retries). The RETRY parameter may also be changed after the tape has been mounted (see the section "Control Commands").

TRANSLATE Keyword

The TRANSLATE keyword parameter specifies the character translation scheme to be used for ANSI-labeled or ASCII unlabeled tapes. The keyword is given in the form

TRANSLATE={IBM | MTS(parity) | MTS | NONE}

where "parity" may be EVEN, ODD, ZERO, or ONE. The default is MTS(ZERO).

TRANSLATE=MTS(EVEN) specifies that characters are translated according to the MTS translate tables ASCEBC and EBCASC (see the descriptions of the ASCEBC and EBCASC subroutines in this volume). On output, even-parity output characters are produced, that is, characters with the

high-order parity set to either one or zero such that the number of one-bits in each eight-bit character is even.

TRANSLATE=MTS(ODD) specifies that characters are translated according to the MTS translate tables. On output, odd-parity output characters are produced, that is, characters with the high-order parity set to either one or zero such that the number of one-bits in each eight-bit character is odd.

TRANSLATE=MTS(ZERO) specifies that characters are translated according to the MTS translate tables. On output, ASCII characters with the high-order parity bit set to zero are produced.

TRANSLATE=MTS(ONE) specifies that characters are translated according to the MTS translate tables. On output, ASCII characters with the high-order parity bit set to one are produced.

TRANSLATE=MTS is the same as TRANSLATE=MTS(ZERO).

TRANSLATE=IBM specifies that characters are translated according to the IBM translate tables IASCEBC and IEBCASC (see the descriptions of the ASCEBC and EBCASC subroutines in this volume). This translation corresponds to that used in IBM operating systems for generating ASCII 9-track magnetic tapes.

TRANSLATE=NONE specifies that no automatic character translation is done either at input or output. ANSI labels are still translated if label processing is enabled. This is an alternative to specifying the binary @BIN I/O modifier on input or output operations.

File Protection Keywords (RING and WRITE)

Either the RING or WRITE keyword parameter may be used to specify the placement of the file-protect ring in the tape reel. The placement of the ring can be changed *only* by dismounting and remounting the tape. The default condition is RING=OUT (or WRITE=NO) except for pool tapes, which are always mounted with the ring in.

WAIT Keyword

The WAIT keyword parameter specifies the action to be taken in the case that a tape drive of the required type is currently unavailable. If WAIT=YES is specified, the mount request is queued automatically. If WAIT=PROMPT is specified (the default), the user is asked whether or not to queue the mount the request. If WAIT=NO is specified, the mount request is not queued but a message is printed indicating that the required tape drive is currently unavailable. This option is effective only in terminal mode.

TIMER Keyword

The TIMER keyword controls the inactive-tape warning timer which prints a warning message whenever the tape has not been used for a specified interval of time. The timer is ON by default, with a timer interval of 15 minutes. The timer interval can be changed by specifying TIMER=n, in which case the timer interval is set to "n" minutes. The timer can be disabled completely by specifying TIMER=OFF or TIMER=0.

Mount Examples

(1) Assume that the program TRANSFORM reads 100-byte input data records from logical I/O unit 5 and prints the results on unit 6. The input data for this particular run is in the third file of the 9-track, unlabeled tape with tape name DAY26. The output from unit 6 is to be written on the 9-track, labeled tape with tape name UTILITY and volume name UTIL.

```
$MOUNT
DAY26 *IN* POSN=*3* FMT=FB(12000,100)
UTILITY *OUT* RING=IN FMT=VB(15000,133) POSN=*EOT* -
   NAME=PASS3OUT EXPDT=04-01-84
$ENDFILE
$RUN TRANSFORM 5=*IN* 6=*OUT*
```

The input file is blocked with format FB, SIZE of 12000, and LRECL of 100; the output is to be format VB with a maximum block size of 15000 and a maximum LRECL of 133. The parameter RING=IN is given for the output tape so that it will be possible to write on it. The POSN=*EOT* keyword causes the output tape to be positioned to its logical end, i.e., after the last file currently on the tape. Had the POSN keyword not been given, the new file would have been written at the beginning of the tape, and any data currently on the tape would have been lost. The NAME keyword indicates that the new file is to be named PASS3OUT; the EXPDT keyword indicates that the new file is to expire on 1 April 1984.

(2) The output from the program in the preceding example could be printed at a later time as follows:

```
$MOUNT UTILITY *TAPE* POSN=PASS3OUT
$COPY *TAPE* *PRINT*
```

In this example, it is not necessary to specify the blocking parameters for the file since they are automatically obtained from the file labels.

(3) In this example, the *SORT program is used to sort data which is stored on the blocked, labeled tape with tape name MYDATA and volume name MYDATA. The sorted output is to be saved on the labeled tape with tape name SORTOUT and volume name SAVE. The system tape-blocking facilities are disabled for both tapes because it is more efficient to allow *SORT to do the blocking. The input data is in the file SEPT on tape MYDATA, and the output data is to be written into a new file named SEPT.SORT on tape SORTOUT.

```
$MOUNT
MYDATA *INPUT* POSN=SEPT BLK=OFF
SORTOUT *OUTPUT* WRITE=YES NAME=SEPT.SORT INIT=YES -
   FMT=FB(12000,60) BLK=OFF
$ENDFILE
$RUN *SORT
SORT=CH,A,5,30 INPUT=*INPUT*,FB,60,2400
OUTPUT=*OUTPUT*,FB,60,12000 REC=11500
```

On the output tape, the INIT=YES keyword is given to “empty” the tape. The new file SEPT.SORT will be the first file on the tape. Even though blocking has been disabled for the output tape, the blocking parameters are specified in the mount request so that the labels for the file SEPT.SORT will reflect the way in which *SORT has done the blocking.

Note that the output blocking specification on the *SORT control statement specifies exactly the same blocking parameters as were specified in the mount request. For a complete description of the *SORT program, see *MTS Volume 5: System Services*.

(4) This example shows how to mount a pool tape. For pool tapes, RING=IN is the default, so this parameter need not be specified. Pool tapes are labeled, but the volume name (VOLUME keyword) must be omitted from the mount request. (The volume name will be supplied by the system operator.) Pool tapes are always initialized as they are mounted.

```
$MOUNT POOL *P*
```

The tape will be 9-track, 6250 BPI. To mount a 1600 BPI pool tape, the user should add MODE=1600 to the request. To mount a cartridge pool tape, the user should enter:

```
$MOUNT POOL CTP *P*
```

DISMOUNTING MAGNETIC TAPES

In *dismounting* a tape, the following steps are taken by the system:

(1) if the most recent operation was a write, the tape is *terminated* with several tape marks;

(2) the tape is rewound; and

(3) the tape drive is released, thus ending the elapsed-time charge for its use.

When the user signs off, all tapes are automatically dismounted. However, the user may finish using a tape before being ready to sign off. In this case, the $RELEASE command may be used to dismount the tape (see *MTS Volume 1: The Michigan Terminal System*). The DISMOUNT system subroutine may be used by a program to dismount tapes.

For example, the following command will dismount the tape whose pseudodevice name is *INPUT*:

$RELEASE *INPUT*

After the tape has been dismounted, the system will print a message of the form

```
Txxx Released
```

The absence of this message indicates that the system has not released the tape but is keeping the tape available for a program. When the program using the tape is terminated (unloaded), the tape will then be released.

REMOUNTING MAGNETIC TAPES

Sometimes, the user will want to dismount a tape and either mount a different tape in its place or remount the same tape after inserting or removing the file-protect ring. In either case, the user will want to dismount the old tape and immediately mount a new tape (or perhaps the same tape) *without releasing* the tape drive. This can be done by giving a new mount request in which the tape PDN is the *same* as the PDN of the old tape. The system will then dismount the old tape and ask the operator to

mount the new tape (or the same tape) on the same drive. The obvious restriction here is that the mode of the new tape must be compatible with the tape drive. See the description of the MODE keyword parameter in the section "Mounting Magnetic Tapes."

Whenever a tape PDN is reused in this manner, the values of the following parameters will remain unchanged unless explicitly changed by mount keyword parameters: BLOCKING, NAME, DTCHK, EXPDT, FMT, LP, LRECL, MODE, RETRY, MINSIZE, SIZE, TRANSLATE, and BLKPFX.

CONTROL COMMANDS

After a tape has been mounted, the MTS $CONTROL command and the CONTROL system subroutine may be used to position the tape and to specify any of the several parameters which control blocking, label processing, error recovery, and mode. The general form of the $CONTROL command is:

$CONTROL *pdn* command

Here, *pdn* is the pseudodevice name of a currently mounted tape and "command" is a *control command* specifying an operation to be performed on the tape.

For example, the following command will rewind the tape with pseudodevice name *T*:

$CONTROL *T* REW

Programs may call the system subroutine CONTROL to perform any control operation. FORTRAN users should call CNTRL; PL/I-(F) users should call CNTL.

For example, the following FORTRAN sequence will position the tape attached to logical I/O unit 12 to the beginning of the 10*th* file:

```
INTEGER*2 LEN
LEN=9
CALL CNTRL('POSN=*10*',LEN,12)
```

Here, POSN=*10* is the control command to be executed and LEN is a halfword integer giving the number of characters in the control command. In this example, no attempt is made to recover from any errors which may result from the execution of the POSN command. For details on the calling sequence and error recovery, see the CONTROL subroutine description in this volume.

Only one tape control command may be issued on each MTS $CONTROL command or call to the CONTROL subroutine.

Return Codes

Every input, output, and control operation signals a *return code* (RC) on completion. Return codes are always zero or positive multiples of 4. Appendix B includes an explanation of all magnetic tape return codes.

When an operation finishes normally (without any error or exceptional condition), the return code will be zero. On the other hand, when an exceptional condition occurs, the return code will be nonzero. For example, RC 4 on an input operation indicates an end-of-file condition. When an error condition

occurs, a higher RC will be signaled along with a descriptive error message. A user program must explicitly indicate to the system that it wishes to continue executing after an error condition occurs on any I/O operation. On a control operation (i.e., a call to the CONTROL subroutine) no such explicit indication is required, but MTS will still signal the return code and make the error message available to the user program.

Control Command Summary

The following table summarizes all of the available control commands for magnetic tapes. Each command is explained in detail in the subsequent text. The commands have been divided into functional groups related to positioning, blocking, label processing, error recovery, and miscellaneous operations. Wherever a command parameter is shown, it may be separated from the command name by zero or more blanks or by a single equal sign.

Control Command	*Function*
Positioning	
REW	Rewind
FSR [n]	Forward space "n" records ($0 \le n \le 32767$)
BSR [n]	Backspace "n" records ($0 \le n \le 32767$)
FSF [n]	Forward space "n" files ($0 \le n \le 32767$)
BSF [n]	Backspace "n" files ($0 \le n \le 32767$)
POSN={*n* | * | *EOT* | name}	Position to n*th* file, current file, end-of-tape, or file name, respectively
Blocking	
{FORMAT | FMT | RECFM}=fmt[([size][,lrecl])]	Specify blocking format and, optionally, block size and/or logical record length
{SIZE | BLKSIZE}=n	Specify block size ($18 \le n \le 32767$)
LRECL=n	Specify logical record length ($1 \le n \le 32767$)
BLK={ON | OFF}	Enable or disable blocking
BLKPFX=n	Specify ANSI block-prefix length for ANSI labeled tapes ($0 \le n \le 99$)
Label Processing	
{NAME | DSN}=[name]	Specify name for next new file

DTCHK={ON | OFF}	Enable or disable expiration date checking
EXPDT[={mm-dd-yy | mm/dd/yy}]	Specify expiration date for new files
INIT	Initialize (empty) a labeled tape
LP={ON | OFF}	Enable or disable label processing
EOV	Terminate tape with end-of-volume labels
CC={A | M}	Specify nonblank control character for next new file written

Error Recovery

RETRY=n	Specify read error retry count ($0 \le n \le 15$)
SNS	Return sense data

Miscellaneous

{WTM | EOF} [n]	Write "n" tape marks ($0 \le n \le 32767$)
MODE=mode	Specify tape mode
PUSH	Push current tape parameters onto stack
POP	Pop tape parameters from stack
MINSIZE=n	Specify the minimum length block which can be written ($1 \le n \le 100$)
TIMER={ON | OFF | n}	Enable, disable, or specify elapsed-time interval (in minutes) for inactive tape warning message
TRANSLATE={MTS(parity) | MTS | IBM | NONE}	Specify translation scheme for ANSI labeled or ASCII unlabeled tapes; parity is EVEN, ODD, ZERO, or ONE

Positioning Commands

REW This command rewinds a tape to the load point (the beginning of the first file on the tape).

Example: `$CON *TAPE* REW`

FSR The FSR (forward-space record) command forward spaces the tape past the next "n" records. If "n" is omitted, it defaults to 1. If the tape is unblocked or if blocking is off (BLK=OFF), the FSR command forward spaces the tape past the next "n" blocks. If the tape is unlabeled, the correct blocking parameters must be specified before the FSR command will work properly. The FSR command will terminate prematurely if a tape mark is passed. In this case, return code 4 (end-of-file) will be given and the tape will be

left in position immediately *after* the tape mark.

Example: `$CON *INPUT* FSR 106`

BSR — The BSR (backspace record) command backspaces the tape past "n" records. If "n" is omitted, it defaults to 1. If the tape is unblocked or if blocking is off (BLK=OFF), the BSR command backspaces the tape past "n" blocks. If the tape is unlabeled, the correct blocking parameters must be specified before the BSR command will work properly. The BSR command will terminate prematurely if a tape mark is passed. In this case, return code 4 (end-of-file) will be signaled and the tape will be positioned immediately *before* the tape mark, i.e., after the last record of the preceding file. The BSR command will also terminate prematurely if the tape is backspaced into the load point. In this case, return code 8 will be signaled and the tape will be left in position at the load point.

Example: `$CON *306* BSR 5`

FSF — The FSF (forward space file) command forward spaces the tape past the next "n" tape marks. If "n" is omitted, it defaults to 1. The FSF command always leaves the tape in position between files, i.e., after the tape mark at the end of one file and before the first block of the next file. For labeled tapes, an attempt to skip beyond the last file on the tape (the logical end of the tape) will cause return code 12 to be signaled.

Example: `$CON *DATA* FSF 3`

BSF — The BSF (backspace file) command backspaces the tape past "n" tape marks. If "n" is omitted, it defaults to 1. The tape will be left in position immediately *before* the n*th* tape mark which was passed. In other words, the tape will be left in position after the last record of a file, but before the tape mark at the end of the file. The BSF command will terminate prematurely if the tape is backspaced into the load point. In this case, return code 8 will be signaled and the tape will be left in position at the load point.

Example: `$CON *OLD* BSF 2`

POSN — The POSN (position) command moves the tape to the beginning of a specified file or to the logical end of the tape. The first form of the command (POSN=*n*) positions the tape to the beginning of the n*th* file, regardless of the current position. The first file on a tape is file 1.

Example: `$CON *QQSV* POSN=*13*`

The second form of the command (POSN=*) positions the tape to the beginning of the current file. If the tape is already at the beginning of the file, it is not moved.

Example: `$CON *T* POSN=*`

The third form of the command (POSN=*EOT*) may be used only on labeled tapes. It positions the tape to a point immediately after the end of the last existing file (i.e., to the logical end of the tape). This form of the command is generally used to position a tape before adding a new file.

Example: `$CON *ABC* POSN=*EOT*`

The fourth form of the command (POSN=name) may be used only on labeled tapes. The tape is searched in a forward direction for a file "name". If the specified file has not been found when the end of the last file is reached, the tape is rewound and the search is continued. If the correct file is found, the tape is left in position at the beginning of that file. Otherwise, error return code 16 is signaled and the tape is left in position at the beginning of the file in which the search began.

Example:
```
$CON *WKSI* POSN=WEST
```

Blocking Commands

For unlabeled tapes, the current blocking parameters may be specified in the mount request. If they are not specified, the default blocking parameters U(32767) will be used. Subsequently, the current blocking parameters will remain in effect until changed by a blocking control command (see below).

For a labeled tape that has no existing files (i.e., a pool tape or a tape that has been freshly initialized), the default blocking parameters are VBS(16384,32767). However, if the tape does have existing files, the current blocking parameters are normally obtained from the file labels when each file is entered, i.e., when the tape is positioned to the beginning of the file (if it is to be read forward) or to the end of the file (if it is to be read backward). Thus, when reading a file, the system will automatically deblock the file according to the blocking parameters in the file labels.

When rewriting an existing file, the system will use the blocking parameters from the labels of the old file, and when adding a new file to the logical end of the tape, the system will use the blocking parameters from the labels of the preceding file. However, the user may override the blocking parameters in the file labels by giving a blocking control command immediately before the file is entered, i.e., when the tape is in position at the beginning of the file. If the current blocking parameters are respecified in this manner, the write operation must start at the *beginning* of the file (it may not add blocks to the end of the file). In this case, the file labels will be rewritten and will include the current blocking parameters. (If the current blocking parameters are respecified prior to a *read* operation, the file labels will not be affected.) In summary, when adding a completely new file to a labeled tape, the user may specify the blocking parameters by giving a blocking control command *after* positioning the tape but before writing the first record.

FORMAT The FORMAT command may be used to specify the current blocking format and optionally, block size and/or logical record length. In other words, this command may be used to specify all of the current blocking parameters. These may be changed only when the tape is positioned between files unless blocking has been disabled (see the BLK control command description). An attempt to change them at any other time will cause a sequence error (RC 32). For unlabeled tapes, the specified blocking parameters will remain in effect until explicitly changed by another blocking control command. When reading labeled tapes, the specified blocking parameters will apply *only* to the next file entered.

Examples:
```
$CON *B* FORMAT=VB
$CON *XYZ* FMT=U(3600)
$CON *Z9* FMT=FB(12000,80)
```

SIZE The SIZE command may be used to specify the current block size. This may be changed only when the tape is positioned between files unless the format is U or blocking has been disabled (see the BLK control command description). An attempt to change the SIZE at

any other time will cause a sequence error (RC 32). For unlabeled tapes, the specified SIZE will remain in effect until explicitly changed by another blocking control command. When reading labeled tapes, the specified SIZE will apply *only* to the next file entered.

Example: `$CON *T1* SIZE=1600`

LRECL The LRECL command may be used to specify the current logical record length. This may be changed only when the tape is positioned between files unless the format is U or blocking has been disabled (see the BLK control command description). An attempt to change the LRECL at any other time will cause a sequence error (RC 32). For unlabeled tapes, the specified LRECL will remain in effect until explicitly changed by another blocking control command. When reading labeled tapes, the specified LRECL will apply *only* to the next file entered.

Example: `$CON *T2* LRECL=68`

BLK The BLK command may be used to enable or disable the system tape blocking facility. By default, blocking is enabled (ON). If blocking is disabled (OFF), the system will not perform any blocking of input or output records regardless of what format is specified, i.e., it will operate as though the blocking format were set to U. This is useful when reading or writing a blocked, labeled tape where the blocking is to be done by a program and not by the system tape-blocking facility. For example, if the program is writing data which *it* has blocked into format FB blocks and if blocking has been disabled, then the system will write each block directly onto the tape without performing any blocking function. If the tape is labeled, however, the system will store the current blocking parameters in the file labels. Thus, even though blocking is done by the program, the correct blocking parameters should be specified before the first block in the file is written. If blocking is disabled *while* a file is being written, any partial block in the system blocking buffer will be written immediately. Blocking may be enabled (ON) only when the tape is positioned between files unless the format is U; otherwise, a sequence error (RC 32) will be signaled.

Example: `$CON *B* BLK=OFF`

BLKPFX The BLKPFX command specifies the length of the block-prefix field (BDW field) preceding variable-length blocks on ANSI labeled tapes. This defaults to the value specified in the ANSI HDR2 label on input, and on output, to the value from the label field of the previous file. If the block-prefix length is not given in a label and not specified by the BLKPFX command, MTS will use a value of zero. If the block-prefix length is 4, MTS will interpret the first four characters as a 2-byte binary number for format V, and as a 4-character ASCII decimal number for format D. On output, MTS will generate such a block descriptor if BLKPFX=4.

Example: `$CON *B* BLKPFX=4`

Label Processing Commands

NAME The NAME (or DSN) command may be used to specify a name for the next new file (i.e., the next complete file) to be written on a labeled tape. If a name is not specified before the first record of the file is written, the system will generate a default name of the form

Vvvvvv.Fnnnn

where "vvvvvv" is the tape volume name and "nnnn" is the four-digit file number. This command affects only the *next* new file. Each subsequent file will have a default name unless a NAME command is issued before the first record is written.

Example: `$CON *60DATA* NAME=SEPTOUTPUT`

DTCHK The DTCHK (date checking) command may be used to enable or disable expiration date checking for labeled tapes. When enabled (ON), the system will not allow a file to be modified prior to its expiration date. If other files follow the one being modified or replaced, they are assumed to expire on the same date, i.e., their expiration dates are *not* checked by the system. When disabled (OFF), expiration dates are not checked and any file may be modified or replaced (provided that the tape was mounted with the file-protect ring in).

Example: `$CON *SAVE* DTCHK=OFF`

EXPDT The EXPDT command may be used to specify the expiration date for all new files to be written on a labeled tape. To be effective for a particular file, this command must be given before the first record is written. Once an expiration date has been specified, it will be used for all new files until another EXPDT command is given. If no expiration date is given and the tape has no existing files, the system will use the date 01-00-00 for all new files (i.e., the files will already have expired). If no expiration date is given and the tape does have existing files, then when an existing file is replaced, the system will use the expiration date from the existing file; when a new file is added to the logical end of the tape, the system will use the expiration date from the preceding file.

Example: `$CON *JSG* EXPDT=03-31-85`

INIT The INIT command may be used to initialize or "empty" a labeled tape (provided that the tape was mounted with the file-protect ring in). A volume label and tape mark are written at the load point, effectively deleting all files currently on the tape. Then, the tape is rewound. Note: if the tape is 9-track and the user has given a MODE command (or the MODE keyword on the mount request), then the INIT command may actually change the density of the tape.

