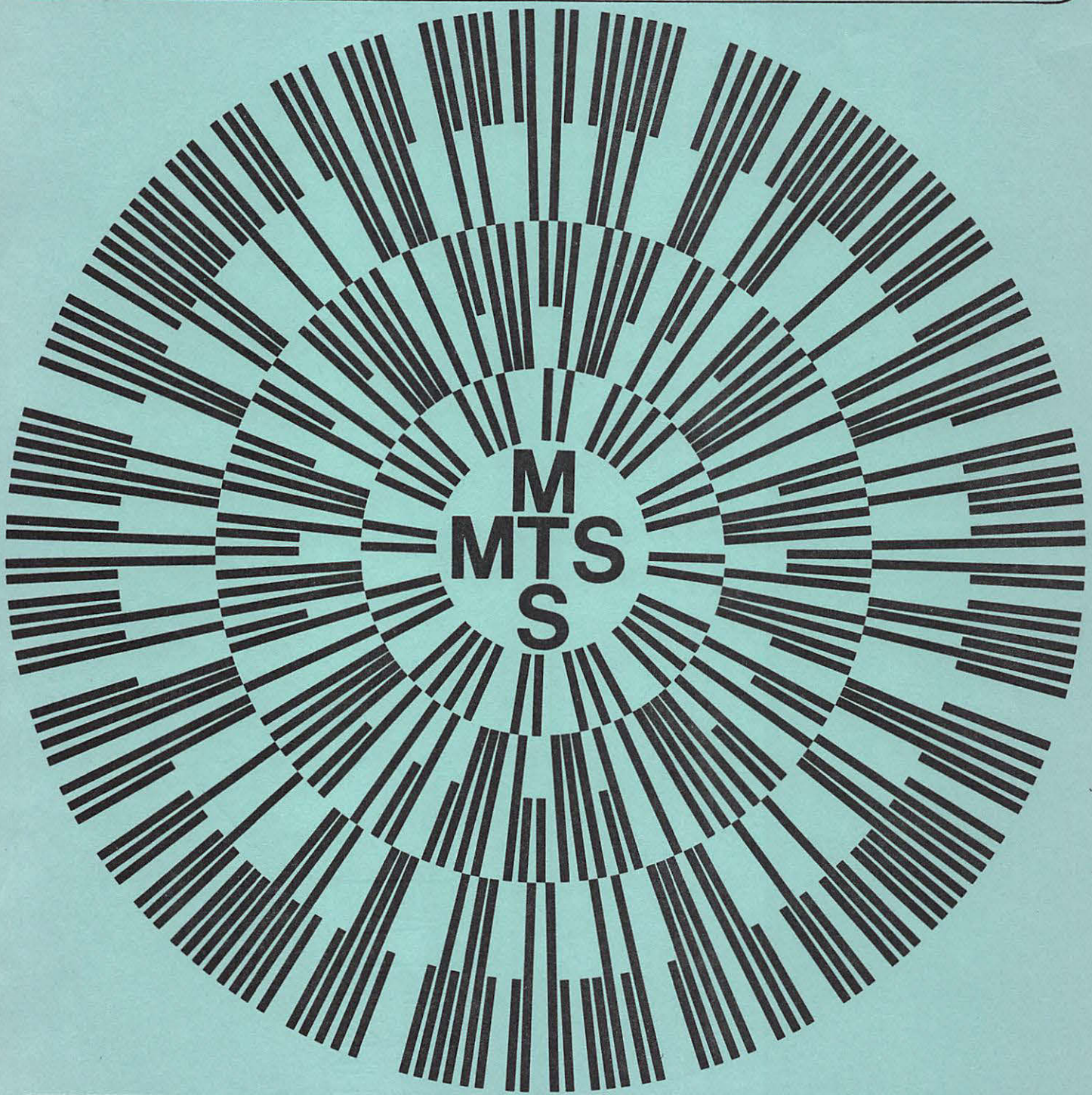
THE UNIVERSITY OF ALBERTA
COMPUTING CENTER
PUBLICATION



MAG TAPE USERS' GUIDE

ACKNOWLEDGEMENTS

THIS MANUAL WAS LARGELY COMPILED FROM MATERIAL PREPARED BY THE STAFF OF THE UNIVERSITY OF MICHIGAN COMPUTING CENTER. THEIR DOCUMENTATION WAS INVALUABLE AND WE ARE INDEBTED TO THEM FOR ALLOWING US TO USE IT. IN PARTICULAR, THE FOLLOWING WERE MOST USEFUL:

MTS USERS' MANUAL, SECOND EDITION, VOLUMES I AND II

MTS USERS' MANUAL, THIRD EDITION, VOLUME 2

INTRODUCTION TO MTS AND THE COMPUTING CENTER (FLANIGAN)

COMPUTING CENTER NEWS ITEMS

COMPUTING CENTER MEMOS

THE COMPUTING CENTER WISHES TO PERSONALLY ACKNOWLEDGE THE ASSISTANCE OF MIKE ALEXANDER AND DON BOETTNER WHO HELPED US TO ESTABLISH MTS AT THE UNIVERSITY OF ALBERTA.

ACKNOWLEDGEMENT SHOULD ALSO BE MADE TO THE COMPUTING CENTRE, UNIVERSITY OF BRITISH COLUMBIA, FOR INFORMATION OBTAINED FROM SOME OF THEIR DOCUMENTATION AND TO I.B.M., WHOSE MANUALS PROVIDED CERTAIN SECTIONS FOR OUR MANUALS.

DISCLAIMER

This MTS manual is a combination of earlier manuals, update notices, memos and limited experience with the system itself. Because of this, certain discrepancies are bound to occur and the Computing Center would appreciate being notified of all differences between what this manual says and what the system actually does.

This publication is intended to represent the current state-of-the-system. However, it should not be construed as an obligation to maintain the system as so stated. The MTS system, like most good systems, is continually being improved. As a result, additions, extentions, changes and deletions will occur. Notice of such changes will be made and provision for a manual updating service has been planned.

Errors, comments and suggestions should be sent to:

Information Coordinator
Computing Center
University of Alberta

MAG TAPE USERS' GUIDE

TABLE OF CONTENTS

1. INTRODUCTION

This document gives instructions for mounting and creating or accessing data stored on magnetic tapes.

The Computing Center has currently installed two seven-track tape drives and six nine-track tape drives. For more detailed specifications of either the tape drives' operating characteristics or magnetic tape format, see the IBM 2400 Series Magnetic Tape Units Component Description (IBM form number A22-6866).

2. BASIC CONCEPTS

In order to resolve any ambiguities which might arise, the following definitions are offered.

7 and 9 Track Tapes

A 7-track tape frame contains 6 bits of data and one parity bit; a 9-track tape frame contains 8 bits of data and one parity bit. This property of the tape is a direct function of the tape unit upon which the tape was written; that is, all 7-track tapes must be read and/or written on 7-track tape units, all 9-track tapes must be read and/or written on 9-track tape units.

Odd and Even Parity

The above mentioned parity bit may be manipulated so that there are either an odd (odd parity) or an even (even parity) number of bits set one in each tape frame. If binary records are to be written odd parity must be used to prevent loss of data. Common usage conventions also require that 7-track BCD tapes be written with even parity.

Densities

7-track tapes are written at densities of 200, 556, or 800 BPI. 9-track tapes are written at 800 BPI.

Translator and Data Convertor

Since 7-track tape characters contain only 6 bits of data and 360 data bytes contain 8 bits of data, there are conversion problems. To solve these problems, the translator and data convertor features are provided. By playing the appropriate games, it is possible to write a 7-track tape with either translator on, data convertor on, or both features off.

If translator feature is on, an automatic conversion is invoked which translates most 8 bit EBDIC characters into their corresponding 6 bit BCD counterparts.

If data convertor feature is on, 4 tape characters will be produced for each 3 data bytes. All 8 bits in each byte are thus transmitted. Data convertor on forces odd parity.

If neither translator or data convertor is specified the low order 6 bits of each data byte are transmitted to/from the tape. To insure accurate data transmission odd parity should be specified when both translator and data convertor are off.

9-track tape users can ignore this. All 9-track tapes are written in odd parity with both data convertor and translator off.

Mode

Each reel of tape has a mode associated with it. The normal procedure is to specify the mode of the tape at the time it is mounted; however, because it is possible to have mixed mode 7-track tapes, a schema is provided to dynamically change modes at any time. This will be discussed later.