Example: `$CON *ZAP* INIT`

LP The LP command may be used to enable or disable label processsing. For labeled tapes, LP defaults ON (enabled); for unlabeled tapes, it defaults OFF (disabled). When label processing is enabled, the system expects to find the appropriate labels for each existing file and will generate the appropriate labels whenever a new file is written. When disabled, the system assumes nothing about the tape structure, i.e., all blocks are treated as data blocks on input and no labels are generated on output. Label processing may be disabled at any time, except that it may never be disabled for pool tapes. Label processing may be enabled only if the tape is positioned at the beginning of file number (3n+1), where "n" is a positive integer or zero; otherwise, a sequence error (RC 32) is signaled.

Example: `$CON *WORK* LP=OFF`

EOV The EOV (end-of-volume) command is similar to the WTM command. If a file on a labeled tape is currently open, the EOV command will terminate the file. However, the

trailer labels generated will be special end-of-volume trailer labels, EOV1 and EOV2, instead of normal end-of-file trailer labels, EOF1 and EOF2. (EOV trailer labels are used in IBM OS/VS installations, but not in MTS, to indicate that a file is to be continued on another reel of tape.) If no file is currently open (i.e., if the tape is positioned between files or at its logical end), the EOV command will write an empty file with EOV trailer labels. This command is illegal for unlabeled tapes.

Example: `$CON *JH* EOV`

CC — The CC command may be used to set the control-character field in the HDR2 and EOF2/EOV2 labels. The control-character field is used at OS/VS installations, but not at MTS installations, to indicate the type of carriage-control characters used. The control-character values are:

A	uses ASCII control characters
M	uses machine control characters
blank	uses no control characters

MTS normally writes a blank in this field (byte 37) unless the CC command is given before the first record of the file is written. This keyword affects only the *next* new file written.

Example: `$CON *B* CC=A`

Error Recovery Commands

RETRY — The RETRY command may be used to specify the number of times the system will retry a read operation following a data check (see Appendix B). For 9-track tapes, the actual number of retries is 4 times the value specified in this command. The value recommended for normal use is 10 (40 retries for 9-track). This command may be issued at any time.

Example: `$CON *TEST* RETRY=0`

SNS — The SNS (sense) command returns 72 or more bytes of *sense information* about the tape. The SNS command may be used only with the CONTROL subroutine, not with the $CONTROL command. (Note: the $DISPLAY command can be used to print the sense information.) To give the SNS command, FORTRAN users should call CNTRL, and PL/I-(F) users should call CNTL. The calling program must provide an information area (the first argument of the call to CONTROL) which has "SNS" as the first three characters and is at least as long as the value specified by the second argument, but never less than 72 bytes long. The length (the second argument) determines how many bytes of information the system will return in the information area. This argument should never have a value less than 72. The format of the sense information is given in Appendix B.

Examples:

```
      INTEGER*2 SNSLEN/78/
      INTEGER INFO(20)/'SNS'/
C     GET SENSE INFORMATION FROM TAPE ON UNIT 15
      CALL CNTRL(INFO,SNSLEN,15)

$DISPLAY *T*
```

Miscellaneous Commands

WTM — The WTM (write tape mark) command writes "n" tape marks (end-of-files) on the tape. If "n" is omitted, it defaults to 1. If the operation immediately preceding the WTM command was a write, then the system will write any partial block in the system blocking buffer before writing the "n" tape marks. Also, the system will automatically write any partial block followed by several tape marks whenever a write (or WTM) operation is followed by a REW, POSN, BSR, BSF, dismount, or signoff. In this way, the system makes sure that the tape is always properly terminated, even if, for example, a batch job exceeds its execution time limit while writing a tape.

Example: `$CON *T13* WTM`

MODE — The MODE command may be used to specify the tape density (see the description of the MODE keyword in the section "Mounting Magnetic Tapes"). An invalid mode, that is, one that is not available on the tape unit, will cause a mode error (RC 40). The entire tape must be of uniform density. Thus, following a MODE command, the mode will not actually be changed until the tape is reinitialized or relabeled (for labeled tapes) or until the first block is rewritten (for unlabeled tapes).

Example: `$CON *P* MODE=800`

PUSH — The PUSH command causes the system to save the following parameter values on a last-in-first-out stack associated with the tape: the current values of LP, DTCHK, BLK, FORMAT, SIZE, LRECL, MODE, NAME, EXPDT, CC, LBLTYPE, BLKPFX, and RETRY. These values may be restored to active use by giving the POP command. Several system programs such as *TAPEDUMP use the PUSH command to save the status of a tape before modifying the blocking parameters. The original parameters are restored by giving the POP command.

Example: `$CON *MT* PUSH`

POP — The POP command causes the system to retrieve the parameter values saved by the most recent PUSH command. However, certain conditions must exist in order for the POP command to succeed. Restoration of the BLK status (enabled or disabled) is subject to the same conditions as execution of the BLK command. Likewise, restoration of the LP status (enabled or disabled) is subject to the same conditions as execution of the LP command. If any of these conditions is violated, *none* of the values is restored, i.e., the POP command does nothing. A sequence error (RC 32) is signaled if the last-in-first-out stack for the tape is empty.

Example: `$CON *TAP* POP`

MINSIZE — The MINSIZE command specifies the minimum length for blocks to be written. Any block with fewer bytes than MINSIZE will be padded with blanks. The default MINSIZE is 18 bytes. This minimum is generally recommended by the hardware manufacturers.

Example: `$CON *JSG* MINSIZE=1`

TIMER — The TIMER command controls the inactive-tape warning timer which prints a warning message whenever the tape has not been used for a specified interval of time. The timer

is ON by default, with a timer interval of 15 minutes. The timer interval can be changed by giving the control command TIMER=n, in which case the timer interval is set to "n" minutes. The timer can be disabled completely by giving the control command TIMER=OFF or TIMER=0.

TRANSLATE

The TRANSLATE command specifies the character translation scheme to be used for ANSI-labeled or ASCII unlabeled tapes. The keyword is given in the form

TRANSLATE={IBM | MTS(parity) | MTS | NONE}

where "parity" may be EVEN, ODD, ZERO, or ONE. The default is MTS(ZER0). This command is allowed only when the tape is positioned between files. See the description of the TRANSLATE keyword parameter for the $MOUNT command for complete details.

Control Command Examples

(1) The following example shows how to *add* additional data to the end of the last file on a tape. In this example, the tape is unlabeled and the last file is file 9.

```
$MOUNT DSAVE *T* RING=IN
$CON *T* POSN=*10*
$CON *T* FMT=FB(4200,60)
$CON *T* BSF
$COPY DATA *T*
```

The tape is positioned immediately after file 9 by the POSN=*10* command. Then, the blocking parameters for file 9 are established by the FMT command. The BSF command backspaces the tape past the tape mark at the end of file 9, and then the $COPY command adds new data to the end of the file. At this point, the file has not been terminated, i.e., no tape mark has been written. Thus, the following command adds still more data to the end of the file:

```
$RUN GENERATE 6=*T*
```

The following sequence of commands lists the last 3 records written and then adds additional data from the file SAVE and a tape mark:

```
$CON *T* BSR 3
$LIST *T*
$CON *T* BSF
$COPY SAVE *T*
$CON *T* WTM
```

When the BSR 3 command is given, the system first terminates the tape, then backspaces it 3 logical records. The $LIST command lists the last 3 records of the tape and stops after passing over the tape mark. The BSF command then backspaces the tape past the tape mark into file 9. Then, the $COPY command adds still more data to the end of the file. The WTM command terminates the file with a tape mark. This last command would be unnecessary if the next command were a REW, POSN, BSR, BSF, dismount, or signoff.

(2) A very detailed examination of the use of the FSR and FSF commands is given in the following example:

```
$MOUNT CSAVE *N* VOL=IN
$CON *N* FSR 300
$LIST *N*(1,100)
$CON *N* FSF
$LIST *N*(1,10)
```

The FSR 300 command forward spaces the tape past the first 300 logical records of the first file. Blocking is handled automatically because the tape is labeled, i.e., the blocking parameters are obtained from the file labels. If the first file contained fewer than 300 records, the FSR command would terminate after passing over the tape mark. The $LIST command lists the next 100 records (either continuing in the first file or beginning in the second file). Due to the manner in which MTS processes line-number ranges with devices, whenever an explicit line-number range is exceeded, one *extra* record is always read. When the extra record (in this example, the 101st record) is read, an end-of-file (RC 4) is signaled. If a tape mark is passed before the line-number range is exceeded, an end-of-file is signaled, but no additional record is read.

If the first file contained more than 400 records, the FSF command would skip the remainder of the file; if it contained 300 to 400 records, the $LIST command would have read the remainder of the first file so that the FSF command would skip the second file; if the first file contained fewer than 300 records, the $LIST command would have listed either some of the second file (if it contained more than 100 records) or all of it so that the FSF command would skip either the remainder of the second file or all of the third file. Therefore, the second $LIST command will list the first 10 records of either the second, third, or fourth file, leaving the tape positioned after the 11*th* record or, if there are fewer than 11 records, after the end of the file.

(3) In the following example, two new files, PA54 and PA55, are added to tape BSAVE:

```
$MOUNT BSAVE *N* WRITE=YES VOL=PA
$CON *N* POSN=*EOT*
$CON *N* NAME=PA54
$CON *N* FMT=VB(10000,255)
$COPY DT4 *N*
$CON *N* WTM
$CON *N* NAME=PA55
$COPY DT5 *N*
$LIST *N*(3,1)@BKWD
$CON *N* REW
```

Both files are to be blocked in the same way thus it is not necessary to respecify the blocking parameters for the file PA55. The last three records of the last file are listed (in reverse order) by using an inverted line-number range and the @BKWD (backward) modifier.

(4) In the following example, an existing, 1600 BPI labeled tape is converted to 6250 BPI. It is assumed that the tape is certified by the manufacturer for use at 6250 BPI.

```
$MOUNT
ASAVE *T* WRITE=YES MODE=6250 VOL=JH
POOL *P*
$ENDFILE
$RUN *TAPECOPY 0=*T* 1=*P*
$CON *T* INIT
$RUN *TAPECOPY 0=*P* 1=*T*
```

APPENDIX A
CARING FOR MAGNETIC TAPES

The reliability of a tape can be preserved through proper care. Mishandling may result in damage to the tape and loss of some of the data. When a tape drive is unable to read a portion of the tape, it will signal a *permanent read error* (RC 16). The tape drive may sometimes be responsible for the error, but more often, it is the tape that is at fault. The user must then take steps to recover what is left of the data (see Appendix B).

The following is a list of guidelines for handling tapes:

(1) Use only high quality tapes.

Buy only tapes that are fully surface-tested.

(2) Maintain backup copies of important tapes.

When a tape has been damaged, it is much easier to recover data from a backup copy than from the tape itself. Backup copies may be stored at the same location as the original copies, but if additional security is required, the backup copies should be stored elsewhere.

(3) Replace old tapes.

Tape emulsions tend to deteriorate with time. After three to five years, the contents of a tape should be copied to a new tape.

Tapes may deteriorate through insufficient use. A tape should be unwound and rewound at least once a year.

(4) Replace tapes before they wear out.

If a tape is used frequently (for example, several times a day), its useful life may be only a few months. The tape should be replaced before permanent read/write errors occur. The following command sequence may be used to test the tape for incipient read errors:

```
$CONTROL *T* RETRY=0
$RUN *TAPECOPY 0=*T* 1=*DUMMY* PAR=COPYERRORS
```

The RETRY=n control command determines the number of times the system will retry a read operation when it senses an error. The default is RETRY=10. When a lower RETRY value is used, any incipient read errors are revealed; an error comment (indicating the file and record in error) is printed on SERCOM by the *TAPECOPY program. The printing of error comments is enabled by the COPYERRORS parameter.

(5) Protect tapes against extreme environmental conditions.

Tapes should not be exposed to dust or dirt or to extremes of temperature or humidity. When not in use, tapes should be kept in closed tape containers. After transporting a tape through a cold environment, the tape container should not be opened until the tape has come to room temperature. Otherwise, moisture may condense on the tape.

Tapes should be protected against mechanical damage. The edges of a tape can be damaged by pressure on the sides of the tape reel. If the sides of the tape reel have openings, careless handling may damage the tape. For shipping, the tape should be sealed in a tape container and placed in a double-thickness cardboard box.

APPENDIX B
MAGNETIC TAPE ERRORS

Many of the errors that occur with tapes are recognized by the system and reported to the user. Whenever possible, error messages are specific and point directly to the cause. Sometimes, however, error messages are less specific and may have a variety of possible causes: a bad tape, a malfunctioning tape drive, a problem with the system, or a problem with the user program or command.

Not all of the mistakes made by the user will be caught by the system. Sometimes, the user will give commands that are legal but that produce unintended results. For example, the user may accidentally overwrite a tape and destroy some or all of the existing data (see the section "Tape Accidentally Overwritten" below).

Mount Request Errors

Every tape mount request is scanned by the system for errors in syntax. When the system finds such an error, it will print one of the following error messages:

Tape name was not given.
Device type was not specified.
Pseudodevice name was not given.
Invalid pseudodevice name.
Pseudodevice name too long.
Invalid device type.
Invalid tape name.
Invalid block size.
Invalid logical record length.
Invalid keyword "xxx".
Invalid expiration date.
Invalid data set name.
"xxx" has invalid syntax.
Missing required prime field.

The "prime field" is the external tape ID, enclosed in primes. The rest of the messages are self-explanatory. The system also checks for semantic errors, possibly printing one of the following error messages:

Read access not allowed to tape.
Write access not allowed to tape (cannot mount tape with RING=IN).
INIT=YES valid only for labeled tape with RING=IN.
MODE=xxx is inconsistent with device type.
MODE=xxx is not available on device "yyy".
Not enough devices available to satisfy this request.
Pseudodevice name already requested for "xxx".
Pseudodevice name is in use by MTS.

Pseudodevice name is already in use for device type “xxx”.

After a mount request has been determined to be free of syntax and semantic errors, it is displayed on the operator console in the machine room of the Computing Center. At this time, the operator may determine that, for some reason, the tape cannot be mounted; one of the following error messages will be printed:

System in unattended mode; no mounts allowed at this time.
Mounts are temporarily disabled; try again later.
All units busy at this time.
Aborted by operator (reason given).
Not available (reason given).
Aborted (by user attention interrupt).
Aborted (due to error in another request).

If any tape-identifying parameters were given incorrectly in the mount request, one of the following error messages will be printed:

Incorrect tape name or tape ID.
Volume “xxx” is incorrect.
Tape is not of specified mode.
Volume name not given for labeled tape.

These messages are self-explanatory. If the tape drive is unable to read the first block on the tape, the following error message will be printed:

Permanent I/O error on first tape block.

This usually indicates that the beginning of the tape is damaged. In this case, the user should seek the assistance of a consultant.

If any of the above errors occur, the tape will not be mounted. In batch mode, any errors will normally cause the job to be signed off. However, this automatic sign-off can be inhibited by giving the QUIT=NO keyword option in the mount request.

Return Codes

The system uses *return codes* to indicate the outcome of magnetic-tape I/O or control operations. If an error occurs in a control command, for example, the system will print a message of the form,

```
CONTROL COMMAND ABORTED -- ERROR nnnn
```

where “nnnn” is the return code. In some cases, the system will also print a more specific message describing the error.

The following list gives the meaning of each magnetic-tape return code. The nonzero return codes marked with an asterisk are fatal errors. The other nonzero return codes are “exceptional”, but not fatal errors.

Input:	0	Successful return
	4	Tape-mark (end-of-file) sensed on read, read backward, BSR, or FSR operation
	8	Load point reached on read backward, BSR, or BSF operation
	12*	Logical end of labeled tape reached on read, FSR, or FSF operation
	16*	Permanent read error, invalid control command, invalid control command parameter, or file not found on POSN operation
	20*	Should not occur
	24*	Fatal error (may be due to tape unit malfunction, label error in which the position of the tape is uncertain, or pulling the tape off the end of the reel during a read, FSR, or FSF operation); following a fatal error, the tape must be rewound before any other I/O operation is allowed
	28*	Volume or file (data set) label in error
	32*	Sequence error (may be caused by issuing a control command when the tape is not positioned properly, or by a read, FSR, or FSF operation following a write operation)
	36*	Deblocking error caused by improper blocking parameters, e.g., attempting to deblock a format FB file using a format VB specification
	40*	A mode command (or the current mode) is not compatible with the tape unit on which the tape is mounted
Output:	0	Successful return
	4	End-of-tape marker sensed during write or WTM operation, i.e., the tape is full
	8	Load point reached on read backward, BSR, or BSF operation
	12*	Attempt to write more than 5 additional records after end-of-tape marker sensed
	16*	Permanent write error, invalid control command, or invalid control command parameter
	20*	Attempt to write on file-protected tape or unexpired file
	24*	Fatal error (may be due to tape unit malfunction, label error in which the position of the tape is uncertain, or pulling the tape off the end of the reel during a read, FSR, or FSF operation); following a fatal error, the tape must be rewound before any other I/O operation is allowed
	28*	Volume or file (data set) label in error
	32*	Sequence error (may be caused by issuing a control command when the tape is not positioned properly, or by a read, FSR, or FSF operation following a write operation)
	36*	Blocking error caused by improper blocking parameters or parameters which are inconsistent with the labels of the file being written
	40*	A mode command (or the current mode) is not compatible with the tape unit on which the tape is mounted
Control:	0	Successful return
	4	End-of-file (BSR or FSR) or end-of-tape (FSF)
	8	Unit check
	12	End-of-tape
	16	Invalid CONTROL command or parameter, file not found (POSN), or permanent read/write error
	20	Attempt to write on unexpired file or without ring
	24	Fatal error
	28	Invalid volume, header, or trailer label

32	Invalid I/O region or mode/blocking error
36	Invalid blocking parameter
40	Invalid mode
44	Access not allowed

Permanent Read Errors (RC 16)

When reading a block from a tape, the tape drive will signal a *data check* whenever it detects an incorrect parity. The system will then make several attempts to reread the block. One of these attempts will often be successful.

The RETRY=n control command sets a limit on the number of times the system will retry a read operation. For 9-track tapes, the actual limit is 4 times "n". For example, the command

```
$CONTROL *T* RETRY=5
```

sets a limit of 20 retries if the tape is 9-track, 5 retries if it is a cartridge. The default is RETRY=10.

If all of the retries fail, the system gives up and signals a *permanent read error* (RC 16). The tape is left in position after the offending block (or before the offending block on a read backward). Permanent read errors have several possible causes, some of which are given in the following list:

(1) The tape was written at a density greater than the manufacturer's specification.

(2) The tape is physically damaged as a result of mishandling, wear, or aging.

(3) The tape surface is dirty.

(4) The tape drive on which the tape is being read is improperly aligned or is malfunctioning in some other way.

(5) The tape drive on which the tape was written was improperly aligned.

(6) The tape was improperly terminated due to a system crash that occurred while the tape was being written. This problem is discussed in more detail in the section "Tape Improperly Terminated" below.

If there is reason to believe that nothing is wrong with the tape, the user may dismount it and ask the operator to mount it on a different drive. If a permanent read error still occurs, the probability is very great that the tape is responsible. (It is not necessary to try all of the tape drives.)

If something *is* wrong with the tape, the user should switch to a backup copy (if there is one). Otherwise, the user will have to retrieve what is left of the data on the tape.

Tapes can be tested for incipient read errors by giving the following commands:

```
$CONTROL *T* RETRY=0
$RUN *TAPECOPY 0=*T* 1=*DUMMY* PAR=COPYERRORS
```

Lowering the RETRY count also lowers the threshold for read errors. In this way, it is possible to spot read errors and replace the tape (if the tape is responsible) before the read errors become more serious.

The user should note that reliability increases with tape density. For 9-track tapes, 1600 BPI is somewhat less reliable than 6250 BPI.

Permanent Write Errors (RC 16)

After writing a block on a tape, the tape drive will read the block and perform some parity checking. If the tape drive detects an error, the system will backspace the tape, erase the block (leaving a longer than normal IBG), and attempt to rewrite the block. If this procedure fails after 15 retries, the system will give up and signal a *permanent write error* (RC 16). Such an error may be caused by one or more of the following:

(1) The tape is being written at a density that exceeds the manufacturer's specification.

(2) The tape is defective over a length of several feet.

(3) The tape drive is malfunctioning.

Tape Runs Off Reel

If a tape runs off the reel during a write, read, FSR, or FSF operation, the system will print the message:

```
FATAL ERROR (OFF END OF REEL)
```

The following are some possible causes for a tape running off the reel:

(1) When writing a tape, RC 4 indicates that the EOT (end-of-tape) marker has been reached. If this warning is ignored and more records are written, the tape may run off the reel.

(2) With unlabeled tapes, it is possible to read or forward space past the tape marks that terminate the tape. If this is done, the tape may run off the reel.

(3) The tape may be improperly terminated due to a system crash that occurred while the tape was being written. If the tape is positioned past the end of the data, it may run off the reel.

If the tape runs off the reel, the operator will rethread the tape through the tape drive. However, because the position of the tape will now be uncertain, the user will have to rewind the tape before proceeding with any other operation.

Tape Improperly Terminated

Following any write operation, the system will terminate the tape with several tape marks before repositioning the tape. If, however, the system crashes during a write operation, it will leave the tape improperly terminated. The easiest way to remedy this condition is to rewrite the last new file on the tape.

If this condition is not corrected, several errors may result:

(1) If the tape is positioned past the end of the new data, it may run off the reel.

(2) There may be some "junk" and/or old data following the end of the new data. An attempt to read this area may cause a permanent read error (RC 16) or a labeling error (RC 28).

Tape Accidentally Overwritten

If a tape is accidentally overwritten, some or all of the existing data will be lost. It is easier to *prevent* such a mistake than to recover from it. The following precautions will help to protect the existing data on a tape:

(1) Do not mount a tape with WRITE=YES (RING=IN) unless data is to be written.