The significance of the mode varies slightly between seven and nine-track tapes. For a seven-track tape the mode includes the density, parity and translator-data convertor attributes desired. The parameter used to set the mode is a three character sequence and is constructed in one of the following two ways:

(1) Type 1 mode commands are of the form "dCV"

(2) Type 2 mode commands are of the form "dpt" where
"d" is the code for density:
2 for 200 BPI
5 for 556 BPI
8 for 800 BPI
and "p" is the code for parity:
O for odd parity
E for even parity
and "t" specifies whether or not the translator feature is on:
N for translator on
F for translator off
and "CV" specifies that Data Convertor is on. Note that if data convertor is on there is no choice as to parity, it is always odd.

The default mode for seven-track tapes is 5EN, that is, 556 BPI even parity with translator on.

For a nine-track tape the mode specifies only the density, 800 BPI. The mode is specified as the sequence "800". All tape maintain odd parity, translator and data convertor features are not available. The default mode is 800.

File Protect Ring

Since writing on a tape destroys any information which may have been stored upon it, a file protection device is provided to prevent accidental erasure of tapes. This device, hereafter called file protect ring, or just ring for short, must be inserted in a groove in the back of the tape reel to allow writing of the tape. The user must specify at the time the tape is mounted whether the ring is to be in (allowing writing) or out (prohibiting writing).

Record Size

Every normal read or write operation transmits a block of data to/from a tape unit. This block of data is called a record. Usually a record corresponds to one line in a file; but is also possible to write very large records. One of the attributes associated with a tape is an upper bound to the size of a record.

End of Tape Area

Approximately 20 feet from the end of each reel of tape there is a silver strip known as the end of tape marker. There is enough tape after this marker to allow the writing of a short (5 record or less) trailer label. No more than 5 records may be written in this area.

Pseudo-Devices Names

Because there will usually be more than one active user at any given time it is not always possible to predict with any great probability of success what particular tape unit will be free. Therefore all references to tapes should be made through pseudo-devices names (PDname). MTS will then determine which unit the PDname refers to and the appropriate read/write function will be performed. A PDname may be used exactly as if it were a "FDname".

A PDname consists of an asterisk followed by a string of from one to fourteen alphanumeric characters and terminated by another asterisk. For example, *NAME* and *TAPE* are valid device names. Note that MTS preempts certain PDnames. These names may not be used as PDnames for magnetic tape units:

```
*MSOURCE*
*MSINK*
*SOURCE*
*SINK*
*AFD*
*DUMMY*
*PUNCH*
```

3. LABEL PROCESSING

The current version of MTS does not support label processing. Specifications for label processing support have been announced by the University of Michigan and the support is being implemented. Labels to be supported will be OS compatible.

4. BLOCKING/DEBLOCKING

The current version of the MTS tape support routine does not support blocking/deblocking of magnetic tape records i.e. a logical record is identical to a physical record

The University of Michigan has published specifications for this facility and is currently implementing it. The facility will be OS compatible.

Blocking/deblocking of tape records can be achieved using either *UPDATE or *DEB (see Utilities section).

5. MOUNTING TAPES

To request, from either a terminal or a batch job, that the operator mount a tape, the user runs the public file *MOUNT, supplying parameters which provide the operator with information so that the desired tape can be mounted properly and provide the system with information so that the physical device can be linked to the user's pseudo-device. A detailed description of the use of *MOUNT follows.

*MOUNT

Contents: The object module of the volume mounting program.

Purpose: To allow users to mount private volumes and attach certain attributes to those volumes.

Usage: The program is invoked by a $RUN command specifying *MOUNT as the file where the object cards are to be found. This program may also be called as a subroutine (see the description of MTS Mount subroutine in the LIBRARIES manual.

Prototype:
```
$RUN *MOUNT PAR=rackname [ON] devicetype
        [PNAME=]*pseudodevicename* [keyword  LHS=RHS]
        'printed label' [comments to operator] [;...]
```

Logical I/O Units Referenced:
GUSER - if "PAR=" field is omitted on $RUN
SERCOM - error comments and mount confirmation

Examples:
```
$RUN *MOUNT PAR=M000 ON 9TP, PNAME=*TEST*,QUIT=YES, SIZE=512,
RING=IN, 'TEST TAPE #1'; POOL 9TP *9*

$RUN *MOUNT
T00123 7TP *IN* RING=OUT 'INPUT TAPE'
T00747 9TP *OUT* RING=IN 'SIZE=8000 'OUTPUT TAPE'
$ENDFILE
```

Positional Parameters:

Note: Whenever a space is shown separating parameters in the prototype, multiple spaces and/or commas are acceptable.

rackname is the name of the rack at the Computing Center in which the volume is stored, If this is given as 'POOL', a system scratch tape will be mounted At the present time the rack number corresponds to the tape label.

devicetype is the type of device upon which the volume is to be mounted. Currently 7TP, 9TP, are supported, standing for 7 track magnetic tape unit, 9 track magnetic tape unit,

pseudodevicename is the pseudodevice name by which the user will refer to the mounted device.