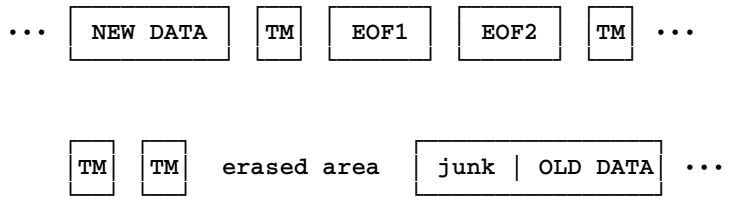
(2) When writing the first file on a labeled tape, assign an appropriate expiration date.

(3) Before writing new data on a tape, be sure of the position of the tape. The current position can easily be checked by using the $DISPLAY command, for example:

```
$DISPLAY *T*
```

(4) Before writing new data on a tape, make sure there is a backup copy of the old data on the tape.

If a tape *is* accidentally overwritten and there is no backup copy, the user must proceed to recover whatever is left of the old data. The new file will be followed by trailer labels (if the tape is labeled) and several tape marks to indicate the new logical end of the tape. Following this, there will be an erased area, then some unreadable "junk", then the remainder of the old data:



If the overwritten tape is labeled, the *TAPEFIXER program may be used to retrieve the remainder of the old data and copy it to a second tape. This program is interactive and must be run from a terminal. Before attempting to run *TAPEFIXER, the user should read the program description in this volume. The user should also refer to the above diagram and keep in mind that the area marked "old data" may include file labels (in full or in part) as well as data blocks.

If the overwritten tape is unlabeled, the user might proceed as follows:

(1) Position the tape to the end of the new data.

(2) Give a BLK=OFF control command.

(3) Position the tape forward. The safest way to do this is to repeatedly give a command of the form:

$CON *pdn* FSR

At first, this command will produce a series of end-of-file messages while advancing the

tape past the tape marks following the new data. Next, the command may produce permanent read errors while attempting to read past the "junk" following the tape marks. When the command stops producing permanent read errors, the tape should be in position at a good block in the old data.

(4) Give commands to copy the remainder of the old data to a disk file or to a second tape.

Sense Information

Sense information about a tape can be obtained by giving the $DISPLAY command, for example

```
$DISPLAY *T*
```

will print the sense information for tape *T* as follows:

```
*T* on T904(9TP): Rack=C9999
Vol=MYDATA, Lbltype=OS/VS, Owner=UNIVOFMICH
DSN=DATA1, Credt=01-01-80, Expdt=12-31-90
File=1, Block=0, Record=0, Fmt=U(255,255)
Mode=160(C3), Bpi=1600, Bpmm=63.0, Retry=10
 Statistics (Read,Write,Control):
   Total tape operations (CCWs): 11,1,29
   Recovered errors: 0,0,0  Fatal errors: 0,0,0
Sense=01:004000040040 2C0000080000 000A8C372D92 001192000080
Status:  BLK ON, positioned at load point, LP ON, EXPDT
         given, date checking
```

The sense field in the example above is the hardware sense data, 24 bytes (see below).

Sense information can also be obtained by calling the CONTROL subroutine specifying the SNS control command. The SNS control command may be used only with the CONTROL subroutine, not with the $CONTROL command. The SNS control command returns to the calling program 72 or more bytes of sense information about the tape. The calling program must provide an information area (the first argument of the call to CONTROL) which has "SNS" as the first three characters and is at least as long as the value specified by the second argument, but never less than 72 bytes long. The length (the second argument) determines how many bytes of information the system will return in the information area. This argument should never have a value less than 72. The format of the sense information is shown in the following table.

Bytes	*Function*
0-2	"SNS" (supplied by calling program)
3	Hardware modeset command (see the MODE keyword description in the section "Mounting Magnetic Tapes")
4-7	Tape drive name (4 characters)
8-11	Tape drive type ("9TP " or "3480")
12-13	Block size (2-byte integer)
14	Sense flag (see Note 1 below)
15-20	Hardware sense data, first 6 bytes (see Note 2 below)
21-23	Blocking format (3 characters)
24-25	Logical record length (2-byte integer)
26	Status (see Note 3 below)
27	Retry count (1-byte integer)
28-29	File number (2-byte integer)

30-31	Block number within file (2-byte unsigned integer)
32-37	Volume name (6 characters)
38-54	File (data set) name (17 characters)
55-57	Mode (first 3 characters)
58	Status (see Note 3 below)
59	Control character ("A", "M", or blank)
60-63	Record number within file (4-byte integer)
64-71	Expiration date (MM-DD-YY)
72-77	Rack Number
78-82	Density in bits/inch (5 characters, right-justified)
83-88	Density in bits/mm (6 characters, right-justified)
89-106	Hardware sense data, last 18 bytes
107-114	Creation date (MM-DD-YY)
115-122	UserID (left-justified)
123-130	Batch receipt number (left-justified)
131-140	OwnerID (from volume label)
141-146	Label type (OS/VS, VLO, or ANSI)
147	ANSI block prefix (1-byte, unsigned integer)
148-171	Tape-use statistics (see Note (6) below)
172-180	Translation scheme for ANSI labeled tapes (9 characters, left-justified)

Notes:

(1) The low-order 2 bits of the *sense flag* (byte 14) are coded to indicate the validity of the hardware sense data (bytes 15-20) and the high-order 6 bits represent 6 independent channel-detected (as opposed to tape unit-detected) I/O error conditions. Tape unit-detected errors are detailed by the sense data (see Note 2 below). The sense flag bits are encoded as follows:

Hex	*Meaning*
00	System software error
01	Sense data valid
02	Tape unit malfunction
03	Tape unit malfunction
04	Channel chaining check
08	Channel interface control check
10	Channel control check
20	Channel data check
40	Channel protection check
80	Channel program check

The meaning of the 6 channel-detected error conditions is described in the IBM publication, *ESA/390 Principles of Operation*, form SA22-7201.

(2) The first 6 bytes of hardware *sense data* (bytes 15-20) are meaningful only if the sense flag has a value of 01 (see Note 1). The following table indicates the meaning of each relevant bit. Note that the last 18 bytes of hardware sense data (bytes 6-23) are not particularily relevant (they include such items as the serial numbers for the tape drive and tape controller).

Byte	*Hex*	*Meaning*
15	80	Command reject—generally caused by attempting to write on a tape in which there is no ring
15	40	Operator intervention required—tape unit is not "ready"
15	20	Bus out check—bad parity on information transfer from main memory to tape unit
15	10	Equipment check—tape unit malfunction
15	08	Data check—any read or write tape parity error (see byte 18)
15	04	Overrun—system unable to transfer data fast enough between tape unit and main memory
15	01	Data converter check
16	80	Noise—1600 BPI data check or data detected in IBG
16	40/20	X'00'—tape unit disconnected X'20'—tape unit not "ready" X'40'—tape unit "ready" and not rewinding
16	10	7-track tape unit
16	08	Tape at load point
16	04	Tape unit writing
16	02	Ring out
16	01	Not capable—tape unit cannot process tapes of this density
18	80	Read-write vertical redundancy check (VRC)—one or more bytes have bad parity
18	40	Multiple track error (MTE) or longitudinal redundancy check (LRC)—for 1600 BPI, indicates that the signal was weak in more than one track; otherwise, the LRC was in error
18	20	Skew—excessive skew (between tracks) detected
18	10	End data check or cyclic redundancy check (CRC)—for 1600 BPI, end of block sync burst in error; otherwise, the CRC was in error
18	08	Envelope check or skew register VRC—for 1600 BPI write, at least one track had low amplitude; otherwise incorrect parity in tape unit auxiliary register
18	04	Tape unit operating at 1600 BPI
18	02	Tape unit in backward status
18	01	C compare—parity error in tape unit
19	40	Reject tape unit—tape unit dropped "ready" while tape moving
19	20	Tape indicate—tape has passed the end-of-tape marker
19	10	Write trigger VRC—byte written had in correct parity

(3) The *status* (bytes 26 and 58) indicates the condition of several software parameters. The meaning of each bit is shown below.

Byte	*Hex*	*Meaning*
26	01	Blocking enabled (BLK=ON)
26	02	Pool tape
26	04	Positioned at logical end of labeled tape
26	08	Name has been specified for next new file written

26	10	Positioned at load point
26	20	End-of-tape marker has been passed
26	40	Labeled tape file has been entered
26	80	Label processing enabled (LP=ON)
58	01	Expiration date has been specified (EXPDT)
58	02	Check expiration dates (DTCHK=ON)
58	04	Tape has ring in
58	08	Control character specified for next new file written

(4) The 2-byte block number (bytes 30-31) is an unsigned, 16-bit, positive integer. If the value is zero, the tape is positioned at the beginning of a file. If the value is 65535 (maximum), the block number is unknown (this occurs only when an *unlabeled* file is entered from the end).

(5) The 4-byte record number (bytes 60-63) will have a value of –1 if the record number is unknown (this occurs only when a file is entered from the end).

(6) The 24-byte tape-use statistics (bytes 148-171) are organized as follows:

Bytes	*Meaning*
148-151	Total number of read operations (fullword)
152-155	Total number of write operations (fullword)
156-159	Total number of control operations (fullword)
160-161	Number of recovered read errors (halfword)
162-163	Number of recovered write errors (halfword)
164-165	Number of recovered control errors (halfword)
166-167	Number of fatal read errors (halfword)
168-169	Number of fatal write errors (halfword)
170-171	Number of fatal control errors (halfword)

APPENDIX C
EXCHANGING MAGNETIC TAPES

Since magnetic tapes are compact and portable, they are an ideal medium for exchanging data between computer installations. An exchange tape should utilize a format that is compatible with both the sending and receiving installations. For example, if the sending and receiving installations have different labeling schemes, the exchange tape may have to be unlabeled. When deciding on a format for an exchange tape, it is helpful to have documentation for the tape-processing facilities at each installation.

The sender of an exchange tape should always provide the recipient with an exact description of the format and contents of the tape (see "A Checklist for Exchanging Tapes" below). If the sender is an MTS user, the *LABELSNIFF program can be used to generate a concise description of the files on the tape. This can then be sent along with the tape.

Exchanging Tapes with IBM OS/VS Installations

Before exchanging a tape with an IBM OS/VS installation, the user should check on the types of tape drives (number of tracks, density) available at that installation.

In general, the MTS labeling scheme is a *subset* of the IBM OS/VS labeling scheme. Labeled tapes written by MTS can be read by most OS/VS installations. However, OS/VS installations may have additional restrictions on volume names (called *volume serial labels* in IBM documentation) or file names (called *data set names* in IBM documentation). For example, OS/VS installations might not allow certain characters to be used in file names.

The IBM OS/VS labeling scheme has some options that are not currently available in MTS. For example:

(1) The OS/VS labeling scheme allows *multiple volumes*; that is, allows a file to be continued on multiple volumes (reels) of tape.

(2) The OS/VS labeling scheme allows user-defined file labels in addition to the standard header and trailer labels.

As long as these extra options are not used, a labeled tape written by an OS/VS installation can be read by MTS. For a complete description of the OS/VS labeling scheme, see the IBM publication, *MVS/ESA Magnetic Tape Labels and File Structure Administration* (SC26-4511). For details on the MTS labeling scheme, see Appendix D.

Exchanging ANSI-Labeled Tapes

The American National Standards Institute (ANSI) has defined a standard tape-labeling scheme to facilitate the exchange of tapes among computers of various manufacturers. This labeling scheme is similar to the IBM standard labeling scheme, with the following major differences:

(1) ANSI labels are recorded in 8-bit ISO 8859 ASCII. To ensure that the label is 7-bit ASCII-compatible, do not pick characters that are defined in the upper-half of the ISO standard for volume, owner, or data set name. In other words, avoid using special characters.

(2) The ANSI labeling scheme allows an unlimited number of user-defined file labels in addition to the standard header and trailer labels.

(3) The formats of the ANSI labels are slightly different from the formats of the IBM labels.

(4) The ANSI labels HDR2 and EOF2 are optional.

The ANSI standard labeling scheme is described in the ANSI publication, *American National Standard Magnetic Tape Labels for Information Interchange*, number ANSI X3.27-1969, and in the IBM publication, *MVS/ESA Magnetic Tape Labels and File Structure Administration* (SC26-4511).

MTS now supports a *subset* of the ANSI standard labeling scheme. A tape may be initialized as ANSI-labeled by checking the appropriate box on the Tape Initialization Form.

When MTS writes an ANSI-labeled tape, it generates exactly two header and two trailer labels for each file; it does not generate user-defined file labels. Such a tape can be read by most installations that support the ANSI standard labeling scheme. Before exchanging an ANSI-labeled tape, however, the sender should check on any restrictions (e.g., on densities, volume or file names, or blocking formats) that might be in effect at the receiving installation. To be on the safe side, blocking formats should be limited to the following: U, F, FB, D, or DB.

When MTS reads an ANSI-labeled tape, it tolerates the presence of user-defined file labels (although it does not process them) and/or the absence of the HDR2 and EOF2 labels. Thus, MTS can read ANSI-labeled tapes that are in a more general format than those it can write.

The mount request for an ANSI-labeled tape has the same form as that for a regular labeled tape. It is not necessary to specify the LBLTYPE keyword on the mount request because MTS will detect that the volume label is encoded in ISO 8859 ASCII or 7-bit ASCII and thus deduce the label type.

The data on ANSI-labeled tapes are encoded in ISO 8859 ASCII or 7-bit ASCII. When writing an ANSI-labeled tape, MTS will automatically translate from EBCDIC to ISO 8859 ASCII. If you want it translated to 7-bit ASCII instead, specify TRANSLATE=MTS(ZERO) either on the $MOUNT command or via a $CONTROL command. This will insure that any non-7-bit-ASCII character in the data will be translated to ASCII 3F rather than to an ISO character representation above ASCII 3F. Some installations require parity for 7-bit ASCII data. If this is needed, use TRANSLATE=MTS(ODD | EVEN | ONE) instead of TRANSLATE=MTS(ZERO).

When reading an ANSI-labeled tape, MTS will automatically translate from ISO 8859 ASCII (or 7-bit ASCII) to EBCDIC. Once a tape has been initialized as ANSI-labeled, therefore, it may be used in much the same way as a regular labeled tape. Note that ANSI-labeled tapes are intended mainly for exchange with non-IBM-compatible system installations. For use within MTS or for exchange with IBM-compatible system installations, regular labeled tapes are preferred.

A Checklist for Exchanging Tapes

The following checklist indicates some tape options that may vary between installations. An exchange tape should utilize only those options which are available at both the sending and receiving installations.

The sender of an exchange tape should provide the recipient with an exact description of the options used in writing the tape.

Density (Mode)

This may be 1600 or 6250 BPI for a 9-track tape; use 6250 BPI if possible. All 3480-compatible cartridges are the same density.

Most installations do not support all of the above densities. When there is a choice, higher densities are preferred.

Multiple Volumes

Some installations allow tape files to be continued on multiple volumes (reels) of tape. MTS does not support this option. In MTS, tape files must be limited to a single volume (reel).

Installations that allow multiple volumes may have restrictions on recording data near the EOT (end-of-tape) marker.

Multiple Files

In MTS, a tape may contain several files. Some installations do not allow a tape to contain more than one file.

The sender of an exchange tape should provide the receiver with a description of each file on the tape:

File number
File name (if labeled)
Blocking format
Data coding
Contents

Labeling

A tape may be either labeled or unlabeled.

Different installations may have different labeling schemes. If these are incompatible, an exchange tape may have to be unlabeled.

The MTS labeling scheme is a subset of the IBM standard labeling scheme. For details on the MTS labeling scheme, see Appendix D.

For exchange purposes, MTS also supports a subset of the ANSI standard labeling scheme.

If an exchange tape is labeled, the sender should provide the recipient with the following information:

Type of labels, e.g., IBM standard, ANSI standard
Tape volume name
File names

Blocking

In MTS, the maximum block size is 32767 bytes. At other installations, the maximum block size may be smaller, and some additional restrictions may be in effect, e.g., the block size may have to be a multiple of 6 or 10 bytes.

In MTS, the available blocking formats are U, F, FB, FBS, V, VB, VS, and VBS. These formats are compatible with the IBM formats of the same names. When exchanging a tape with an MTS or IBM-compatible installation, the sender should provide the recipient with the blocking parameters (FORMAT, SIZE, LRECL) for each file on the tape.

In addition to the blocking formats mentioned above, ANSI-labeled tapes may utilize the formats D and DB (which are similar to V and VB except that the block- and record-descriptor words are represented as ASCII decimal numbers). Other installations might not allow the use of V-type or spanned formats. When writing an ANSI-labeled tape, therefore, the blocking formats should be limited to the following: U, F, FB, D, or DB.

In general, fixed-length (FB type) blocking formats are more commonly available than variable-length formats.

Non-IBM-compatible installations might use a different terminology for describing blocking formats. When exchanging a tape with such an installation, the sender should provide the recipient with the following information about the blocking of each file:

Block length. Are the blocks fixed-length or variable-length? If variable, what are the minimum and maximum lengths?

Record length. Are the records fixed-length or variable-length? If variable, what are the minimum and maximum lengths, and how is the length of each record indicated? (For example, the length of each record might be indicated by a record-descriptor word preceding the record.)

Blocking format. Does each block contain one or several records? Can a record span several blocks? Are there nondata characters (record descriptors or terminators) interspersed with the data? Are short records padded with blanks or zeros to bring them up to a given length?

Data Coding

The data on a tape may consist of characters, represented in one of the following codes:

IBM Code Page 37 EBCDIC (8 bit) (Post T-day EBCDIC on MTS)
MTS EBCDIC (8 bit) (Pre-T-day EBCDIC on MTS)
ISO 8859 ASCII (8 bit)
ASCII (7 bit with odd, even, one or zero parity bit)

Other

MTS and the IBM systems use IBM Code Page 37 EBCDIC. Prior to February 22, 1988 (T-day), MTS used a slightly different version of EBCDIC (called MTS EBCDIC). Tapes written before that date may need to be converted to the IBM EBCDIC by copying them to MTS files, converting them, and copying them back to the tape. See Computing Center Memo 480, "Translation Day Character Code Changes in MTS," for further details about this conversion process.

The data on ANSI-labeled tapes are encoded in ISO 8859 or ASCII. When writing an ANSI-labeled tape, MTS will automatically translate from IBM EBCDIC to ISO 8859 (or ASCII by setting the TRANSLATE keyword appropriately). When reading an ANSI-labeled tape, MTS will automatically translate from ISO 8859 or ASCII to IBM EBCDIC. The system will also do these translations for an unlabeled tape, provided the keyword LBLTYPE=ANSI, and possibly the appropriate TRANSLATE keyword, was given on the mount request.

The data on a tape may also be in a noncharacter form, for example:

Binary fixed-point numbers
IBM 360/370 floating-point numbers

The schemes used to encode noncharacter data vary widely between computer manufacturers. When writing an exchange tape, it is usually preferable to represent data in character form, e.g., to represent floating-point numbers as character strings.

Reading Unknown Tapes

It may be necessary to read an exchange tape for which there is incomplete or erroneous documentation. If the volume name, or even whether the tape is labeled, is unknown, submit it with volume name "UNKNOWN", and attach an R sticker to the tape. The ITD operations staff will determine the volume name. If the tape has labels of some type other than IBM standard or ANSI standard labels, MTS will treat it as unlabeled.

Run *LABELSNIFF to retrieve information about the files on the tape. If the tape is unlabeled, file contents and blocking factors will have to be determined by inspection.

The *TAPEDUMP program may be used to print a character or hexadecimal dump of part of the tape. See the *TAPEDUMP description in this volume. From the dump, it may be possible to figure out the structure and coding of the tape.

The ITD consultants can also help you find information about a "mystery tape"; call 764–HELP for assistance.

APPENDIX D
MTS TAPE LABELS

MTS tape label formats are compatible with IBM OS/VS tape label formats. For full details on the OS/VS formats, see the IBM publication, *MVS/ESA Magnetic Tape Labels and File Structure Administration* (SC26-4511).

Volume Label

Each labeled tape begins with an 80-byte *volume label*, which contains the tape volume name and ownerID.

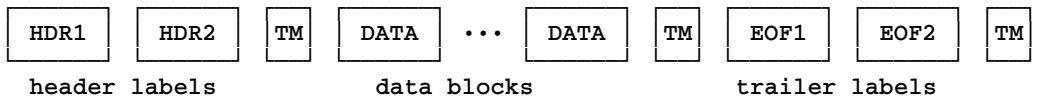
The volume label is created when the tape is first initialized. It is rewritten whenever the tape is reinitialized (by the INIT mount request keyword or the INIT control command) or relabeled (by the *LABEL utility program or the TAPEINIT subroutine).

The volume label has the following format. All label contents are recorded as EDCDIC characters. Items marked with a bullet (•) are functional, in the sense that they are checked by the system whenever the label is processed. Items not marked with a bullet are generated by the system at the time the label is created, but are not checked when the label is processed.

	Bytes	*Contents*
•	1-4	Label type—VOL1
•	5-10	Volume name—up to 6 characters, left-justified, padded with blanks
	11	Reserved—contains 0
	12-41	Reserved—contains blanks
	42-51	OwnerID—up to 10 characters, left-justified, padded with blanks
	52-80	Reserved—contains blanks

File Labels

Each file on a labeled tape has two 80-byte *header labels* (HDR1 and HDR2) and two 80-byte *trailer labels* (EOF1 and EOF2, or EOV1 and EOV2). The labels are used to record information about the file. Each file has the following format:



During normal operations, the labels are processed automatically by the system. The user may generally disregard the presence of the labels and extra tape marks. Each file appears to consist of zero or more data blocks followed by a single tape mark.

The trailer labels may be either EOF labels or EOV labels. EOF labels simply indicate the end of the file. EOV labels indicate the logical end of the volume (reel of tape) and are used in OS/VS installations (but not in MTS) to indicate that the file is continued on another volume.

File Label 1 (HDR1/EOF1/EOV1)

The HDR1, EOF1, and EOV1 labels have the same basic format, as shown in the following table. All label contents are recorded as EBCDIC characters. Items marked with a bullet (•) are functional, in the sense that they are checked by the system whenever the label is processed. Items not marked with a bullet are generated by the system at the time the label is created, but are not checked when the label is processed.