'printed label' is the identifying name associated with the volume. This name should be written upon a paper label and attached to the volume when it is first submitted to the Computing Center. This parameter is meaningless for POOL magnetic tape mounting and may be omitted.

comments are any special directions to the operator.

Keyword Parameters:

See the summary below for a listing of the available keyword parameters. Descriptions of the keyword parameters for magnetic tapes are given below.

Description:

If the optional PAR= field is present on the $RUN command, several mount requests may be given, separated by semicolons (;).

If the PAR= field is omitted, input will be read from GUSER. If a mount request will not fit on one GUSER input line, the current MTS command line continuation character (normally a minus sign) may be entered as the last character of an input line to indicate that the next line is a continuation of the current line. As many requests as desired (each on a separate line(s), i.e. semicolons should not be used) may be entered. An end-of-file signifies the end of the requests. If a particular request is mounted without error, the name of the physical device used will be printed. If not, a comment from the operator will be printed indicating why the mount request cannot be fulfilled.

It is strongly recommended that users who need to mount more than one device at a time, not use a separate $RUN *MOUNT for each request. Not only is this inefficient since *MOUNT must be loaded several times, but in addition there is then no way for *MOUNT to inform the MTS operator for the total device requirements of the job. Therefore, any user who needs to mount more than one device should either specify all of the requests separated by semicolons in one PAR= field or enter the requests as data cards as shown in the last of the examples above.

Volumes mounted in this way will automatically be removed when a second volume is mounted with the same pseudodevice name or when the user signs off.

The following rules describe the interaction between *MOUNT and *DISMOUNT:

1. If *MOUNT is run (either explicitly or by calling the subroutine MOUNT during execution) with a pseudodevice name specified which is already active, then the device type must be the same as the current type and all outstanding references to the pseudodevice name will be changed to refer to the new volume.

2. If *MOUNT is run with a pseudodevice name which has been previously used but for which *DISMOUNT has been run, then the new device type may be anything legal and any outstanding references to the old pseudodevice name are unchanged.

In general a use of *DISMOUNT signifies the end of a particular use of a pseudodevice name and a use of *MOUNT for an active pseudodevice name signifies a new volume for a previous use. The following example may clarify this. Suppose the following commands were issued during an MTS session:

```
$RUN *MOUNT PAR=G000 ON 7TP PNAME=*TP* 'MY TAPE'
$RUN -OBJ SCARDS=*TP*
```

If during execution of the program in -OBJ, the subroutine MOUNT is called and the pseudodevice name *TP* is specified then the device type must be 7TP and a new reel will be mounted on the same tape drive, which SCARDS will still refer to. However if during execution of the program in -OBJ, DISMOUNT is called for *TP* and then MOUNT is called for it, the device type may be any legal type, SCARDS will still refer to the old tape, and any new use of the name *TP* (for example as a parameter to GETFD) will refer to the new volume. In this case the original tape reel will not actually be removed until execution is terminated.

KEYWORD PARAMETERS

Left Side Legal Values	Right Side Legal Values	Default If Omitted	Function
For magnetic tapes:			
MODE	any 7TP mode or 9TP mode	5EN for 7TP 800 for 9TP	Control tape mode
RING	IN,OUT	OUT, except for 'POOL' tapes which are always IN	Control file protect ring
QUIT	YES,NO	YES	Terminate batch job if the mount is not successful
SIZE BLKSIZE	18-32767	255	Set maximum physical record size
RETRY	1-10	10	Read retry count

6. READING/WRITING TAPES

To use magnetic tapes in the command mode, use the PDname as the FDname. For example

```
$COPY  MYFILE TO  *TAPE*
```

To use magnetic tapes in the program mode, the logical I/O device used in the program must be linked to the pseudo-device name by a keyword parameter on the $RUN command. For example if a FORTRAN program contains the statement

```
WRITE (1,100) DATA
```

the logical device 1 might be linked to the pseudo-device *TAPE* by

```
$RUN PROG  1=*TAPE*
```

Return Codes

The return codes resulting from reading from or writing to tapes extend beyond those returned from other devices. They are:

0	Normal return
4	(from read operations) End-of-file mark detected. (from write operations) End-of-file strip sensed.
8	Tape load point has been sensed on backspace command.
12	User attempted to write more than five records in end of tape area.
16	Permanent read/write error (on read, tape will be positioned past bad record).
20	Attempt to write on a file-protected tape.
24	Equipment malfunction.