	Bytes	*Contents*
•	1-4	Label type—HDR1, EOF1, or EOV1
•	5-21	File name—up to 17 characters, left-justified, padded with blanks
	22-27	Tape volume name—up to 6 characters, left-justified, padded with blanks
	28-31	Reserved—contains 0001
•	32-35	File sequence number, giving the order of the file on the tape—from 0001 to 9999
	36-41	Reserved—contains blanks
	42-47	Creation date—in the form byyddd where b is a blank, yy is the year (00-99), and ddd is the day (001-366)
•	48-53	Expiration date—in the form byyddd where b is a blank, yy is the year (00-99), and ddd is the day (001-366)
	54	Reserved—contains 0
•	55-60	Block count—for an HDR1 label this is always 000000; for an EOF1 or EOV1 label this gives the number of blocks in the file, from 000000 to 999999
	61-73	Installation name—e.g., MTS ANN ARBOR
	74-80	Reserved—contains blanks

File Label 2 (HDR2/EOF2/EOV2)

The HDR2, EOF2, and EOV2 labels have the same basic format, as shown in the following table. All label contents are recorded as EBCDIC characters. Items marked with a bullet (•) are functional, in the sense that they are checked by the system whenever the label is processed. Items not marked with a bullet are generated by the system at the time the label is created, but are not checked when the label is processed.

	Bytes	*Contents*
•	1-4	Label type—HDR2, EOF2, or EOV2
•	5	Blocking format type—U, F, or V
•	6-10	Block size—from 00018 (normally) to 32767
•	11-15	Logical record length—from 00001 to 32767, except that for format U this is always 00000
	16	Tape density—represented in one of the following codes: 3—1600 BPI 4—6250 BPI 0—38000 BPI
	17	Reserved—contains 0

	18-25	UserID—left-justified, padded with blanks
	26-34	Batch receipt number (if the file was written by a batch job)—in the form /nnnnnn, left-justified, padded with blanks
•	35-36	Unused
	37-38	Reserved—contains blanks
•	39	Blocking attribute—a code that, combined with byte 5, indicates the blocking format: B—FB or VB format S—VS format R—FBS or VBS format b—U, F, or V format (b represents blank)
	40-41	Reserved—contains blanks
	42-47	Tape drive serial number (if available)
	48-80	Reserved—contains blanks

Note: When reading a tape, MTS will tolerate the absence of HDR2 and EOF2/EOV2 labels. This allows MTS to read labeled tapes produced by IBM DOS/TOS systems. When reading such a tape, the user must explicitly specify the blocking parameters.

APPENDIX E
MTS VLO TAPE LABELS

MTS VLO (Volume Label Only) tape-label format is similar to the MTS label format except that only the volume label is present. The individual files on the tape are unlabeled.

The VLO labeling scheme is useful for tapes containing large numbers of small files where the presence of separate file labels would consume too much space on the tape.

Volume Label

Each labeled tape begins with an 80-byte *volume label*, which contains the tape volume name and ownerID.

The volume label is created when the tape is first initialized. It is rewritten whenever the tape is reinitialized (by the INIT mount request keyword or the INIT control command) or relabeled (by the *LABEL utility program or the TAPEINIT subroutine).

The volume label has the following format. All label contents are recorded as EBCDIC characters. Items marked with a bullet (•) are functional, in the sense that they are checked by the system whenever the label is processed. Items not marked with a bullet are generated by the system at the time the label is created, but are not checked when the label is processed.

	Bytes	*Contents*
•	1-4	Label type—VOL1
•	5-10	Volume name—up to 6 characters, left-justified, padded with blanks
	11	Reserved—contains 0
	12-41	Reserved—contains blanks
	42-51	OwnerID—up to 10 characters, left-justified, padded with blanks.
	52-80	Reserved—contains blanks

APPENDIX F
MTS ANSI TAPE LABELS

MTS ANSI tape-label formats conform to the definition given by the American National Standard Institute, Inc. and are described in their publication, *American National Standard Magnetic Tape Labels for Information Interchange*, number ANSI X3.27-1969. This format is compatible with IBM ANSI tape-label formats. For full details on the IBM ANSI formats, see the IBM publication, *MVS/ESA Magnetic Tape Labels and File Structure Administration* (SC26-4511).

Volume Label

Each labeled tape begins with an 80-byte *volume label*, which contains the tape volume name and ownerID.

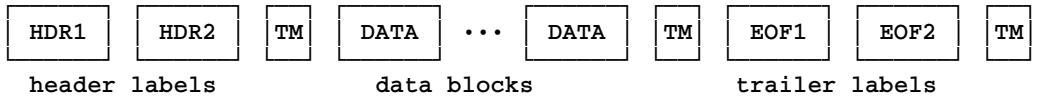
The volume label is created when the tape is first initialized. It is rewritten whenever the tape is reinitialized (by the INIT mount request keyword or the INIT control command) or relabeled (by the *LABEL utility program or the TAPEINIT subroutine).

The volume label has the following format. All label contents are recorded as ASCII characters. Items marked with a bullet (•) are functional, in the sense that they are checked by the system whenever the label is processed. Items not marked with a bullet are generated by the system at the time the label is created, but are not checked when the label is processed.

	Bytes	*Contents*
•	1-4	Label type—VOL1
•	5-10	Volume name—up to 6 characters, left-justified, padded with blanks
	11	Reserved—contains a blank
	12-37	Reserved—contains blanks
	38-51	OwnerID—up to 14 characters, left-justified, padded with blanks. MTS limits the owner information to the first 10 characters.
	52-79	Reserved—contains blanks
	80	Reserved—contains 1 which signifies that the tape adheres to the ANSI interchange standards.

File Labels

Each file on an ANSI labeled tape generated by MTS has two 80-byte *header labels* (HDR1 and HDR2) and two 80-byte *trailer labels* (EOF1 and EOF2, or EOV1 and EOV2). The labels are used to record information about the file. Each file has the following format:



During normal operations, the labels are processed automatically by the system. The user may generally disregard the presence of the labels and extra tape marks. Each file appears to consist of zero or more data blocks followed by a single tape mark.

The trailer labels may be either EOF labels or EOV labels. EOF labels simply indicate the end of the file. EOV labels used indicate the logical end of the volume (reel of tape) and are at OS/VS and other installations (but not in MTS) to indicate that the file is continued on another volume.

When reading an ANSI labeled tape, MTS will allow the absence of HDR2 and EOF2/EOV2 labels or will allow the presence of HDR3-HDRn and EOF3-EOFn/EOV3-EOVn labels.

File Label 1 (HDR1/EOF1/EOV1)

The HDR1, EOF1, and EOV1 labels have the same basic format, as shown in the following table. All label contents are recorded as ASCII characters. Items marked with a bullet (•) are functional, in the sense that they are checked by the system whenever the label is processed. Items not marked with a bullet are generated by the system at the time the label is created, but are not checked when the label is processed.

	Bytes	*Contents*
•	1-4	Label type—HDR1, EOF1, or EOV1
•	5-21	File name—up to 17 characters, left-justified, padded with blanks
	22-27	Tape volume name—up to 6 characters, left-justified, padded with blanks
	28-31	Reserved—contains 0001
•	32-35	File sequence number, giving the order of the file on the tape—from 0001 to 9999
	36-41	Reserved—contains blanks
	42-47	Creation date—in the form byyddd, where b is a blank, yy is the year (00-99), and ddd is the day (001-366)
•	48-53	Expiration date—in the form byyddd where b is a blank, yy is the year (00-99), and ddd is the day (001-366)
	54	Reserved—contains a blank
•	55-60	Block count—for an HDR1 label this is always 000000; for an EOF1 or EOV1 label this gives the number of blocks in the file, from 000000 to 999999
	61-73	Installation name—e.g., MTS ANN ARBOR
	74-80	Reserved—contains blanks

File Label 2 (HDR2/EOF2/EOV2)

The HDR2, EOF2, and EOV2 labels have the same basic format, as shown in the following table. All label contents are recorded as ASCII characters. Items marked with a bullet (•) are functional, in the sense that they are checked by the system whenever the label is processed. Items not marked with a bullet are generated by the system at the time the label is created, but are not checked when the label is processed.

	Bytes	*Contents*
•	1-4	Label type—HDR2, EOF2, or EOV2
•	5	Blocking format type—U, F, V, or D
•	6-10	Block size—from 00018 (normally) to 32767
•	11-15	Logical record length—from 00001 to 32767, except that for format U this is always 00000
	16	Tape density—represented in one of the following codes: 3—1600 BPI 4—6250 BPI 0—38000 BPI
	17	Reserved—contains 0
	18-25	UserID—left-justified, padded with blanks
	26-34	Batch receipt number (if the file was written by a batch job)—in the form /nnnnnn, left-justified, padded with blanks
	35-36	Tape recording technique—contains blanks
	37	Control characters—not used by MTS but may be b—contains no control characters (b represents blank) A—contains ANSI control characters
	38	Reserved—contains a blank
•	39	Blocking attribute—a code that, combined with byte 5, indicates the blocking format: B—blocked records b—unblocked records (b represents blank)
	40-41	Reserved—contains blanks
	42-47	Tape drive serial number (if available)
	48-50	Reserved—contains blanks
	51-52	Block-prefix length or buffer offset—from 00 to 99.
	53-80	Reserved—contains blanks

Note: Bytes 16-50 are operating system dependent and are interpreted as above only if the tape was created by an MTS system, or an OS/VS system identified by “OS360” in the installation name field (bytes 61-73) of the corresponding HDR1 label.

Note: When reading a tape, MTS will tolerate the absence of HDR2 and EOF2/EOV2 labels. When reading such a tape, the user must explicitly specify the blocking parameters.

APPENDIX G
THE TAPE CATALOG DATABASE

When a magnetic is submitted to ITD, it is entered into the tape catalog database by a staff member.

The tape catalog database contains a record of each tape that has been submitted to ITD. The tapes are cataloged by *tape name*.

Each tape record in the database has several items of information. Among these items are:

the current tape name
the receipt code
the owner
the permit status
the volume name
the pseudodevice name
the density of the tape
the submission date (creation date)
the time of the last mount (last mounted)
the time of the last mount with write-protect off (last change)
the time of the last mount attempt (last reference)
the success or failure of the last mount attempt (mount result)
the userID of the last user
the number of days since the last mount (idle days)
the number of times the tape has been mounted (references)
the location of the tape (machine room or off-site storage)
the status (ok, damaged, lost, borrowed, or unknown)
the rack number

For new tapes submitted by running the *TAPESUBMIT program, the initial tape name is the same as the assigned receipt code. For tapes that were submitted before the tape catalog database system was instituted (September 1986), the initial tape name is the same as the old rack number. In either case, the tape name may be changed by running the *TAPES program and issuing the RENAME command (see below).

The *TAPES program may be run to query the information in the tape catalog database. The *TAPES program also allows the user to change the tape name, the permit status, and the ownership. The program is invoked with the command

```
$RUN *TAPES
```

The program prompts for commands with "<", and the response should be one of the commands listed below. Further information about the *TAPES program is given later in this volume.

STATUS tapename [options ...]

The STATUS command prints a complete list of the user's own tapes as well as all tapes permitted to him or her, but belonging to another user. The STATUS command prints specific information about the user's tapes or tapes permitted but belonging to another user. The STATUS command in *TAPES is very similar to the $FILESTATUS command in MTS.

"tapename" may be the name of a tape optionally preceded by a userID and colon, or a question mark, or a question mark preceded by a userID and a colon, for example,

```
STATUS MYTAPE
STATUS WABC:MYTAPE
STATUS ?
STATUS WABC:?
```

The first example produces specific information about the tape MYTAPE. The second example produces specific information about the tape MYTAPE that belongs to the userID WABC. The third example produces a list of tape names that are owned by the current userID. The last example produces a list of tapes that may be accessed by the current userID, but belong to the userID WABC.

Several options may be specified on the STATUS command. Some of the more common options are listed below.

The following option redirects the output stream:

OUTPUT=Fdname

The following options select groups of items:

TOTAL

This prints all the information about a particular tape available in *TAPES.

SHORTINFO

This prints all information except the tape's reference count and references per month.

The following options print specific information about the tape:

ACCESS
COMMENT
CREATE DATE
DENSITY (or MODE)
DRIVES
IDLE DAYS
LASTCHG
LAST MOUNTED
LASTREF
LAST USER
LOCATION
MEDIA

MOUNT RESULT
NAME (owner's given name)
OWNER
PDN (preferred pseudodevice name)
RACKNUMBER
RECEIPT
REFERENCES
SITE
STATUS
TAPENAME

For example, to display all information about all tapes on *PRINT*, enter

```
STATUS ? TOTAL, OUTPUT=*PRINT*
```

To display the last time the tape WABC:VOL.99 was mounted with a ring in, enter

```
STATUS WABC:VOL.99 LASTCHG
```

PERMIT tapelist [access [accessor]]
PERMIT tapename LIKE tapename

The PERMIT command permits a tape, much like the MTS $PERMIT command for files, to other userIDs or groups of userIDs. The syntax follows the syntax of the $PERMIT command except for the following differences:

(1) When permitting a tape "LIKE" another tape, only one tape name may appear after LIKE, for example,

```
PERMIT TAPE4 LIKE TAPE1
```

(2) Only the access types listed below are valid with tapes.

For more information on the $PERMIT command, see *MTS Volume 1: The Michigan Terminal System*.

The valid access types are as follows (minimum abbreviations are underlined):

READ
WRITE
{READWRITE | RW}
EDIT
RUN
{RENAME | RNA}
DESTROY
PERMIT
FULL
UNLIMITED
DEFAULT
NONE

Accessors are generally the same as for files:

ALL
ME
OTHERS
OWNER
userID
PROJECT=projectID

Program key access is not supported.

For example, to permit TAPE1 so that user WXYZ can read it, enter

```
PERMIT TAPE1 READ WXYZ
```

Like MTS files, users only have access to tapes that have been permitted to them. Remember that tapes, like files, are subject to theft if they are improperly permitted, for example, unlimited to others. ITD recommends that tapes be permitted carefully and accurately. ITD cannot be responsible for the theft of improperly permitted tapes.

RENAME oldtapename [TO] newtapename

The RENAME command renames a tape, much as a file is renamed in MTS by using the $RENAME command. The syntax is the same as the $RENAME command except that confirmation following the new name is neither required nor allowed. For example, to change the name of TAPE1 to POPULATION.DATA, enter

```
RENAME TAPE1 TO POPULATION.DATA
```

The maximum length of a tape name is 44 characters.

MODIFY tapename option

The MODIFY command changes the pseudodevice name (PDN) for a tape (the default is *T*). For example,

```
MODIFY TAPE1 PDN=*OUT*
```

The above example changes the PDN for TAPE1 to *OUT*. The MODIFY command also can change any comments associated with a tape. For example:

```
MODIFY TAPE2 COMMENT="Includes data from TAPE1"
```

The comment string must be enclosed in double quotes and be less than 30 characters in length.

STOP

The STOP command terminates execution of *TAPES and returns to MTS command mode or to the MTS command that was in control when execution of *TAPES was initiated.

MTS

The MTS command returns to MTS command mode or to the MTS command that was in control when execution of *TAPES was initiated. Execution of *TAPES may be resumed by giving the MTS $RESTART command.

APPENDIX H
NOTES ON PL/I BLOCKING

The section "PL/I Input/Output in MTS" in *MTS Volume 7: PL/I in MTS*, should be consulted for detailed information on the usage of PL/I in MTS. However, for convenience, a few notes relevant to the usage of magnetic tapes are included here.

(1) The PL/I-(F) run-time I/O library always disables the MTS tape blocking facility for each tape being processed by a PL/I-(F) program since the I/O library provides its own blocking facility. This does not apply to PL/I Optimizer programs.

(2) PL/I users who use any of the V formats should note that the PL/I logical record length value must be specified as 4 greater than the actual logical record length to allow for the record descriptor word.

(3) PL/I does not allow concatenated input tapes with different record formats. Non-PL/I MTS programs are not subject to this restriction.

(4) When *reading* variable-spanned tapes, PL/I users must specify format VS or VBS, whereas other MTS users can specify any of the V formats.

PROGRAMS FOR HANDLING MAGNETIC TAPES

The following programs are useful for handling tapes. Their descriptions follow this page.

*ASCEBCD	Translates ASCII characters into EBCDIC characters.
*CMSTAPELOAD	Restores CMS tapes to MTS line files.
*CMSTAPESCAN	Displays CMS tapes.
*EBCDASC	Translates EBCDIC characters into ASCII characters.
*FS	Saves disk files (including line numbers) on tape. Restores tape files to disk.
*LABEL	Initializes (empties) labeled or unlabeled tapes.
*LABELSNIFF	Scans tapes and prints label information.
*TAPECOPY	Copies files from one tape to another.
*TAPEDUMP	Produces binary or character tape dumps.
*TAPEFIXER	Assists in recovering data from a labeled tape that was accidentally overwritten.
*TAPERETRIEVE	Retrieves tapes from the tape catalog database.
*TAPES	Gives tape-status information and allows users to permit tapes.
*TAPESUBMIT	Submits tapes into the tape catalog database.

*ASCEBCD

Contents: The ASCII-to-EBCDIC translation program.

Purpose: To translate records from ASCII to EBCDIC code.

Use: The program is invoked by the $RUN command.

Program Key: *ASCEBCD

Logical I/O Units Referenced:

SCARDS	ASCII records to be translated.
SPRINT	listing of the records resulting from the translation.
SPUNCH	EBCDIC records resulting from the translation.

Return Codes:

0	Successful return.
4	Invalid parameter specified.

Parameters: At most, one of the following parameters may be specified in the PAR field of the $RUN command. The default parameter is TDAY88.

TDAY88 Characters are translated according to the TDAY88 translation table. This translation corresponds to that used in MTS for translating ISO 8859/1 8-bit codes (ASCII) to the IBM Code Page 37 codes (EBCDIC) used in MTS.

7BIT Characters are translated according to a 7-bit translation table. This translation corresponds to that used in MTS for translating the low-order 7 bits of ISO 8859/1 codes (ASCII) to the IBM Code Page 37 codes (EBCDIC) used in MTS.

Description: Records are read from SCARDS, translated according to the appropriate table as specified by the parameter field, and written to SPUNCH. Each output record is also prefixed by a blank character (for logical carriage control) and written to SPRINT. An end-of-file on SCARDS terminates the program.

The 7BIT translation uses a 256-entry "folded" table in which the first and second halves are identical so that the effect is to ignore the high-order ASCII bit.

See the *EBCDASC description in this volume for a program to translate from EBCDIC into ASCII.

The TDAY88 translation table is given below. This table is also contained in the file DOC:ALLCHARTABLE.

Example:

```
$RUN *ASCEBCD SCARDS=*T* SPUNCH=NEWFILE SPRINT=*PRINT*
```

In the above example, ASCII records are read from the tape *T*, translated to EBCDIC, written to the file NEWFILE, and printed.