Unless the device error exit has been set by the user, all return codes greater than 8 will be intercepted by MTS and the current RUN will be terminated. Control will be returned to the user in command mode. If the device error exit has been set, the user may recover from the error in any way he chooses and still remain in execution. (See a description of the SETIOERR subroutine.)

Several sequences of operation are illegal. The following table defines which operations are illegal when following certain other operations.

		Current Operation					
		read	FSR	FSF	BSR	BSF	REW
Preceeding Operation	write	X	X	X	T	T	T
	WTM	X	X	X	T	T	T
	REW				N	N	N

where X - tape is terminated with 5 tape marks and rewound and unloaded.
T - tape is terminated with 5 tape marks and backspaced over them before the current operation is performed.
N - no operation is performed.

For example, reading a record immediately after writing a record is illegal and causes five tape marks to be written on the tape before the tape is unloaded.

7. TAPE POSITIONING AND CONTROL

Users may want to initiate such control functions as rewinding and backspacing tapes. This can be done from either the command or program modes.

In the program mode, system subroutines, that perform the control functions, can be called explicitly using, for example, the CALL statement in FORTRAN or the CALL macro in Assembler, or implicitly using, for example, REWIND, and ENDFILE statements in FORTRAN or REWIND, ENDFILE, and CHMODE macros in Assembler. The subroutines are described in the LIBRARY manual. More detail on FORTRAN I/O statements is found in the FORTRAN manual and detail on the Assembler macros is found in the ASSEMBLER manual.

In the command mode, attaching the logical carriage control attribute @CC to the FDname or PDname causes data lines being written to a tape to be examined for logical commands in the first, usually, three bytes of data. If a logical command is recognized the system performs the tape control function specified and the line is <u>not</u> treated as data. A complete table of the logical commands is given below. For example the sequence of lines

```
$COPY *SOURCE* TO *TAPE*@CC
FSF
$ENDFILE
```

causes the tape *TAPE* to be forward spaced one file.

Tape Control Functions

Logical Command	Action Taken	Non-Zero Return Code Meaning
WTM	End of file record written.	None
FSF n 1	Tape spaced forward past next n end of file records.	4=in end of tape area
FSR n 1	Tape spaced forward n records	4=tape mark sensed
BSF n 1	Tape spaced backward past next n end of file records.	8=loadpoint sensed
BSR n 1	Tape spaced backward n records	4=tape mark sensed 8=loadpoint sensed
REW	Tape rewound to loadpoint	None
RETRY n 10	Control number of retry operations performed when tape read errors are encountered $1 \leq n \leq 15$	None
SRL n	Resets maximum record length to n.	4=illegal length
SNS	Return detailed information about the status of the tape routines.	None
dCV or dpt	Changes mode during execution, perhaps to read a record with bad parity. (See definition of modes in Basic Concepts Section)	None

In the above table n is an integer of one to five digits.

Return Codes:

0 Normal return
4 see above table
8 see above table
16 first 3 bytes are not recognized as a legal control function.

The Sense Command

The sense command is to be used primarily by system components in attempting to correct for tape errors. It returns much detailed and cryptic information about the exact status of the tape unit and tape routines at the time of the last operation. That information is placed in a 72 byte buffer specified by the region parameter in the parameter list. As with all control commands the control sequence "SNS" must be in the first three bytes of the region. Upon return from the tape routines the buffer will contain the following information.

Bytes	0 - 2	"SNS"
	3	One byte current machine modeset
	4 - 7	Four byte device name
	8 - 11	Four byte device type
	12 - 13	Two byte binary maximum record length
	14	One byte tape status
	15 - 20	Six bytes of tape sense data
	21 - 71	Reserved

The sense command does not alter the length parameter as specified in the parameter list.