EBCDIC	ASCII	Graphic	Description
X'00'	X'00'	NUL	null
X'01'	X'01'	SOH	start of heading (Ctrl-A)
X'02'	X'02'	STX	start of text (Ctrl-B)
X'03'	X'03'	ETX	end of text (Ctrl-C)
X'37'	X'04'	EOT	end of transmission (Ctrl-D)
X'2D'	X'05'	ENQ	enquiry (Ctrl-E)
X'2E'	X'06'	ACK	acknowledge (Ctrl-F)
X'2F'	X'07'	BEL	bell (Ctrl-G)
X'16'	X'08'	BS	backspace (Ctrl-H)
X'05'	X'09'	HT	horizontal tabulation (Ctrl-I)
X'25'	X'0A'	LF	line feed (Ctrl-J)
X'0B'	X'0B'	VT	vertical tabulation (Ctrl-K)
X'0C'	X'0C'	FF	form feed (Ctrl-L)
X'0D'	X'0D'	CR	carriage return (Ctrl-M)
X'0E'	X'0E'	SO	shift-out (Ctrl-N)
X'0F'	X'0F'	SI	shift-in (Ctrl-O)
X'10'	X'10'	DLE	data link escape (Ctrl-P)
X'11'	X'11'	DC1	device control 1 (X-Off, Ctrl-Q)
X'12'	X'12'	DC2	device control 2 (Ctrl-R)
X'13'	X'13'	DC3	device control 3 (X-On, Ctrl-S)
X'3C'	X'14'	DC4	device control 4 (Ctrl-T)
X'3D'	X'15'	NAK	negative acknowledge (Ctrl-U)
X'32'	X'16'	SYN	synchronous idle (Ctrl-V)
X'26'	X'17'	ETB	end of transmission block (Ctrl-W)
X'18'	X'18'	CAN	cancel (Ctrl-X)
X'19'	X'19'	EM	end of medium (Ctrl-Y)
X'3F'	X'1A'	SUB	substitute character (Ctrl-Z)
X'27'	X'1B'	ESC	escape (Escape)
X'1C'	X'1C'	FS	file separator
X'1D'	X'1D'	GS	group separator
X'1E'	X'1E'	RS	record separator
X'1F'	X'1F'	US	unit separator
X'40'	X'20'		space (blank)
X'5A'	X'21'	!	exclamation mark
X'7F'	X'22'	"	quotation mark (double quote)
X'7B'	X'23'	#	number sign (hash mark, sharp sign)
X'5B'	X'24'	$	dollar sign
X'6C'	X'25'	%	percent sign
X'50'	X'26'	&	ampersand (and sign)
X'7D'	X'27'	'	apostrophe (single quote)
X'4D'	X'28'	(	left parenthesis
X'5D'	X'29'	)	right parenthesis
X'5C'	X'2A'	*	asterisk (star)
X'4E'	X'2B'	+	plus sign
X'6B'	X'2C'	,	comma
X'60'	X'2D'	-	minus sign or hyphen
X'4B'	X'2E'	.	period, full stop
X'61'	X'2F'	/	solidus (slash)
X'F0'	X'30'	0	digit zero

X'F1'	X'31'	1	digit one
X'F2'	X'32'	2	digit two
X'F3'	X'33'	3	digit three
X'F4'	X'34'	4	digit four
X'F5'	X'35'	5	digit five
X'F6'	X'36'	6	digit six
X'F7'	X'37'	7	digit seven
X'F8'	X'38'	8	digit eight
X'F9'	X'39'	9	digit nine
X'7A'	X'3A'	:	colon
X'5E'	X'3B'	;	semicolon
X'4C'	X'3C'	<	less-than sign
X'7E'	X'3D'	=	equals sign
X'6E'	X'3E'	>	greater-than sign
X'6F'	X'3F'	?	question mark
X'7C'	X'40'	@	commercial at
X'C1'	X'41'	A	capital A
X'C2'	X'42'	B	capital B
X'C3'	X'43'	C	capital C
X'C4'	X'44'	D	capital D
X'C5'	X'45'	E	capital E
X'C6'	X'46'	F	capital F
X'C7'	X'47'	G	capital G
X'C8'	X'48'	H	capital H
X'C9'	X'49'	I	capital I
X'D1'	X'4A'	J	capital J
X'D2'	X'4B'	K	capital K
X'D3'	X'4C'	L	capital L
X'D4'	X'4D'	M	capital M
X'D5'	X'4E'	N	capital N
X'D6'	X'4F'	O	capital O
X'D7'	X'50'	P	capital P
X'D8'	X'51'	Q	capital Q
X'D9'	X'52'	R	capital R
X'E2'	X'53'	S	capital S
X'E3'	X'54'	T	capital T
X'E4'	X'55'	U	capital U
X'E5'	X'56'	V	capital V
X'E6'	X'57'	W	capital W
X'E7'	X'58'	X	capital X
X'E8'	X'59'	Y	capital Y
X'E9'	X'5A'	Z	capital Z
X'BA'	X'5B'	[	left square bracket
X'E0'	X'5C'	\	reverse solidus (backslash)
X'BB'	X'5D'	]	right square bracket
X'B0'	X'5E'	^	circumflex accent
X'6D'	X'5F'	_	low line (underscore)
X'79'	X'60'	`	grave accent
X'81'	X'61'	a	small a
X'82'	X'62'	b	small b
X'83'	X'63'	c	small c

X'84'	X'64'	d	small d
X'85'	X'65'	e	small e
X'86'	X'66'	f	small f
X'87'	X'67'	g	small g
X'88'	X'68'	h	small h
X'89'	X'69'	i	small i
X'91'	X'6A'	j	small j
X'92'	X'6B'	k	small k
X'93'	X'6C'	l	small l
X'94'	X'6D'	m	small m
X'95'	X'6E'	n	small n
X'96'	X'6F'	o	small o
X'97'	X'70'	p	small p
X'98'	X'71'	q	small q
X'99'	X'72'	r	small r
X'A2'	X'73'	s	small s
X'A3'	X'74'	t	small t
X'A4'	X'75'	u	small u
X'A5'	X'76'	v	small v
X'A6'	X'77'	w	small w
X'A7'	X'78'	x	small x
X'A8'	X'79'	y	small y
X'A9'	X'7A'	z	small z
X'C0'	X'7B'	{	left curly bracket (left brace)
X'4F'	X'7C'	|	vertical line (bar, "or sign)
X'D0'	X'7D'	}	right curly bracket (right brace)
X'A1'	X'7E'	~	tilde (wavy line)
X'07'	X'7F'	DEL	delete (rubout, DEL control char)
X'20'	X'80'		...
X'21'	X'81'		...
X'22'	X'82'		...
X'23'	X'83'		...
X'24'	X'84'		...
X'15'	X'85'		...
X'06'	X'86'		...
X'17'	X'87'		...
X'28'	X'88'		...
X'29'	X'89'		...
X'2A'	X'8A'		...
X'2B'	X'8B'		...
X'2C'	X'8C'		...
X'09'	X'8D'		...
X'0A'	X'8E'		...
X'1B'	X'8F'		...
X'30'	X'90'		...
X'31'	X'91'		...
X'1A'	X'92'		...
X'33'	X'93'		...
X'34'	X'94'		...
X'35'	X'95'		...
X'36'	X'96'		...

X'08'	X'97'		...
X'38'	X'98'		...
X'39'	X'99'		...
X'3A'	X'9A'		...
X'3B'	X'9B'		...
X'04'	X'9C'		...
X'14'	X'9D'		...
X'3E'	X'9E'		...
X'FF'	X'9F'		...
X'41'	X'A0'		no-break space
X'AA'	X'A1'	¡	inverted exclamation mark
X'4A'	X'A2'	¢	cent sign
X'B1'	X'A3'	£	pound sign (Sterling currency)
X'9F'	X'A4'	◊	currency sign (lozenge)
X'B2'	X'A5'	¥	yen sign (Nipponese currency)
X'6A'	X'A6'		broken bar
X'B5'	X'A7'	§	section sign (S-half-above-S sign)
X'BD'	X'A8'	¨	diaeresis or umlaut
X'B4'	X'A9'	©	copyright sign (circled capital C)
X'9A'	X'AA'	a	ordinal indicator feminine
X'8A'	X'AB'	«	angle quotation mark left (<< mark)
X'5F'	X'AC'	¬	not sign
X'CA'	X'AD'		soft hyphen
X'AF'	X'AE'	®	registered sign (circled capital R)
X'BC'	X'AF'	¯	macron
X'90'	X'B0'	°	degree sign
X'8F'	X'B1'	±	plus or minus sign
X'EA'	X'B2'	2	superscript two (squared)
X'FA'	X'B3'	3	superscript three (cubed)
X'BE'	X'B4'	´	acute accent
X'A0'	X'B5'	µ	micro sign (small mu)
X'B6'	X'B6'	¶	pilcrow (paragraph, double-barred P)
X'B3'	X'B7'	·	middle dot (scalar product)
X'9D'	X'B8'	¸	cedilla
X'DA'	X'B9'	1	superscript one
X'9B'	X'BA'	o	ordinal indicator, masculine
X'8B'	X'BB'	»	angle quotation mark right (>> mark)
X'B7'	X'BC'		fraction one-quarter (1/4)
X'B8'	X'BD'		fraction one-half (1/2)
X'B9'	X'BE'		fraction three-quarters (3/4)
X'AB'	X'BF'	¿	inverted question mark
X'64'	X'C0'	À	capital A with grave accent
X'65'	X'C1'	Á	capital A with acute accent
X'62'	X'C2'	Â	capital A with circumflex accent
X'66'	X'C3'	Ã	capital A with tilde
X'63'	X'C4'	Ä	capital A with diaeresis
X'67'	X'C5'	Å	capital A with ring
X'9E'	X'C6'	Æ	capital AE dipthong
X'68'	X'C7'	Ç	capital C with cedilla
X'74'	X'C8'	È	capital E with grave accent
X'71'	X'C9'	É	capital E with acute accent

X'72'	X'CA'	Ê	capital E with circumflex accent
X'73'	X'CB'	Ë	capital E with diaeresis
X'78'	X'CC'	Ì	capital I with grave accent
X'75'	X'CD'	Í	capital I with acute accent
X'76'	X'CE'	Î	capital I with circumflex accent
X'77'	X'CF'	Ä	capital I with diaeresis
X'AC'	X'D0'		capital D with stroke, Icelandic eth
X'69'	X'D1'	Ñ	capital N with tilde
X'ED'	X'D2'	Ò	capital O with grave accent
X'EE'	X'D3'	Ó	capital O with acute accent
X'EB'	X'D4'	Ô	capital O with circumflex accent
X'EF'	X'D5'	Õ	capital O with tilde
X'EC'	X'D6'	Ö	capital O with diaeresis
X'BF'	X'D7'	×	multiply sign (vector product)
X'80'	X'D8'	Ø	capital O with slash
X'FD'	X'D9'	Ù	capital U with grave accent
X'FE'	X'DA'	Ú	capital U with acute accent
X'FB'	X'DB'	Û	capital U with circumflex accent
X'FC'	X'DC'	Ü	capital U with diaeresis
X'AD'	X'DD'	Ÿ	capital Y with acute accent
X'AE'	X'DE'		capital thorn, Icelandic
X'59'	X'DF'	ß	small sharp s, German
X'44'	X'E0'	à	small a with grave accent
X'45'	X'E1'	á	small a with acute accent
X'42'	X'E2'	â	small a with circumflex accent
X'46'	X'E3'	ã	small a with tilde
X'43'	X'E4'	ä	small a with diaeresis
X'47'	X'E5'	å	small a with ring above
X'9C'	X'E6'	æ	small ae dipthong
X'48'	X'E7'	ç	small c with cedilla
X'54'	X'E8'	è	small e with grave accent
X'51'	X'E9'	é	small e with acute accent
X'52'	X'EA'	ê	small e with circumflex accent
X'53'	X'EB'	ë	small e with diaeresis
X'58'	X'EC'	ì	small i with grave accent
X'55'	X'ED'	í	small i with acute accent
X'56'	X'EE'	î	small i with circumflex accent
X'57'	X'EF'	ï	small i with diaeresis
X'8C'	X'F0'		small eth, Icelandic
X'49'	X'F1'	ñ	small n with tilde
X'CD'	X'F2'	ò	small o with grave accent
X'CE'	X'F3'	ó	small o with acute accent
X'CB'	X'F4'	ô	small o with circumflex accent
X'CF'	X'F5'	õ	small o with tilde
X'CC'	X'F6'	ö	small o with diaeresis
X'E1'	X'F7'	÷	divide sign (dot over line over dot)
X'70'	X'F8'	ø	small o with slash
X'DD'	X'F9'	ù	small u with grave accent
X'DE'	X'FA'	ú	small u with acute accent
X'DB'	X'FB'	û	small u with circumflex accent
X'DC'	X'FC'	ü	small u with diaeresis

X'8D'	X'FD'	ý	small y with acute accent
X'8E'	X'FE'		small thorn, Icelandic
X'DF'	X'FF'	ÿ	small y diaeresis

Note: On some printers, a few of the above graphics may not print.

*CMSTAPELOAD

Contents: A CMS tape-to-disk file restore program.

Purpose: To restore tape files in the CMS TAPE DUMP format to MTS line files.

Use: *CMSTAPELOAD is invoked by the $RUN command.

Program Key: *CMSTAPELOAD

Logical I/O Units Referenced:

0	The pseudodevice name of a magnetic tape written by the CMS TAPE utility.
SPRINT	the listing of restored files.
SERCOM	error comments and prompts.
GUSER	responses to prompting messages.

Parameters: The PAR field is used to specify how to handle CMS fileIDs which would result in a MTS file names longer than 12 characters. Two options are allowed:

TRUNCATE / FDname Default: *CMSFILETYPEMAP

TRUNCATE indicates that the CMS FILENAME and FILETYPE are to be concatenated with a delimiting period and the resultant string truncated to 12 characters.

Alternatively, the PAR field may specify a file or device (FDname) which contains a CMS to MTS file-type map such as may be found in the file *CMSFILETYPEMAP (see description in *MTS Volume 2*).

Return Codes:

0	Successful return.
8	Error return.

Description: *CMSTAPELOAD restores files saved with the ”TAPE DUMP“ command available on IBM’s CMS (VM/370 and VM/SP) operating system to MTS line files. *CMSTAPELOAD is designed to restore the entire contents of the tape, spanning physical tape-file boundaries (single tape marks). At present, CMS file-ID filter patterns are not supported.

The program reads and decodes each logical file and copies and copies its contents into a temporary file named ”–CMSUT1“. When an entire file is copied, the program will attempt to rename the temporary file to a permanent MTS line file using the original file’s ”file name“ and ”file type“ separated by a period. The file mode is ignored. If the TRUNCATE keyword is given in the PAR field and the length of the resulting file-name string exceeds 12 characters, the new file name is truncated to 12 characters. For example,

```
"MYFILE DATA A1"         becomes   "MYFILE.DATA"

"YOURFILE LONGTYPE B1"   becomes   "YOURFILE.LON"
```

If a file-type map is specified in the PAR field, the map is searched for a file-type abbreviation code which is used as part of the MTS file name.

If the file already exists or if file-name truncation results in a duplicate new file name, the user is prompted for a replacement new file name.

The program processes all files until the CMS end-of-volume marker (two consecutive file marks) is encountered.

For further information on CMS tapes, see the IBM publication, *VM/ESA CMS Command Reference*, form SC24-5461.

*CMSTAPESCAN

Contents: A CMS tape-scanning program.

Purpose: To display the contents of a magnetic tape written in the CMS TAPE DUMP format.

Use: The program is invoked by the $RUN command.

Program Key: *CMSTAPESCAN

Logical I/O Units Referenced:

0	the pseudodevice name of the CMS TAPE DUMP tape to be scanned.
SPRINT	the listing of the files on the scanned tape.

Return Codes: Always zero.

Description: *CMSTAPESCAN lists contents of the magnetic tape assigned to logical I/O unit 0 on the SPRINT file in an intelligble format. A descriptive header is written every 50 output lines.

For further information on CMS tapes, see the IBM publication, *VM/ESA CMS Command Reference*, form SC24-5461.

*EBCDASC

Contents: The EBCDIC-to-ASCII translation program.

Purpose: To translate lines from EBCDIC to ASCII code.

Use: The program in invoked by the $RUN command.

Program Key: *EBCDASC

Logical I/O Units Referenced:

SCARDS	EBCDIC records to be translated.
SPUNCH	ASCII records resulting from the translation.

Parameters: At most, one of the following parameters may be specified in the PAR field of the $RUN command. The default parameter is TDAY88.

TDAY88 Characters are translated according to the TDAY88 translation table. This translation corresponds to that used in MTS for translating the IBM Code Page 37 codes used in MTS (EBCDIC) to ISO 8859/1 8-bit codes (ASCII).

EVEN Characters are translated according to a 7-bit translation table; even-parity ASCII characters are produced, i.e., characters with the high-order parity bit set to either one or zero such that the number of one-bits in each eight-bit character is even.

ODD Characters are translated according to a 7-bit translation table; odd-parity ASCII characters are produced.

OFF or ZERO
Characters are translated according to a 7-bit translation table; ASCII characters with the high-order parity bit set to zero are produced.

ON or ONE
Characters are translated according to a 7-bit translation table; ASCII characters with the high-order parity bit set to one are produced.

Return Codes:

0	Successful return.
4	Invalid parameter specified.

Description: Records are read from SCARDS, translated according to the appropriate table as specified by the parameter field, and written to SPUNCH. An end-of-file on SCARDS terminates the program.

For EVEN, ODD, OFF, ZERO, ON, or ONE, all EBCDIC characters that would translate to codes in the top half of the ISO 8859/1 table (hex 80-FF) are translated into the ASCII SUB character (hex 1A).

See the *ASCEBCD description in this volume for a program to translate from ASCII to EBCDIC.

The TDAY88 translation table is given below. This table is also contained in the file DOC:ALLCHARTABLE.

Example:

```
$RUN *EBCDASC SCARDS=OLDFILE SPUNCH=*T* PAR=EVEN
```

In the above example, EBCDIC records are read from the file OLDFILE, translated to even-parity ASCII using the 7-bit translation table, and written to the tape *T*.

EBCDIC	ASCII	Graphic	Description
X'00'	X'00'	NUL	null
X'01'	X'01'	SOH	start of heading (Ctrl-A)
X'02'	X'02'	STX	start of text (Ctrl-B)
X'03'	X'03'	ETX	end of text (Ctrl-C)
X'04'	X'9C'		...
X'05'	X'09'	HT	horizontal tabulation (Ctrl-I)
X'06'	X'86'		...
X'07'	X'7F'	DEL	delete (rubout, DEL control char)
X'08'	X'97'		...
X'09'	X'8D'		...
X'0A'	X'8E'		...
X'0B'	X'0B'	VT	vertical tabulation (Ctrl-K)
X'0C'	X'0C'	FF	form feed (Ctrl-L)
X'0D'	X'0D'	CR	carriage return (Ctrl-M)
X'0E'	X'0E'	SO	shift-out (Ctrl-N)
X'0F'	X'0F'	SI	shift-in (Ctrl-O)
X'10'	X'10'	DLE	data link escape (Ctrl-P)
X'11'	X'11'	DC1	device control 1 (X-Off, Ctrl-Q)
X'12'	X'12'	DC2	device control 2 (Ctrl-R)
X'13'	X'13'	DC3	device control 3 (X-On, Ctrl-S)
X'14'	X'9D'		...
X'15'	X'85'		...
X'16'	X'08'	BS	backspace (Ctrl-H)
X'17'	X'87'		...
X'18'	X'18'	CAN	cancel (Ctrl-X)
X'19'	X'19'	EM	end of medium (Ctrl-Y)
X'1A'	X'92'		...
X'1B'	X'8F'		...
X'1C'	X'1C'	FS	file separator
X'1D'	X'1D'	GS	group separator
X'1E'	X'1E'	RS	record separator
X'1F'	X'1F'	US	unit separator
X'20'	X'80'		...
X'21'	X'81'		...
X'22'	X'82'		...
X'23'	X'83'		...
X'24'	X'84'		...
X'25'	X'0A'	LF	line feed (Ctrl-J)
X'26'	X'17'	ETB	end of transmission block (Ctrl-W)
X'27'	X'1B'	ESC	escape (Escape)
X'28'	X'88'		...
X'29'	X'89'		...
X'2A'	X'8A'		...
X'2B'	X'8B'		...
X'2C'	X'8C'		...
X'2D'	X'05'	ENQ	enquiry (Ctrl-E)
X'2E'	X'06'	ACK	acknowledge (Ctrl-F)
X'2F'	X'07'	BEL	bell (Ctrl-G)
X'30'	X'90'		...

X'31'	X'91'		...
X'32'	X'16'	SYN	synchronous idle (Ctrl-V)
X'33'	X'93'		...
X'34'	X'94'		...
X'35'	X'95'		...
X'36'	X'96'		...
X'37'	X'04'	EOT	end of transmission (Ctrl-D)
X'38'	X'98'		...
X'39'	X'99'		...
X'3A'	X'9A'		...
X'3B'	X'9B'		...
X'3C'	X'14'	DC4	device control 4 (Ctrl-T)
X'3D'	X'15'	NAK	negative acknowledge (Ctrl-U)
X'3E'	X'9E'		...
X'3F'	X'1A'	SUB	substitute character (Ctrl-Z)
X'40'	X'20'		space (blank)
X'41'	X'A0'		no-break space
X'42'	X'E2'	â	small a with circumflex accent
X'43'	X'E4'	ä	small a with diaeresis
X'44'	X'E0'	à	small a with grave accent
X'45'	X'E1'	á	small a with acute accent
X'46'	X'E3'	ã	small a with tilde
X'47'	X'E5'	å	small a with ring above
X'48'	X'E7'	ç	small c with cedilla
X'49'	X'F1'	ñ	small n with tilde
X'4A'	X'A2'	¢	cent sign
X'4B'	X'2E'	.	period, full stop
X'4C'	X'3C'	<	less-than sign
X'4D'	X'28'	(	left parenthesis
X'4E'	X'2B'	+	plus sign
X'4F'	X'7C'	|	vertical line (bar, "or" sign)
X'50'	X'26'	&	ampersand (and sign)
X'51'	X'E9'	é	small e with acute accent
X'52'	X'EA'	ê	small e with circumflex accent
X'53'	X'EB'	ë	small e with diaeresis
X'54'	X'E8'	è	small e with grave accent
X'55'	X'ED'	í	small i with acute accent
X'56'	X'EE'	î	small i with circumflex accent
X'57'	X'EF'	ï	small i with diaeresis
X'58'	X'EC'	ì	small i with grave accent
X'59'	X'DF'	ß	small sharp s, German
X'5A'	X'21'	!	exclamation mark
X'5B'	X'24'	$	dollar sign
X'5C'	X'2A'	*	asterisk (star)
X'5D'	X'29'	)	right parenthesis
X'5E'	X'3B'	;	semicolon
X'5F'	X'AC'	¬	not sign
X'60'	X'2D'	-	minus sign or hyphen
X'61'	X'2F'	/	solidus (slash)
X'62'	X'C2'	Â	capital A with circumflex accent
X'63'	X'C4'	Ä	capital A with diaeresis

X'64'	X'C0'	À	capital A with grave accent
X'65'	X'C1'	Á	capital A with acute accent
X'66'	X'C3'	Ã	capital A with tilde
X'67'	X'C5'	Å	capital A with ring
X'68'	X'C7'	Ç	capital C with cedilla
X'69'	X'D1'	Ñ	capital N with tilde
X'6A'	X'A6'		broken bar
X'6B'	X'2C'	,	comma
X'6C'	X'25'	%	percent sign
X'6D'	X'5F'	_	low line (underscore)
X'6E'	X'3E'	>	greater-than sign
X'6F'	X'3F'	?	question mark
X'70'	X'F8'	ø	small o with slash
X'71'	X'C9'	É	capital E with acute accent
X'62'	X'C2'	Â	capital A with circumflex accent
X'73'	X'CB'	Ë	capital E with diaeresis
X'74'	X'C8'	È	capital E with grave accent
X'75'	X'CD'	Í	capital I with acute accent
X'76'	X'CE'	Î	capital I with circumflex accent
X'77'	X'CF'	Ä	capital I with diaeresis
X'78'	X'CC'	Ì	capital I with grave accent
X'79'	X'60'	`	grave accent
X'7A'	X'3A'	:	colon
X'7B'	X'23'	#	number sign (hash mark, sharp sign)
X'7C'	X'40'	@	commercial at
X'7D'	X'27'	'	apostrophe (single quote)
X'7E'	X'3D'	=	equals sign
X'7F'	X'22'	"	quotation mark (double quote)
X'80'	X'D8'	Ø	capital O with slash
X'81'	X'61'	a	small a
X'82'	X'62'	b	small b
X'83'	X'63'	c	small c
X'84'	X'64'	d	small d
X'85'	X'65'	e	small e
X'86'	X'66'	f	small f
X'87'	X'67'	g	small g
X'88'	X'68'	h	small h
X'89'	X'69'	i	small i
X'8A'	X'AB'	«	angle quotation mark left (<< mark)
X'8B'	X'BB'	»	angle quotation mark right (>> mark)
X'8C'	X'F0'		small eth, Icelandic
X'8D'	X'FD'	ý	small y with acute accent
X'8E'	X'FE'		small thorn, Icelandic
X'8F'	X'B1'	±	plus or minus sign
X'90'	X'B0'	°	degree sign
X'91'	X'6A'	j	small j
X'92'	X'6B'	k	small k
X'93'	X'6C'	l	small l
X'94'	X'6D'	m	small m
X'95'	X'6E'	n	small n
X'96'	X'6F'	o	small o

X'97'	X'70'	p	small p
X'98'	X'71'	q	small q
X'99'	X'72'	r	small r
X'9A'	X'AA'	ª	ordinal indicator feminine
X'9B'	X'BA'	º	ordinal indicator, masculine
X'9C'	X'E6'	æ	small ae dipthong
X'9D'	X'B8'	¸	cedilla
X'9E'	X'C6'	Æ	capital AE dipthong
X'9F'	X'A4'	◊	currency sign (lozenge)
X'A0'	X'B5'	µ	micro sign (small mu)
X'A1'	X'7E'	~	tilde (wavy line)
X'A2'	X'73'	s	small s
X'A3'	X'74'	t	small t
X'A4'	X'75'	u	small u
X'A5'	X'76'	v	small v
X'A6'	X'77'	w	small w
X'A7'	X'78'	x	small x
X'A8'	X'79'	y	small y
X'A9'	X'7A'	z	small z
X'AA'	X'A1'	¡	inverted exclamation mark
X'AB'	X'BF'	¿	inverted question mark
X'AC'	X'D0'		capital D with stroke, Icelandic eth
X'AD'	X'DD'	Ý	capital Y with acute accent
X'AE'	X'DE'		capital thorn, Icelandic
X'AF'	X'AE'	®	registered sign (circled capital R)
X'B0'	X'5E'	^	circumflex accent
X'B1'	X'A3'	£	pound sign (Sterling currency)
X'B2'	X'A5'	¥	yen sign (Nipponese currency)
X'B3'	X'B7'	·	middle dot (scalar product)
X'B4'	X'A9'	©	copyright sign (circled capital C)
X'B5'	X'A7'	§	section sign (S-half-above-S sign)
X'B6'	X'B6'	¶	pilcrow (paragraph, double-barred P)
X'B7'	X'BC'		fraction one-quarter (1/4)
X'B8'	X'BD'		fraction one-half (1/2)
X'B9'	X'BE'		fraction three-quarters (3/4)
X'BA'	X'5B'	[	left square bracket
X'BB'	X'5D'	]	right square bracket
X'BC'	X'AF'	¯	macron
X'BD'	X'A8'	¨	diaeresis or umlaut
X'BE'	X'B4'	´	acute accent
X'BF'	X'D7'	×	multiply sign (vector product)
X'C0'	X'7B'	{	left curly bracket (left brace)
X'C1'	X'41'	A	capital A
X'C2'	X'42'	B	capital B
X'C3'	X'43'	C	capital C
X'C4'	X'44'	D	capital D
X'C5'	X'45'	E	capital E
X'C6'	X'46'	F	capital F
X'C7'	X'47'	G	capital G
X'C8'	X'48'	H	capital H
X'C9'	X'49'	I	capital I

X'CA'	X'AD'		soft hyphen
X'CB'	X'F4'	ô	small o with circumflex accent
X'CC'	X'F6'	ö	small o with diaeresis
X'CD'	X'F2'	ò	small o with grave accent
X'CE'	X'F3'	ó	small o with acute accent
X'CF'	X'F5'	õ	small o with tilde
X'D0'	X'7D'	}	right curly bracket (right brace)
X'D1'	X'4A'	J	capital J
X'D2'	X'4B'	K	capital K
X'D3'	X'4C'	L	capital L
X'D4'	X'4D'	M	capital M
X'D5'	X'4E'	N	capital N
X'D6'	X'4F'	O	capital O
X'D7'	X'50'	P	capital P
X'D8'	X'51'	Q	capital Q
X'D9'	X'52'	R	capital R
X'DA'	X'B9'	¹	superscript one
X'DB'	X'FB'	û	small u with circumflex accent
X'DC'	X'FC'	ü	small u with diaeresis
X'DD'	X'F9'	ù	small u with grave accent
X'DE'	X'FA'	ú	small u with acute accent
X'DF'	X'FF'	ÿ	small y diaeresis
X'E0'	X'5C'	\	reverse solidus (backslash)
X'E1'	X'F7'	÷	divide sign (dot over line over dot)
X'E2'	X'53'	S	capital S
X'E3'	X'54'	T	capital T
X'E4'	X'55'	U	capital U
X'E5'	X'56'	V	capital V
X'E6'	X'57'	W	capital W
X'E7'	X'58'	X	capital X
X'E8'	X'59'	Y	capital Y
X'E9'	X'5A'	Z	capital Z
X'EA'	X'B2'	²	superscript two (squared)
X'EB'	X'D4'	Ô	capital O with circumflex accent
X'EC'	X'D6'	Ö	capital O with diaeresis
X'ED'	X'D2'	Ò	capital O with grave accent
X'EE'	X'D3'	Ó	capital O with acute accent
X'EF'	X'D5'	Õ	capital O with tilde
X'F0'	X'30'	0	digit zero
X'F1'	X'31'	1	digit one
X'F2'	X'32'	2	digit two
X'F3'	X'33'	3	digit three
X'F4'	X'34'	4	digit four
X'F5'	X'35'	5	digit five
X'F6'	X'36'	6	digit six
X'F7'	X'37'	7	digit seven
X'F8'	X'38'	8	digit eight
X'F9'	X'39'	9	digit nine
X'FA'	X'B3'	³	superscript three (cubed)
X'FB'	X'DB'	Û	capital U with circumflex accent
X'FC'	X'DC'	Ü	capital U with diaeresis

X'FD'	X'D9'	Ù	capital U with grave accent
X'FE'	X'DA'	Ú	capital U with acute accent
X'FF'	X'9F'		...

Note: On some printers, a few of the above graphics may not print.

*FS

Contents: A file-save and file-restore program.

Purpose: To save files (sequential or line) on magnetic tape and to restore them from the tape by name with the original line numbers preserved.

Use: FS is invoked by the $RUN command.

Program Key: *FS

Logical I/O Units Referenced:

SCARDS	The source of commands to *FS (defaults to *SOURCE*).
SPRINT	A listing of the directory of files on the tape, prompting messages, and program comments (defaults to *SINK*).
SERCOM	Error comments and help information (defaults to *MSINK*).
GUSER	Responses to prompting messages.
0	The pseudodevice name of the magnetic tape to be used by *FS. If files are to be saved, must be mounted with RING=IN.
1	The tape used for output from COPY commands. This must mounted with RING=IN.
2	Documentation for DSAVE. If not specified, documentation will be read from GUSER.

If GUSER is *explicitly* assigned on the $RUN command and SCARDS is not explicitly assigned, then GUSER will be used to read commands, while replies to prompting messages will be read from *MSOURCE*.

If units 0 or 1 are required, but not specified, the user will be prompted for them. The tapes also may be specified by the FSTAPE, COPY, and UPDATE commands. The tapes may be labeled (VLO or OS/VS, but not ANSI) or unlabeled. If a tape is used with FS, it may not contain any non-FS data. Using PAR=INIT (see below) will destroy all previous FS or non-FS data on the tape.

Parameters: The following parameters may be specified in the PAR field of the $RUN command.

INIT	The INIT parameter must be specified when the tape on unit 0 is used with *FS for the first time. The tape is rewound and two tapemarks are written, indicating the tape is empty.
RECOVER	The RECOVER parameter may be specified when INIT is omitted. With this option, if a record cannot be read, FS will attempt to copy or restore as much of the data as possible in the record. If RECOVER is not specified, the record causing the error is skipped. Using RECOVER may cause a file to be copied or restored incorrectly.
DIST	The DIST parameter specifies that FS is being used in distribution mode (see the description of this mode below).

FIX — The FIX parameter may be used to delete a last incomplete file on a labeled tape. This is used when the program was stopped abnormally in the middle of a save operation causing the last file to be incomplete. This parameter has no effect on an unlabeled tape.

QUIT — The QUIT parameter causes the user to be signed off after *FS terminates if any SAVE or RESTORE operation resulted in an error.

Return Codes:

0 Successful return.
4 Warning messages were issued.
8 Error messages were issued.

Examples:

An initial run with *FS. The files DATAFILE1 and DATAFILE2 are saved.

```
$Mount SAVETAPE *T* Ring=In
$Run *FS 0=*T* Par=Init
Save DATAFILE1 DATACOPY
Save DATAFILE2
Stop Nodismount
```

A subsequent run with *FS. The files DATACOPY (originally the file DATAFILE1) and DATAFILE2 are restored. DATACOPY is restored to the file DATAFILE3.

```
$Run *FS 0=*T*
Restore DATACOPY DATAFILE3
Restore DATAFILE2
List on *PRINT*
Stop Dismount
```

Commands:

The following commands are available for *FS. The underlined portion of each command name may be used as an abbreviation.

The term ”FS tape“ refers to the tape assigned to logical I/O unit 0 or assigned by the most recent FSTAPE command. It is used for all commands except the output from the COPY or UPDATE commands. The term ”copy tape“ refers to the tape assigned to logical I/O unit 1 or assigned by the most recent COPY or UPDATE command. Any parameter enclosed in brackets is optional.

The term ”tape-file“ refers to a copy of an MTS file saved on the tape by *FS. Each tape file is assigned a tape-file name as specified by the user of *FS and a tape-file number which indicates its sequential position on the tape. A tape-file name may be up to 32 characters long and may not contain commas, blanks, or parentheses. Each tape file is assigned a ”version number“ which is 1 for the first file on the tape with a given tape-file name and is incremented by one for each subsequent file saved with the same tape-file name. Either the tape-file number or the combination of tape-file name and version number will uniquely identify a file saved on a FS tape.

Whenever a file number is required, (*L) may be used to refer to the last file on the tape unless noted.

The term ”tape-file“ as used in the command descriptions may be a patterned name similar to those in some MTS commands (see ”File-Name Patterns“ below).

CANCEL tape-file[(version)])
CANCEL [(m)] [(n)]

The CANCEL command cancels all queued RESTORE or COPY command operations for the file(s) named. If no file is specified, all queued RESTORE or COPY commands are canceled. If tape-file numbers are specified, all queued commands for files ”m“ through ”n“ inclusive are canceled. If only ”m“ is given, the pending operation on file ”m“ is canceled. The QDISPLAY command may be used to display the queue. The CANCEL command is the same as the DELETE command.

COPY tape-file[(version)] [newname] [pdn]
[PAR={INIT | RESCAN},DOCUMENT,RECOVER]
COPY [(m)] [(n)] [pdn]
[PAR={INIT | RESCAN},DOCUMENT,RECOVER]
COPY (m) [newname] [pdn]
[PAR={INIT | RESCAN},DOCUMENT,RECOVER]

The COPY command copies files from the FS tape to the copy tape. The files may be specified by a tape-file name (which may be a pattern) or by tape-file numbers. If no parameter is specified, all files on the FS tape are copied to the copy tape.

If ”tape-file“ is specified with a version number, that version is copied; otherwise the latest version of the file is copied. If ”newname“ is specified, the copied file will have the new name; otherwise, the name will be the same as the original.

If the tape-file numbers ”m“ and ”n“ are specified, the files numbered from ”m“ to ”n“ are copied. If ”n“ is omitted, only file ”m“ is copied and a ”newname“ may be specified for it. Version numbers may not be used with tape-file numbers.

Depending on the position of the FS tape and the files to be copied, the copy may be done immediately or be queued for later execution.

PAR=INIT indicates the copy tape is to be initialized before the file is copied. Use of PAR=INIT will destroy any data previously on the copy tape.

PAR=RECOVER indicates that the copy should be completed, if possible, even if an error occurs while reading the input tape. If RECOVER has been given previously with a FSTAPE, COPY, or UPDATE command, it will remain in effect until the next FSTAPE command is issued.

PAR=DOCUMENT indicates documentation for the copied file is to be read from logical I/O unit 2 or GUSER. If the original file has documentation, it is replaced by the new documentation in the copied

version. If the parameter is omitted, any existing documentation is copied with the file.

PAR=RESCAN indicates that FS should rescan the copy tape to rebuild the internal directory of files on the tape. This is useful if the tape has been repositioned or changed outside of FS.

Note: Be careful when copying to a tape of lower density: the files may not fit. A 3480-compatible cartridge tape will hold more than a 6250 BPI 9-track tape, unless there are many short files and short blocks.

E.g.,
```
COPY TEST TEST1 PAR=INIT
COPY DATA(1) *COPY*
COPY ?.LIST ?.DOC PAR=DOC
COPY (10) (13) PAR=DOCUMENT
COPY (11) SAVEFILE *COPYTAPE*
COPY PAR=INIT
```

The last example will make an exact copy to the FS tape.

DELETE tape-file[(version)]
DELETE [(m)] [(n)]

The DELETE command is the same as the CANCEL command.

DLIST [pdn] tape-file[(version)] [SORT=NAME] [ON FDname]
DLIST [pdn] [(m)] [(n)] [SORT=NAME] [ON FDname]

The documentation saved on "pdn" by DSAVE for all of the files named is printed. If "pdn" is not specified, the current FS tape is used. If no files are specified, documentation for all files is printed. SORT=NAME sorts the documentation by tape-file name instead of by tape-file number. ON FDname routes the output to "FDname"; the default is the SPRINT assignment.

E.g.,
```
DLIST TEST(2)
DLIST ?DATA
DLIST (10) (20)
DLIST *T* (1) (10) SORT=NAME
```

DSAVE filename [tape-file])

The DSAVE command is similar to (and may be used in place of) the SAVE command except that up to 2000 characters of "documentation" may be saved with the file. This documentation is read from logical I/O unit 2 if it has been assigned, or GUSER if unit 2 is not assigned, and is terminated by an end-of-file condition. Zero-length lines are ignored.

FSTAPE pdn [PAR={INIT | RESCAN}[,FIX][,QUIT][,RECOVER]]

The FSTAPE command reassigns the FS tape (the tape normally attached to logical I/O unit 0). The PAR parameters are the same as the initial parameters on the $RUN command with the addition of the RESCAN

option which indicates that FS should ignore the internal directory it has for this tape (if any) and rescan it to build a new directory.

E.g., `FSTAPE *T2* PAR=INIT`
`FSTAPE *T1* PAR=RESCAN`

HELP [{command | topic}]

The HELP command provides on-line assistance. Entering simply HELP produces a list of commands and topics for which assistance is available. Entering HELP followed by a command name will provide the proper syntax for the command and a description of its function. The information produced is written to SERCOM and may be presented in full-screen format (see the SET command below).

E.g., `HELP COMMANDS`

LIST [pdn] tape-file[(version)] [SORT=NAME] [ON FDname]
LIST [pdn] [(m)] [(n)] [SORT=NAME] [ON FDname]

The directory information for the specified files on tape "pdn" is printed on "FDname". If "tape-file" is specified without a version, the information for the latest version of the file is listed; if a version is specified, that version of the file will be listed. If "tape-file" is a pattern without a version, all versions of the tape-file names matching the pattern will be listed. If "m" and "n" are specified, all of the files in the range will be listed.

If "pdn" is omitted, the current FS tape is used. If "ON FDname" is omitted, the directory is printed on SPRINT. If SORT=NAME is specified, the listing will be sorted by file name instead of file number.

The directory information includes:

The tape-file number
An "*" if the file has documentation
The tape-file name
The version number of the tape-file
The type of the file (line or sequential)
The maximum record size
The size of the file (number of tracks or pages)
The device type of the file (disk or page formatted)
The date and time the file was saved
The approximate total tape length used up to the end of the current file

E.g.,
```
LIST ON *PRINT*
LIST TEST
LIST ?DATA
LIST TEST(2)
LIST (10) (20)
LIST *T* (1) (10) SORT=NAME
```

MTS

Execution of the *FS program is suspended and a return to MTS command mode is made *without* processing queued FS requests, thus allowing changes to be made to files without terminating the program. Execution may be resumed by a $RESTART command.

QDISPLAY

The QDISPLAY command displays all of the files that are queued to be restored or copied.

QPROCESS

All queued restore and copy requests are processed.

RESTORE tape-file[(version)] [filename]
RESTORE (m) (n) [pdn]
RESTORE (m) [filename]

The RESTORE command restores the tape-files specified to the filenames specified.

Depending on the current position of the FS tape, the restore operation may be carried out immediately or may be queued for later execution.

If a tape-file name is specified with a version, that version will be restored; otherwise, the latest version will be restored. If "tape-file" is a pattern, then "filename" (if specified) must either be a pattern or a pseudodevice name.

If the tape-file numbers "m" and "n" are specified, all of the files in the range "m" and "n" inclusive are restored. The parentheses around "m" and "n" are required. "n" may be omitted, in which case only the mth file will be restored. Even if several files between "m" and "n" have the same name, they *all* will be restored. "filename" may be specified if "n" is omitted, in which case, the file will be restored with the specified file name.

If the restore file does not exist, it will be created at the optimal size, with the access specified by the MTS $SET NEWFILEACCESS option, and will correspond to the device type and file type of the original file that was saved. "filename" may be a pseudodevice name such as *MSINK*, *SINK*, *PUNCH*, or a tape pseudodevice name, in which case, the file(s) will be copied to the specified pseudodevice name. If "filename" is a tape pseudodevice name and the original file was saved from a tape (possible only in distribution mode), the format and data set name will be restored. If "filename" is omitted, the tape-file name (possibly altered or truncated to be a legal MTS file name) is used.

Each file that is to be restored is checked for consistency using the checksum information (see "Internal Format" below). Any checksum inconsistency is flagged with a warning message.

E.g.,
```
RESTORE TEST
RESTORE TEST(2) PROG
RESTORE ?DATA ?OLDDATA
RESTORE (10) (20) *TAPE*
RESTORE (3) (*L)
```

SAVE filename [tape-file]

The SAVE command specifies that the file "filename" is to be saved using "tape-file" as the name of the file on the tape. "filename" may include a line number range, indicating a particular segment of a line file to be saved. If "tape-file" is omitted, "filename" is used as the name of the file. As files are saved, they are assigned version numbers. If the name on the tape is unique, the file is assigned version number 1; otherwise, the file is assigned a version number which is the currently highest version number for that name incremented by 1.

If "filename" is a pattern, all files matching the pattern are saved on the tape in alphabetical order. If "tape-file" is specified when "filename" is a pattern, it must also contain a pattern.

E.g.,
```
SAVE PROG TEST
SAVE PROG.?
SAVE ?DATA ?OLDDATA
```

SET option **...**

The following options may be specified:

HELP={LINE | SCREEN}

This option specifies whether help information provided by the HELP command is printed in line mode or full-screen mode. The default is full-screen mode for terminals that support it.

TERSE={ON | OFF}

This option specifies whether messages from *FS are given in full or abbreviated form. The default is OFF (messages are given in full).

STOP [{DISMOUNT | NODISMOUNT}]

All queued restore and copy requests are processed; then FS returns to the program which called it. DISMOUNT or NODISMOUNT may be specified to control whether or not the tapes used by FS are dismounted. If no parameter is specified in terminal mode, the user will be prompted whether or not the tapes should be dismounted. In both batch and terminal mode, undismounted tapes will be rewound.

In batch mode, a $ENDFILE in the command stream is identical to a STOP command with no parameters. In conversational mode, entering $ENDFILE causes all queued restore requests to be processed; the user is then prompted about continuing execution.

UPDATE tape-file[(version)] [newname] [pdn]
[PAR={INIT | RESCAN},DOCUMENT,RECOVER]
UPDATE [(m)] [(n)] [pdn]
[PAR={INIT | RESCAN},DOCUMENT,RECOVER]
UPDATE (m) [newname] [pdn]
[PAR={INIT | RESCAN},DOCUMENT,RECOVER]

The UPDATE command copies the latest version of files on the FS tape to the copy tape. The action of the UPDATE command is identical to the COPY command, with the exception that only the latest version of each file within the specified range is copied.

E.g.,
```
UPDATE (3) (40) *T1*
UPDATE PAR=INIT
```

The last example places the most recent version of each tape-file onto the copy tape.

VERIFY

The VERIFY command verifies the checksum information on the tape. Any checksum error is flagged with a warning message. The entire tape is read during the verification process.

$cmnd

*FS command lines beginning with a "$" (except for $ENDFILE and $CONTINUE WITH) are executed as MTS commands. After the command is executed, control is then returned to *FS (unless a $RUN or $LOAD command was given).

Comments:

FDname modifiers or explicit concatenation included on the file-name parameter of a SAVE or DSAVE command will be ignored. A segment of a line file may be saved by giving a file line-number range on the SAVE and DSAVE commands. Line numbers (including negative numbers) are preserved for LINE files. File permit codes are not preserved. $CONTINUE WITH lines in the file are saved as data lines, i.e., implicit concatenation of files is not allowed.

Logical unit 0 may be assigned to a tape only, and cannot be reassigned during execution, except with the FSTAPE command.

A tape-file may be queued for only *one* operation at a time.

File names on the tape should be limited to 32 characters. A userID may be concatenated to the name if desired and is included in the 32-character limit. UserIDs on tape names are removed prior to restoration of files.

In order to minimize tape movement, RESTORE and COPY requests are queued when the requested file cannot be reached by forward movement of the tape. Using a LIST command, or any FS command which accepts a version number, without specifying the version number, results in the tape being positioned after the last file, and thus should be avoided until the end, if possible.

The contents of the FS tape are altered through the SAVE, DSAVE, or DRIVE commands only. There has been no provision for deletion of files from the tape by *FS. The COPY or UPDATE commands will, in effect, allow deletion of files by creation of a new FS tape. Note: *FS tapes should be copied only by using these commands unless the *entire* tape is copied.