Note: The modeset, unit status and sense data are described fully in the IBM 2400 Tape Drive Manual (Form A22-6866-2) and the values will reflect the last operation applied to the tape. If no sense operation was issued on the last read or write (i.e., the operation was successful) the unit status byte will be set to zero and the sense data will not be meaningful.

Example of use of control functions

The following MTS commands mount a magnetic tape, then execute the same sequence three times, once each from FORTRAN, command mode, and the assembler. The sequence spaces the tape forward two files, writes a tape mark and rewinds the tape. It is not intended to be a useful program, rather merely to illustrate how the control functions are used.

```
$RUN *MOUNT PAR=0001 ON 9TP *TAPE* RING=IN 'TAPE'
$RUN *FORTG
      CALL SKIP (2,0,6)
      ENDFILE 6
      REWIND 6
      END
$RUN -LOAD# 6=*TAPE*
$COP *SOURCE* *TAPE*@CC
FSF 2
WTM
REW
$ENDFILE
$RUN *ASMG SPUNCH=-X
      ENTER  12
      WRITE  6,'FSF',@CC   <- equivalent - each spaces
      CALL   SKIP(1,0,6)   <- forward one file
      ENDFILE 6
      REWIND 6
      EXIT   0
      END
$RUN -X 6=*TAPE*
```

8. REMOVING TAPES

There are three ways to have a tape removed from a tape drive:

1) Mount another tape on the same pseudo-device - that is, invoke *MOUNT using a PDname which is being used for another tape. The first tape will be removed and the second mounted in its place. The mode and record size of the first tape will be used for the second tape unless explicitly reset.
2) Invoke *DISMOUNT (described below).
3) Sign off MTS.

In either case, if the last operation performed on the tape was a write, five end-of-file marks will be written on the tape to terminate it before it is unloaded.

*DISMOUNT

Contents: The object module of the volume dismounting program.

Purpose: To allow users to dismount private volumes.

Usage: The program is invoked by a $RUN command specifying *DISMOUNT as the file where the object cards are to be found.

Prototype: $RUN *DISMOUNT PAR=*pseudodevicename*

[*pseudodevicename* is the pseudo-device name specified when the volume was mounted.]

Examples: $RUN *DISMOUNT PAR=*TAPE*

Description: The volume will be removed and placed in its appropriate spot in the storage rack. See *MOUNT description for further details. If there are any outstanding uses of the pseudo-device name (such as SCARDS) the volume will not be actually dismounted until all of them are released. For a description of the interaction between *MOUNT and *DISMOUNT, see the *MOUNT writeup in this Volume.

9. UTILITIES

The following list summarizes the public files that are available.

*DEB	deblocks/blocks
*IEHMOVE	to "load" unloaded OS data sets on tape
*LABELSNIFF	prints OS compatible tape labels.
*LIST	lists fixed, blocked records.
*TAPECOPY	copies tapes using overlapped I/O.
*TAPEDUMP	dumps a magnetic tape in both character and binary formats.
*UPDATE	copies tapes on files containing card images making insertions and deletions, as well as blocking and unblocking.
*VARRLIST	lists variable length records.

*DEB

Contents: *DEB contains the object module of the deblocking/blocking program.

Usage: *DEB is invoked through the $RUN command.

Logical I/O Units Referenced:
SCARDS - The control statement is read via SCARDS if it is not passed as a parameter.
SERCOM - Diagnostics are issued via SERCOM.

Description: The deblocking/blocking program

(1) provides deblocking and blocking facilities, processing data sets of types U, F, V, FB, and VB;

(2) allows user subroutines to gain control at several points, permitting the generation, modification, deletion, and comparison of records;

(3) is serially reusable and may be called as a subroutine.

1. The control statement

This describes the structure of the I/O data sets. The control statement is read via SCARDS if it is not passed as a parameter. It may extend over any number of records and may be encoded anywhere between columns one and eighty. The statement may be broken for continuation between parameters and is terminated by the first blank following the OUTPUT keyword.

A prototype of the control statement follows:

```
INPUT=name;type;record length;block length...
OUTPUT=name;type;record length;block length...
```

A keyword is that part of the statement which precedes the equal sign and a group consists of the four parameters following the equal sign.

The ellipses denote additional groups that may be appended to the first. Subsequent groups should be separated by semicolons and the last group for each keyword must be followed by a blank.