FS tapes may be used (i.e., their contents may be retrieved readily) only at installations which support MTS. The *FS program and an FS tape may be used to create a tape which can be read at other installations. The procedure is to use the RESTORE command in the form

```
RESTORE (1) (*L) pdn
```

where ”pdn“ is the pseudodevice name of a tape (mounted with RING=IN) on which the converted information will be written. Prior to the conversion run, the record size on the output tape should be set (e.g., via a $CONTROL command) in a manner sufficient to accept the maximum file record length. If ”pdn“ is a labeled tape, the data set names will be set to the same as the tape-file names on the FS tape, modified to be legal data set names.

An attention interrupt is usually acknowledged immediately by *FS. If, however, an attention interrupt is given during a SAVE or COPY operation, the interrupt will not be taken until the operation is completed (since otherwise the tape will not be properly formatted). Should an attention interrupt be given while another attention interrupt is pending, control will be returned immediately to MTS. A $RESTART command can be used to return control to *FS, and *FS will then resume the operation.

Use as a Distribution Program:

*FS may also be used to construct tapes to be used to distribute large sets of programs and data files to other MTS users and installations.

The first step is to construct a ”driver file“ which will contain one line for every program or data file to be distributed. This file may be constructed by the *DEDIT program (see the *DEDIT description in *MTS Volume 2: Public File Descriptions*). The driver file, together with a comment file (also constructed by *DEDIT) will become the description of the contents of the distribution tape constructed by *FS.

To use *FS as a distribution program, DIST must be specified in the PAR field of the $RUN command, e.g.,

```
$RUN *FS 0=*OUT* PAR=INIT,DIST
```

The FS tape is the distribution output tape. The FSTAPE command may be used to change this tape.

PAR=INIT need not be specified if the output tape already contains files in FS format for the distribution.

*FS first prompts for the name of the distribution tape (in order to give a correspondence between lines in the driver file and the output tape used) followed by prompts for premounted input tapes which are entered as

pdn [tape-name] [FMT=fmt]

where "tape-name" is the tape name on the command for mounting the tape, e.g.,

```
*IN* DTAPE
```

Premounted tapes do not remain mounted, but are dismounted when no longer needed. If only a pseudodevice name is given, *FS assumes no tape has been mounted with the pseudodevice name and will use it only if needed. At least one "pdn" must be given. If "FMT=fmt" is specified, the tape is assumed to be of the specified format; otherwise, it may be an FS tape format tape or the format will be taken from the tape labels if it is a labeled tape. An end-of-file condition terminates prompting for premounted tapes, following which the program enters FS command mode. All FS commands are valid in distribution mode.

The following commands are valid only in distribution mode.

DRIVE [filename] [REVL=[a,...]]

The DRIVE command, valid only in distribution mode, may be used to initiate work from the previously constructed driver file. Tapes required, but not premounted, will be automatically mounted (using the pseudodevice names given) according to parameters given in the driver file.

If the REVL parameter is given, it specifies which revision levels from the driver file are to be saved. The value of the REVL parameter is zero or more one-character codes separated by commas. The revision level field from each driver file line (see the *DEDIT description in this volume) is compared against this list and the line is processed only if a match is found. If no REVL parameter is given, the REVL parameter from the previous DRIVE command is used. If the REVL parameter is given with no list of revision levels, all lines in the driver file are processed. This is the default if no REVL parameter is given on the first DRIVE command.

An end-of-file, a severe error condition, or an attention interrupt will suspend the drive. If a DRIVE command with no filename is given at this point, the previous drive is resumed at the line *following* the one causing the suspension. An attention interrupt will stop a drive after the current line is completed.

As files are saved, information is added to columns 138-229 (see the *DEDIT description) of the corresponding lines in the driver file. In addition, information is added to the FS documentation file (but not the FS header)—the information consists of the driver file line, the blocking information if the input is a tape, the first five records of the file being saved, and any $CONTINUE WITH lines, COPY statements, INCLUDE

statements, and REP records found in the file. This information may be listed by using the DLIST command.

If a file is saved from a non-FS tape, corresponding blocking information is saved in the FS header. If this file is subsequently restored to a tape, the saved blocking information will be used to regenerate the tape in its original format.

MTS files or partial files to be saved are indicated by giving one parameter, the MTS file name, in columns 59-117 in the driver line. More than one parameter in the FDname field indicates a file is to be saved from a tape. The tape name and volume name are positional parameters; the keyword parame ters are FSNAME for FS tapes, and POSN and FMT for non-FS tapes. POSN and FMT are optional. A quote string, to be passed to the mount program, may also appear in this field.

FS names are generated as

```
COMP_NAME.COMP_SUB-NAME.TYPE.COMP_SUB-NUMBER
```

On restoration, long file names are compressed to 12 characters by appending the component sub-number to the first 9 characters of the component name.

If the output tape is labeled, each file will have a data set name of

```
COMP_NAME.COMP_SUB-NUMBER
```

DRCOPY filename tape copypar

The DRCOPY command is the same as the COPY command except that it also updates a driver file. It is valid only in distribution mode. ”filename“ must be the name of a driver file which was used to construct the FS tape using the DRIVE command. ”tape“ must be the output tape name in this driver file corresponding to the FS tape. ”copypar“ is any legal combination of parameters for the COPY command.

As each file is copied, the corresponding line is found in ”filename“. This line must have ”tape“ as the output tape name and must have the correct output tape file number, FS name, and version. These four fields are then updated to correspond to the COPY tape.

DRCOPY FINISH

Once a DRCOPY command has been given, all subsequent COPY or UPDATE commands will cause the same driver file to be updated. The DRCOPY FINISH command processes all queued operations and turns this off.

RELEASE pdn

The RELEASE command may be used to dismount a tape that has been mounted by the DRIVE command. ”pdn“ is the pseudodevice name of the tape to be dismounted.

TAPENAME tape-file

The TAPENAME command may be used to change the current output tape to the new output tape ”tape-file“.

COPY and RESTORE (in Distribution Mode)

In distribution mode, if either the RESTORE or COPY command is given with parameters ”(m) (n) filename“, and ”filename“ is a driver file which corresponds to the FS tape, then this RESTORE or COPY command and all subsequent RESTORE or COPY commands which are given ”(m)“ or ”(m) (n)“ as parameters will use as the new name the original name given in the driver file in columns 59-117 of the line corresponding to the input file.

Internal Format of an FS Tape:

FS tapes are written in variable format depending on the density of the tape being used. The variable format is as follows:

8-page blocks (U(32767)) for 6250 BPI tapes and cartridge tapes
4-page blocks (U(16384)) for 1600 BPI tapes

Each file saved onto an FS tape consists of a header block followed by data blocks. There is a tapemark preceding each saved file on the tape. Two adjacent tapemarks terminate the tape.

Format of the Header Block for a File:

```
Identification      (4 bytes)     = X'0FE2AE5E' if no checksum
                                  = X'0EE52AFE' if checksummed
File number         (2 bytes)
Version number      (2 bytes)
File name           (32 bytes)
File type           (1 byte)      = X'00' if a LINE file
                                  = X'01' if a SEQuential file
                                  = X'02' if a SEQWL file (obsolete)
Device type         (1 byte)      = X'00' if a 2311 disk
                                  = X'40' if a DISK (2314 disk)
                                  = X'80' if a data CELL
                                  = X'C0' if a PAGE device
Maximum record size (2 bytes)
File size           (2 bytes)     = size in pages if a PAGE device
                                  = size in TRACKS if not PAGE
Date saved          (14 bytes)    = mmmm dd, 19yy
Time saved          (8 bytes)     = hh:mm:ss
DSN name            (17 bytes)
Blocking info       (17 bytes)
Documentation flag  (1 byte)      = X'FF' if documentation present
Checksum info       (12 bytes)    = 2 bytes - File number (mod 2**16)
                                      with first FS file being file
 (Note: These 12 bytes                number 1.
  are only included if              2 bytes - Block number in the
```

```
   the Identification                  file; file header having block
   field is X'0EE52AFE')               number 0 (mod 2**16).
                                     4 bytes - Block check sum
                                     2 bytes - FS block length
                                       (including check area)
                                     2 bytes - Reserved (=X'0000')
```

Up to 2000 bytes of documentation may be saved with a file. This documentation is contained in the header and follows after the ”documentation flag“ if the file is not checksummed or after the ”checksum info“ if the file is checksummed. Each line of documentation is preceded by a byte indicating the length of the line. The checksum information is used to verify the integrity of the tape (see the VERIFY and RESTORE commands).

Format of a Data Block:

```
If the tape is CHECKSUMmed:
Checksum info       (12 bytes)    = 2 bytes - File number (mod 2**16)
                                      with first FS file being file
                                      number 1.
                                    2 bytes - Block number in the
                                      file; file header having block
                                      number 0 (mod 2**16).
                                    4 bytes - Block check sum
                                    2 bytes - FS block length
                                      (including check area)
                                    2 bytes - Reserved (=X'0000')
Regardless of CHECKSUM:
Data line segments                  As many as will fit to fill up
                                    a block.
```

Format of a Data Line Segment:

```
Switches    (4 bits)          0xxx = Is last segment for this line
                              1xxx = More data for this line follows
                              x1xx = Is first segment for this line
                              x0xx = Not first segment for this line
Line length (12 bits)                Number of bytes in this segment
Line number (4 bytes)                Line number for this line
Data from the line
```

File-Name Patterns:

File-name patterns may used with all FS commands that use file names or tape-file names; e.g.,

SAVE filename-pattern [tape-file-pattern]
LIST tape-file-pattern
RESTORE tape-file-pattern [filename-pattern]

The patterns are specified in the same manner as the patterns used with MTS commands such as $COPY and $FILESTATUS using the question mark "?" as the match character (see *MTS Volume 1: The Michigan Terminal System*). A single "?" will match zero or more characters. To match a fixed number of characters, use multiple question marks (one more than the number of characters to be matched); e.g., "??" will match one character, and "???" will match two characters. For all commands except SAVE and RESTORE, a pattern without a version number specified will execute the command on all files matching the pattern regardless of the version number. For these commands, a single "?" with no filename (to match all files) produces the same result. For the RESTORE command, a pattern without a version number will execute the command on the latest version. If a "filename" option is specified on the RESTORE command and a tape-file pattern is given, the "filename" must contain a pattern or be a pseudodevice name. If a "newname" option specified on the COPY or UPDATE commands and a tape-file pattern is given, the "newname" must contain a pattern.

Examples:

`SAVE WABC:?`	Save all files under userID WABC (with "WABC:" as part of the tape names)
`SAVE WABC:? ?`	Save all files under userID WABC without having "WABC:" as part of the tape names
`RESTORE ?DATA ?OLDDATA`	restores the latest version of all files on the tape that end in "DATA" using names that the end in "OLDDATA"

Given a userID or tape with files named MYPROG.OBJ, PROG2.OBJ, ABC.OBJ, and MYFILE_ONE, then

`SAVE ?.obj`	Saves MYPROG.OBJ, PROG2.OBJ, and ABC.OBJ
`DSAVE my?o? ?sav`	Saves MYPROG.OBJ as PROGSAV and MYFILE_ONE as FILE_SAV
`LIST ????.obj`	Lists information for tape-file ABC.OBJ.
`RESTORE ?prog? ?data?`	Restores both MYPROG.OBJ to MYDATA.OBJ and PROG2.OBJ to DATA2.OBJ

*LABEL

Contents: The MTS magnetic tape-initializing program.

Purpose: To initialize a labeled or unlabeled magnetic tape.

Use: The program is invoked by the $RUN command.

Program Key: *LABEL

Logical I/O Units Referenced:

GUSER	input from the user.
SERCOM	prompting for input and error comments.

Return Codes: Always zero.

Description: The program prompts for the pseudodevice name of the tape to be initialized, the density (1600, 6250, or 38000) at which it is to be initialized, the volume name, and an optional ownerID. This tape must have been mounted with RING=IN specified on the mount request. If the last two parameters (volume name and ownerID) are omitted, the tape is initialized as an unlabeled tape, i.e., six filemarks are written at the specified density at the beginning of the tape and the tape is rewound. If the volume name is given (1 to 6 characters without embedded blanks or commas), the tape is initialized as a labeled tape, is rewound, and label processing in enabled. The specified volume name must then be used for all future mounts of the tape. An optional ownerID (the fourth parameter) will be included in the volume label if given. It must not be longer than 10 characters.

If a volume label is given, the tape will be labeled according to the IBM OS/VS magnetic-tape labeling standard, which is the standard in MTS. However, if the keyword parameter LBLTYPE=ANSI is specified as the last parameter (after the ownerID, or volume name if the ownerID is omitted), the program will write the volume label using the American National Standards Institute (ANSI) format. The tape should be ANSI-labeled if it is to be exchanged with a computer installation that does not support the IBM standard, but does support the ANSI standard.

The program will then prompt for another set of parameters to initialize a second tape. An end-of-file will terminate the program.

Examples:

```
$MOUNT MYTAPE *TAPE* RING=IN
$RUN *LABEL
*TAPE* 1600 0039JR UNIVOFMICH
$ENDFILE
```

In the above example, the tape *TAPE* is initialized to 1600 BPI. The tape is labeled with a volume name 0039JR and an ownerID UNIVOFMICH. The IBM OS/VS labeling standard is used.

```
$MOUNT MYTAPE *** RING=IN
$RUN *LABEL
*** 1600 DATA81 LBLTYPE=ANSI
$ENDFILE
```

In the above example, the tape *** is initialized to 1600 BPI. The tape is labeled with a volume name DATA81 using the ANSI-standard label type. The ownerID field will be blank.

*LABELSNIFF

Contents: The tape-identification program.

Purpose: To print tape information in an intelligible format.

Use: The program is invoked by the $RUN command.

Program Key: *LABELSNIFF

Logical I/O Units Referenced:

SPRINT	tape information output.
SERCOM	the message "Enter tape:" as well as confirmations and error comments.
GUSER	input for tape names, if the PAR field is omitted or if unit 0 is not specified.
0	tape to be scanned.

Parameters: The following parameters may be specified in either the PAR field of the $RUN command or in any GUSER input line. The parameters must be separated by a blank.

TAPE=xxx — The name of the tape to be scanned, e.g., TAPE=*T*, if unit 0 is not specified.

FILES=xxx — The number of data sets on the tape to be scanned, e.g., FILES=10. If this parameter is omitted, all of the data sets will be scanned.

FS / NOFS — FS enables the automatic recognition and processing of *FS tapes. NOFS disables this processing. The default is FS.

LP / NOLP — If LP (label processing) is specified, the tape labels will be read by the system tape routines and the tape information will be passed to *LABELSNIFF for interpretation. If NOLP is specified, *LABELSNIFF will attempt to read and interpret the label information. The default is LP. Pool tapes will always be read by the system tape routines.

OS / NOTOS — If OS is specified, the LRECL (logical record length) in V and D record formats is incremented by 4. If NOTOS is specified, the LRECL remains consistent with the MTS tape conventions. The default is NOTOS.

POSN=xxx — This overrides the rewinding of the tape before the processing starts. The tape is positioned to the beginning of the specified data set, if possible. Otherwise, the tape will be rewound. Data set numbers or names may be entered immediately after the equal sign, e.g., POSN=*2* or POSN=COUNT.

REW	The tape is rewound to the beginning of the tape after the processing is completed (the default).
NOREW	The tape is positioned at the logical end of the tape or after the last data set scanned. If an attention interrupt occurs, and if the user wishes to discontinue, the tape is positioned just before the current data set.
ENGLISH	The approximate tape length is given in feet (the default).
METRIC	The approximate tape length is given in meters.
SKIP	All data blocks are skipped. The program does not provide vital information such as record count, tape length, and average and maximum block lengths.
SHORT	A short column format, with only data set name, file number, block count, record count, tape length, and record format, is provided.
LONG	A long column format provides in addition to the short column format described above, average and maximum block lengths, creation date, expiration date, tape mode, userID, and batch receipt number.

The default for SHORT or LONG depends on the maximum output length of the logical I/O device unit SPRINT.

If the tape name is a pseudodevice name, e.g., *T*, or begins with the current device character, e.g., >T901, then the tape name may appear in a GUSER input line, or in the PAR field of the $RUN command. Otherwise, the name must follow the TAPE parameter.

Commands:

PRINT xxx	If "xxx" is "LABELS", then every label read in will be echoed to SPRINT. Labels will be printed only if system label processing is not in effect or if the NOLP parameter is specified. Labels of pool tapes cannot be printed. Normally this echoing will be suppressed.
MCMD xxx	The subroutine MTSCMD will be called with the input string "xxx" (see *MTS Volume 3: System Subroutine Descriptions*).
MTS xxx	The subroutine CMD will be called with an input string "xxx". If no string is specified, the subroutine MTS is called. In particular, the $CONTROL command may be used on tapes. See *MTS Volume 3* for a description of these subroutines.
RETURN	Control returns to MTS command mode. The program may be restarted using the $RESTART command.

STOP	The program is terminated and may not be restarted. An end-of-file encountered on GUSER also terminates the program.
COMMENT	This command is used only for annotation of the program processing. The command is ignored by the program.
$xxx	The command is taken as an MTS command.

Return Codes:

0 Successful return.
4 Minor errors encountered (such as syntax errors in command keywords and parameters).
8 Nonfatal tape error encountered.
12 Fatal tape error encountered.
16 System error encountered.

Description:

If the PAR field in the $RUN command is omitted or if unit 0 is not specified, a message ”Enter tape:“ appears on SERCOM, and input is read from GUSER. Only one tape name can be specified in any GUSER line, on unit 0, or in the PAR field. To continue the current GUSER line onto the next line, one may enter the current MTS command line continuation character (normally a minus sign) as the *last* character of the input line.

If the POSN parameter is given, the program attempts to position the tape according to the parameter.

If the POSN parameter is not given, the tape is rewound to the beginning. The program automatically checks to determine whether the tape is IBM, ANSI, or not labeled.

In both cases, the program prints the general information of the tape which includes the tape name followed by the current date and time, the number of tracks, the density, the type of label, the volume serial number, the tape name, the owner's name, and the tape status. As an example:

```
Tape name = *T* 09:39:05 22 Sept 1976
IBM labeled 1600-BPI 9TP
Volume=SAMPLE Owner=JOHN.DOE Tape=MYTAPE
LP=on BLK=on RING=out DTCHK=on RETRY=4
```

The program then proceeds to print the characteristics for every data set that exists on the tape:

```
File Data set     Block Record  Tapelen    Record
#    Name         Count Count    (Feet)    Format
1   S4.0           48   1900     11.75  F(3200,80)
2   S5.0           10    900      4.00  VB(8000,84)
```

A full description of data set characteristics is given below. The program automatically detects if a tape in process was produced from the public file *FS. The output information will then be similar to that of *FS.

Each time a block is read that results in an error, a comment will be printed:

```
Tape error:  RC=nn, BLK# nnn, File # nnn, SENSE=xxx
```

The return code is always shown. The block number and file number referred to are block and file numbers on the tape. Sense information is given in hexadecimal form only if valid. If error occurs while reading a header or trailer record, HDR# or TLR# is given instead of BLK# in the message above.

When the end of the tape is detected or when the specified number of files have been scanned, the program prints the total tape length in feet or meters. This is useful for determining the amount of unused tape remaining. If the end of the tape was reached, a comment is printed. Then, unless the NOREW parameter is given, the program rewinds the tape back to its beginning; otherwise, the tape is correctly positioned (such as at the logical end). The program then proceeds to process the next tape or terminates if there are no more tapes.

If the user gives an attention interrupt, *LABELSNIFF will respond with:

```
**Attention interrupt**
```

or

```
Attn at file 2 BLK# 57
```

If the former message appears, no tape is currently attached. For the latter, the program shows which file and which block it has read so far. When it is reading labels, the words ”HDR“ or ”TLR“ replace ”BLK“ for header or trailer labels, respectively. The program then will prompt:

```
Do you want to continue?
```

The user must type ”YES“ to continue, ”NO“ to end processing of the tape, or ”end-of-file“ to terminate the program.

Data Set Characteristics:

Data set name	The name of the data set on the tape.
File #	The data set number on the tape, i.e., the logical file on a labeled tape.
Created dd mmm yy	The creation date of the data set.
Expires dd mmm yy	The expiration date of the data set. Omitted if the date is zero.
User I.D.	In MTS, this shows the userID of the creator of the data set. In OS, it is the job name.
Batch receipt #	In MTS, the batch job number is shown if any. In OS, it is the job step name.

Record format fmt(blklen,reclen)
: The record format of the data set. The first letter in fmt shows the format type: undefined (U), fixed (F), variable in EBCDIC (V), or variable in ASCII (D). This letter may be followed by attributes: blocked (B), standard or spanned (S), printer controls American National Standard (A), or Machine (M). Block length (blklen) and record length (reclen) are also given here.

Buffer offset
: The length of the ANSI block prefix; from 0 to 99.

Block count
: The number of blocks in the data set.

Record count
: The number of records in the data set with the record format F or V.

nn read errors
: The number of block read errors in the data set. Generally, this message does not occur.

Block len
Av. max
: Average and maximum block lengths are summed up for every data set with record format other than F (fixed).

Block count discrepancy: count recorded is nn
: The count recorded in the trailer label is not the same as the actual block count, i.e., the number of blocks in the data set.

Continued on another tape
: Program found that the data set is a multivolume data set.

Tapelen (feet)
: The tape length of the data set (including its blocks and labels). This shows approximately how large a data set is and depends on certain tape characteristics such as the number of blocks, size of blocks, tape density, etc. It may also be given in terms of meters if the parameter METRIC is specified.

*FS file
: This is the *FS tape-file number.

*FS filename
: This is the *FS tape-file name.

*FS ver#
: This is the *FS file version number.

File type
: The type of file saved (LINE=line, SEQ=sequential file, or SEQWL= sequential-with-line-numbers file).

Dev type	The type of device the file was saved from (2311=2311 disk drive, DISK=disk drive, CELL=data cell, PAGE=paged disk).
Max rec.	The maximum record size in the saved file.
File size	The size of the saved file.
Date saved	The date the file was saved.

Examples:

```
$RUN *LABELSNIFF PAR=*TAPE*
```

The above example will process the tape *TAPE*. All of the data sets on the tape will be processed; the tape will be rewound before and after the processing is completed.

```
$RUN *LABELSNIFF
*T1*
*T2*
$ENDFILE
```

The above example will process the tapes *T1* and *T2* in the same manner as the first example.

```
$RUN *LABELSNIFF
*P* POSN=*3*
*T* FILES=10 NOREW
PRINT LABELS
*T* POSN=ABC NOLP
$ENDFILE
```

The above example will position the tape *P* to the beginning of the third data set; all of the following data sets on that tape will be processed. The first ten data sets on tape *T* will be processed and the tape will not be rewound; the printing of tape labels will commence, the tape will be positioned to data set ABC, and the remainder of the data sets on the tape will be processed.

*TAPECOPY

Contents: The MTS tape-copying program.

Purpose: To copy magnetic tapes.

Use: The program may be invoked with the $RUN command or may be called as a subroutine from a user-supplied program. In the latter case, the user must concatenate *TAPECOPY with the FDname(s) containing the object modules of his own program, e.g.,

```
$RUN object+*TAPECOPY
```

Program Key: *TAPECOPY

Logical I/O Units Referenced:

SPRINT	the listing of the number of blocks or records in each file.
SERCOM	error comments and prompting messages.
GUSER	input and output tape pseudodevice names.
0	the input tape pseudodevice name.
1	the output tape pseudodevice name.

Parameters: The following parameters may be specified in the PAR field of the $RUN command. The parameters must be separated by commas.

FILES=n — The number of files (or data sets) to be copied is specified by ”n“.

RECORDS=m or BLOCKS=m — The number of blocks or records (see REBLK parameter) to be copied in file n+1 is specified by ”m“, where ”m“ is a one to six digit decimal integer. If ”n“ and ”m“ are both zero or omitted, then all files are copied (see below).

NOREW — The tapes are not rewound before or after copying.

OMITERRORS — Whenever a permanent read error is encountered on the input tape, an error comment (indicating the file and record in error) will be printed on SERCOM. The block in error will be skipped and copying will continue with the next block.

COPYERRORS — This parameter causes the same error comment as OMITERRORS. However, any blocks which are in error *are* copied. Any block so copied will no longer contain the error indication; however, the validity of the data in any such block is unpredictable.

REBLK — This parameter will cause blocking to be enabled for both the input and output tapes so that records, rather than blocks, are copied. Normally, blocking is disabled so that

entire blocks are copied (for efficiency).

REW — The tapes are rewound before and after copying.

Return Codes:

0 Successful return.
4 Invalid parameter.
8 Fatal error.

Description:

The tape-copying program copies blocks (or records if the REBLK parameter is given) and end-of-files from the input tape (unit 0) to the output tape (unit 1). If units 0 and 1 are not assigned on the $RUN command, the program will prompt for the input tape name first and the output tape name second. If no FILES=n and RECORDS=m parameters are given, then the copying continues until the logical end of a labeled input tape is reached or until two consecutive end-of-files are read from an unlabeled input tape. If the FILES=n and/or RECORDS=m parameters are given, then "n" files and the first "m" blocks (or records if the REBLK parameter is given) of file n+1 will be copied. If "n" and "m" are zero, then all files are copied as if no FILES and RECORDS parameters had been given.

If the REW parameter is given, then both tapes are rewound before and after copying. If the NOREW parameter is given, then *no* rewinding takes place either before or after copying. If neither parameter is given, then REW is assumed except that terminal users are requested to first confirm the rewinding if the output tape is not already rewound.

If neither of the parameters OMITERRORS or COPYERRORS was given and a permanent (unrecoverable) read error is sensed, the input tape is left positioned after the error record, the output tape after the preceding record, an error comment is printed, and the system subroutine ERROR is called.

The output tape must be mounted with WRITE=YES (or RING=IN) specified on the mount request.

If both the input and output tapes are labeled, the program will preserve any nondefault data set names. If the NAME tape control command has been used to specify a file name for the output tape, this name will be used for the *first* data set copied onto the output tape. *Unless* the REBLK parameter is given, if both tapes are labeled, the blocking parameters (format, logical record length, and block length) will be preserved. All other information on the tape label is regenerated by the tape routines and hence may be different (e.g., creation date, expiration date, tape density).

If the input tape is unlabeled and REBLK is not specified, the output tape will be written with a block size of 32767. If a different block size is desired, the format should be set on both the input and output tapes and the REBLK parameter should be specified.

Users should be aware that the MTS tape routines will automatically translate ASCII data on ANSI tapes to or from EBCDIC as the tapes are read or written by this program. For this reason, data copied from an ANSI tape to a standard MTS tape will be translated from ASCII to EBCDIC. If the ANSI tape is

blocked, it is strongly recommended that the REBLK parameter be specified. The user may inhibit the ASCII/EBCDIC or EBCDIC/ASCII translation by applying the @BIN modifier to the input or output pseudodevice names, respectively.

The file number field, the creation date, the expiration date, and the userID field in the tape labels are automatically updated to the current their current values when the tape is copied. That is, the creation date is the current date, the expiration date is set to the default date, and the userID is the current userID.

The entry point to the tape-copying program is TAPCPY. General register 1 should be set up to provide TAPCPY with a parameter region. Register 1 should contain the location of a fullword address. This address is the location of a halfword character count which is immediately followed by the EBCDIC characters of the parameter. If no parameter is to be passed, then the halfword count should be zero.

Examples:

```
$RUN *TAPECOPY 0=*IN* 1=*OUT*
```

In the above example, all files on the tape *IN* are copied to the tape *OUT*. In batch mode, both tapes are rewound before and after copying. In conversational mode, the user is prompted for permission to rewind the tapes unless the output tape is already rewound.

```
$RUN *TAPECOPY 0=*OLD* 1=*NEW* PAR=FILES=6,REW
```

In the above example, the first 6 files on tape *OLD* are copied to tape *NEW*. Both tapes are rewound before and after the copying is done.

*TAPEDUMP

Contents: The MTS tape (or file) dumping program.

Purpose: To dump a magnetic tape or disk file in both character and binary formats.

Use: The program is invoked by the $RUN command.

Program Key: *TAPEDUMP

Logical I/O Units Referenced:

SPRINT	dump output.
SERCOM	prompting and error messages.
GUSER	tape or file to be dumped if unit 0 is not assigned.
0	magnetic tape or file to be dumped.

Parameters: The following parameters may be specified in the PAR field of the $RUN command. The parameters must be separated by commas.

FILES=n	Specifies the number of files to be dumped.
RECORDS=m	Specifies the number of records to be dumped.
NODUMP	Suppresses the printing of the actual dump.
EBCDIC	Prints the dump in hexadecimal/EBCDIC.
BCD	Prints the dump in octal/BCD.

Return Codes: Always zero.

Description: The file or device to be dumped may either be assigned to unit 0 on the $RUN command or may be entered via GUSER. In the latter case, *TAPEDUMP will prompt for another file or device name at the end of each dump. An end-of-file will terminate the program.

*TAPEDUMP will dump a specified number of files and records in either BCD and octal or EBCDIC and hexadecimal. The number of files and records to be dumped as well as the format are controlled by the PAR field of the $RUN command.

Specifying "FILES=n" and/or "RECORDS=m" will cause "n" files and "m" records to be dumped. If only a "FILES=n" parameter is given, then "n" complete files will be dumped. If both "FILES=n" and "RECORDS=m" are given, then "n" files plus the first "m" records in the file n+1 will be dumped. If neither of these parameters is given, then the dumping will continue until the logical end of the tape is reached (for labeled tapes), until two consecutive end-of-files are encountered (for unlabeled tapes), or until one end-of-file is encountered (for files and other devices).

The parameter NODUMP will suppress the printing of the dump portion of *TAPEDUMP output. The header line which precedes each record and which gives the file and record numbers, the record length, density, mode, and line number will continue to print, but the data dump will not.

BCD or EBCDIC will force the dump to be in the appropriate format. If BCD format is requested, the two high-order bits of each data byte are masked off and the dump is printed in octal instead of hexadecimal. All nonprinting graphics in the EBCDIC (or BCD) portion of the dump print as the character period.

The BCD character set uses the IBM scientific (ALTERNATE) BCD graphics.

Examples:

```
$RUN *TAPEDUMP 0=*TAPE* PAR=FILES=3,EBCDIC
```

In the above example, the first three files on the tape *TAPE* are dumped in hexadecimal format.

```
$RUN *TAPEDUMP SPRINT=*PRINT* PAGES=100
DATA.36
SSGC6
$ENDFILE
```

In the above example, the disk files DATA.36 and SSGC6 are dumped on *PRINT*. The MTS local page limit on the $RUN command will limit the dump to 100 printed pages.

*TAPEFIXER

Contents: A program to recover data from a labeled tape which has been partially destroyed or contains unreadable blocks.

Use: The program is invoked by the $RUN command.

Program Key: *TAPEFIXER

Logical I/O Units Referenced:

SPRINT	labels encountered on the input tape (see the parameter descriptions below).
SERCOM	diagnostic and prompting messages.
0	input tape to be scanned. This must be a tape and cannot be a pool tape since label processing cannot be disabled for pool tapes.
1	corrected version of input tape. The tape may be either labeled or unlabeled. It need not be positioned at the load point. The tape must be mounted with RING=IN specified on the mount request.

Parameters: The following parameters may be specified in the PAR field of the $RUN command.

LABELS	Labels encountered on the input tape will be echoed on SPRINT. This is the default.
NOLABELS	Labels encountered on the input tape will not be echoed on SPRINT.
NOTEXT	Data records are not copied to the output tape.
TEST	When searching for a label, all blocks are inspected, not just the blocks that are exactly 80 bytes long.

Return Codes:

0	Successful return.
8	Error return.

Description: The program scans the input tape for IBM standard labels in their appropriate format and sequence. When errors are encountered, the user is prompted to specify alternative corrective actions to be taken. The primary purpose of the program is to scan tapes originally produced by a system such as MTS or OS; however, it may be used to verify the correctness of user-produced labeled tapes.

The user must be familiar with the basic structure of a labeled tape in order to use this program. All messages generated by the program will refer to this structure explicitly. If the user is not familiar with the structure, he should consult the above references.

At the start of execution, the program rewinds the input tape and disables label processing. The program does not subsequently rewind the tape or enable

label processing. The output tape is *not* rewound prior to the initiation of the scan nor is it rewound at the end of execution. Blocking is disabled for the output tape and is not subsequently enabled.

The input tape is first examined for the presence of a legal volume label. If a volume label is not found, a message is printed to that effect on SERCOM and the scan of the input tape proceeds to the first file on the tape. If a legal volume label is found, it is echoed on SPRINT, unless NOPRINT has been specified. The program does not initialize the output tape; therefore, the output tape, if it is to be labeled, should be initialized with a volume label and be properly positioned. The volume label of the input tape is *not* copied to the output tape.

The scan of the remaining portion of the tape will result in one the following situations on a file-by-file basis:

(1) If both the header and trailer labels of the file are intact, the file is copied to the output tape verbatim. The header and trailer labels are echoed on SPRINT, unless NOPRINT has been specified. Note that it is impossible to reblock the tape or respecify the data set name. If the file to be copied is empty, i.e., there are no data records, the user is prompted to specify whether the file is to be copied to the output tape or deleted. Responses to the prompt are:

COPY	copy the file
DELETE	delete the file

If unreadable records are encountered on the input tape, the user will be prompted to specify whether the record should be copied to the output tape, deleted from the output tape, or whether the entire file should be deleted from the output tape. Responses are:

COPY	copy the record as read
DR	delete the record
DF	delete the entire file

(2) If both header labels are valid, but one or more of the trailer labels are missing, invalid, or unreadable, the input file will be copied to the output tape as found and trailer labels corresponding to the header labels will be generated. A message will be printed on SERCOM indicating that these labels have been generated.

(3) If one or more header labels are missing, invalid, or unreadable, the input tape position is noted and the tape is forward-spaced past the next end-of-file. The following records are read. If the next records are HDR1 or HDR2 labels, the data is skipped over and is unrecoverable. If the record is an EOF1 label and is immediately followed by an EOF2 label, the data skipped by the forward spacing may be recovered by using the information contained in the trailer labels. The user is prompted to specify whether the file should be recovered or deleted. The responses are:

RECOVER recover the file
DELETE delete the file

If RECOVER is specified, the input tape is repositioned at the beginning of the recoverable data. The data are copied to the output tape with header labels generated using information from the trailer labels.

(4) This case is identical to case (3) except that the trailer labels correspond to the header labels of the previous file. This can occur, for example, if a tapemark is written in the middle of an otherwise valid file. If this should occur, the user has four options:

(a) the currently recoverable data may be appended on the output tape to the data previously recovered for the "front part" of the file as in case (2),
(b) the currently recoverable data may be written on the output tape as a separate file (if this option is taken, there will be two files with the same data set name on the output tape),
(c) the currently recoverable data may be deleted from the output tape, or
(d) the entire file, i.e., both parts, may be deleted.

Responses to the prompting message are for each case:

(a) APPEND
(b) NEWFILE
(c) D2
(d) DF

Miscellaneous consistency checks:

(1) When two or more consecutive tapemarks are encountered, the user will be prompted to specify whether to continue. The responses are:

YES
NO

The program will not recognize EOV labels. When MTS terminates a tape, the system writes 5 consecutive tapemarks. If the user has inadvertently written tapemarks on the tape, it may be necessary to skip over 5 or more such marks. However, since this may also be the end of the tape, discretion must be used.

(2) If tapemarks are missing where required, e.g., after a HDR2 label and before the data, they will be assumed.

(3) If redundant or unexpected tapemarks are found, they will be ignored, subject to previously stated conditions.

(4) If the block count of the EOF1 label does not match the actual number of blocks copied from the input tape to the output tape, a

warning message will be printed to that effect.

(5) If an EOF1 or EOF2 label does not match the corresponding HDR1 or HDR2 label, a warning message is printed and the invalid label is corrected.

(6) If an unreadable record is encountered on input, an error message is written on SERCOM and the user is prompted whether or not to continue.

If the output tape is unlabeled, none of the labels encountered by the program will be written to the tape.

*TAPES

Contents: A magnetic-tape status program.

Purpose: To allow magnetic-tape users to PERMIT, RENAME, and find out the status of magnetic tapes in the possession of ITD.

Use: The program is invoked by the $RUN command.

Program Key: *TAPES

Return Codes: Always zero.

Description: *TAPES is a program for changing the name and permission status of user-owned magnetic tapes, and for displaying the results of those changes.

Commands: The following commands are available for *TAPES. The underlined portion of each command name may be used as an abbreviation.

STATUS [tapename] [options **...**]

The STATUS command gives information about a user's own tapes and of tapes of other users that are accessible. The syntax of this command is very similar to the MTS $FILESTATUS command. If no parameter is given, information is given about all tapes. If "tapename" is specified, information is given about a specific tape or set of tapes. "tapename" may be the name of a tape (optionally preceded by a userID), a question mark, or a question mark preceded by a userID. For example,

```
STATUS MYTAPE
STATUS WABC:MYTAPE
STATUS ?
STATUS WABC:?
```

"options" may be specified to produce specific information or to control the output format or destination. "options" may be one or more of the following:

To select specific items:

ACCESS	COMMENT
CREATE DATE	DENSITY (or MODE)
DRIVES	IDLE DAYS
LASTCHG	LAST MOUNTED
LASTREF	LAST USER
LOCATION	MEDIA
MOUNT RESULT	NAME
OWNER	PDN
RACKNUMBER	RECEIPT
REFERENCES	RING STATUS
STATUS	TAPE NAME

To select groups of items:

SHORTINFO TOTAL

To control formatting:

COLUMNS	HEADING
INDENT	KEYWORD
LABELLED	NOHEADING
PACKED	SPACING
TERSE	VERBOSE

To change the output destination:

OUTPUT=FDname

For example, to display all information about all tapes on *PRINT*, enter

```
STATUS ? TOTAL OUTPUT=*PRINT*
```

To display the last time the tape F000:F000T1 was mounted with write-protection disabled, enter

```
STATUS F000:F000T1 LASTCHG
```

PERMIT tapename [access [accessor]]
PERMIT tapename LIKE tapename

The PERMIT command allows the user to permit a tape in a manner similar to the MTS $PERMIT command. The syntax takes the same form as well with the following differences.

When permitting a tape "LIKE" another tape, only one tape name may appear after "LIKE". For example,

```
PERMIT TAPE4 LIKE TAPE1
```

The valid access types for tapes are as follows:

WRITE	READ
RW	RENAME (or RNA)
RUN	EDIT
DEFAULT	DESTROY
UNLIM	PERMIT
FULL	NONE

Accessors generally are the same as for files:

ALL	ME
OTHERS	OWNER
userID	PROJECT=projectID

Program key access is not supported.

For example, to permit TAPE1 READ to user WXYZ, enter

```
PERMIT TAPE1 R WXYZ
```

MODIFY tapename option

The MODIFY command changes the pseudodevice name (PDN) for a tape (the default is *T*). For example,

```
MODIFY TAPE1 PDN=*OUT*
```

The above example changes the PDN for TAPE1 to *OUT*. The MODIFY command also can change any comments associated with a tape. For example:

```
MODIFY TAPE2 COMMENT="Includes data from TAPE1"
```

The comment string must be enclosed in double quotes and be less than 30 characters in length.

RENAME

The RENAME command takes the same form as the MTS $RENAME command with one difference, confirmation is not allowed. For example to rename TAPE1 to be POPULATION_DATA, enter

```
RENAME TAPE1 TO POPULATION_DATA
```

Note: The RENAME command may also be used to transfer ownership of a tape to another user just like the MTS $RENAME command.

STOP

The STOP command terminates execution of *TAPES.

MTS

The MTS command returns control to MTS command mode. $RESTART can be used to resume execution of this program.

*TAPESUBMIT/*TAPERETRIEVE

Contents: The magnetic-tape submission and retrieval programs.

Purpose: To guide a magnetic-tape user through the tape-submission and retrieval process used by ITD.

Use: The programs are invoked by the $RUN command.

Program Keys: *TAPESUBMIT
*TAPERETRIEV

Logical I/O Units Referenced:

SERCOM	error comments, prompting messages, and summary information.
GUSER	user responses to prompting messages (if any).

Return Codes: Always zero.

Description: TAPESUBMIT is the program that must be run by all magnetic-tape users who wish to submit magnetic tapes to ITD for subsequent mounting and use. Depending on the options selected, the user will supply and the program will collect all of the information necessary for a magnetic tape to be initialized (optional) and placed in the tape racks at the Computing Center.

The following commands may be used with TAPESUBMIT. Single commands may also be entered in the PAR field of the $RUN command. Allowable abbreviations are indicated by underlining.

DELETE receipt-code

The DELETE option deletes the entry specified by ”receipt-code“. This option may be used to purge explicitly an erroneous entry. Alternatively, an erroneous entry is treated as a ”no-show“ if the corresponding tape is not submitted and is automatically purged from the system within three weeks.

DISPLAY

The DISPLAY option prints information about the tapes for which the user has run TAPESUBMIT, but have not yet been entered into the tape database.

HELP topic

The HELP command gives detailed information about the requested topic or about obtaining more help if no topic is given. Possible topics include: any other TAPESUBMIT commands, PROCEDURES, GLOSSARY, BULKINPUT, or EXAMPLES.

MTS

The MTS command returns control to MTS command mode. TAPESUBMIT may be re-entered with the MTS $RESTART command.

SUBMIT [{NEW | OLD | TERSE | VERBOSE}]
SUBMIT [INPUT=filename]

The SUBMIT command initiates the tape-submission process and has two forms.

The first form above is used to submit a small number of tapes. The user will be asked and should respond to a number of questions concerning the tape to be submitted. By specifying the NEW or OLD option, a few questions necessary to determine the status of the tape will be skipped, shortening the process. If the tape has been used prior to this submission but there is no further need to access the data on the tape, then NEW may be specified or both NEW and OLD may be omitted. Once the user has become familiar with the questions and possible answers, the TERSE option may be specified to eliminate the detailed questions. Only use the TERSE option after first running through a submission without the option.

The second form above is used to submit a large number of tapes. For this method the information about each tape is contained on a single line in a file. The INPUT keyword of the command specifies the name of the file. All lines in the file that do not properly describe a tape will be ignored and processing will continue with the next line.

Information collected using the two forms of this command will be recorded for future use when the tapes are actually delivered to the ITD operations staff.

STOP

The STOP command terminates execution of the program.

$mts-command

Any command beginning with a dollar sign ($) is interpreted as an MTS command.

TAPERETRIEVE is the program that must be run when a user wishes to retrieve a magnetic tape that has been previously submitted to ITD. For further information on the tape-submission and retrieval procedures, see *Submitting, Retrieving, and Permitting Magnetic Tapes*, Tutorial T7012.

SYSTEM SUBROUTINES FOR MAGNETIC TAPES

User programs may perform tape operations by calling the following system subroutines. Their descriptions are given in *MTS Volume 3: System Subroutine Descriptions*.

ASCEBC	Translates ASCII characters into EBCDIC characters.
CONTROL	Performs control commands.
DISMOUNT	Dismounts tapes.
EBCASC	Translates EBCDIC characters into ASCII characters.
MOUNT	Mounts tapes.
REWIND#	Rewinds tapes.
SKIP	Skips forward or backward by record or file.
TAPEINIT	Initializes (empties) labeled or unlabeled tapes.

INDEX

Reader's Comment Form

Magnetic Tapes

October 1993

Errors noted in publication:

Suggestions for improvement:

Date _____

Name _____

Address _____

Your comments will be greatly appreciated. Please fold the completed form as shown on the reverse side, seal or staple, and drop in Campus Mail or in the Suggestion Box at any Campus Computing Site.

fold here

Communication Group
ITD User Services
University of Michigan
535 West William
Ann Arbor, Michigan 48103-4943
USA

---

fold here