The INPUT and OUTPUT keyword parameters describe the data sets to be read and written. Each group consists of four parameters, name, type, record length, and block length and describes one data set. Data sets are read in the order in which their descriptions appear. Data sets are written in the order in which their descriptions appear, each being "filled" before the next is used. The characteristics, i.e., type, record length, and block length, of all output data sets must be alike. The following substitutions are necessary:

1. Replace "name" by the file or pseudodevice name of the data set. Modifiers and line number ranges may be appended to the name.

2. Replace "type" by one of the record structure codes appearing in the table below. OS/360 - Supervisor and Data Management Services, IBM publication C28-6646, contains descriptions of the structures.

CODE	RECORD STRUCTURE
U	undefined length
F	fixed length
V	variable length
FB	fixed length blocked
VB	variable length blocked

3. Replace "record length" by the byte length of the longest record occurring in the data set.

4. Replace "block length" by the byte length of the longest block occurring in the data set.

2. Example.

A line file named SIN contains unblocked card images. Each image bears the name of a city, columns 1 - 60, and the city's population, columns 70 - 80. Two tapes are mounted with the pseudo-device names *DATA* and *BYPOP*. *DATA*'s records contain additional information to be included in the

list. Its records have the format of SIN's but are blocked ten to the block. *BYPOP* will contain the list. Its records should be type VB, a maximum of 10,000 bytes per block.

```
$RUN *DEB
INPUT=SIN;F;80;80;*DATA*;FB;80;800
OUTPUT=*BYPOP*;VB;84;10000
```

3. Optional exits.

User subroutines may gain control from the deblocking/blocking program at any of three points or exits. An exit is enabled, i.e., a subroutine is called, if the name of the exit is encountered as an external symbol during the loading of the deblocking/blocking program.

All subroutine calls are made via a standard OS linkage, type (I) S. Upon entry, general register one will point to a parameter list: the address of the first byte of the record or region; the address of the halfword length of that record or region. (See below for details).

The subroutines should return control to the deblocking/blocking program via the standard OS (I) S return. The exits are as follows:

DEBE1 If this exit is enabled, it is taken to acquire a record for processing. The call to the user's subroutine replaces the call that would normally be made to READ. A block may be moved into the region addressed by the first word of the parameter list. The length of this block should be placed in the halfword addressed by the second word of the parameter list.

If this exit is enabled, a dummy data set must be defined following the INPUT keyword. The data set's parameters should describe the block to be passed to the deblocking program by the subroutine. The deblocking facilities may be used to decompose blocks passed by the subroutine.

An end-of-file can be passed to the deblocking program by a non-zero return code in general register 15.

DEBE2 If this exit is enabled, it is taken immediately after a record has been acquired for processing -- following deblocking, if any. The user's subroutine may delete or modify the record before it

enters the program. The first word of the parameter list will point to the record; the second, to its halfword length plus four. If a record's length is altered by the subroutine, the new length, plus four, must replace that passed to the subroutine. If a record's length is made zero, the record is discarded. A non-zero return code will force the end-of-file condition on the current input data set.

DEBE3 If this exit is enabled, it is taken to dispose of a block. The call to the user's subroutine replaces the call that would normally be made to WRITE. The first parameter points to the block and the second to its halfword length.

If this exit is enabled, a dummy data set must be defined following the OUTPUT keyword. The data set's parameters should describe the block or record to be passed to the subroutine. The blocking facilities may be used. A non-zero return code causes the next output data set to be opened and filled.

*IEHMOVE

Contents: The object module of the IEHMOVE program.

Purpose: To "load" unloaded data sets from an input tape.

Usage: The program should be referenced by a $RUN command with *IEHMOVE as the object file, 0 the input tape, 1 the output, and SPRINT the listing.

Logical I/O Units Referenced:

SPRINT - the listing.
SERCOM - comments and diagnostics.
0 - IEHMOVE unloaded data set.
1 - the output.

Parameters: LIST prints the listing of the original data set via SPRINT. NOLIST is the exact opposite. LIST is the default unless lines to be printed are more than 120 characters.

Examples:

```
$RUN *IEHMOVE 0=*TAPE* 1=MACLIB(1000) PAR=LIST

$RUN *IEHMOVE 0=*IN* 1=*PUNCH* SPRINT=PTR6

$RUN *IEHMOVE 0=*IN* 1=*OUT* PAR=NOLIST
```

Description: The program prints the data set name and its record format. All record formats including V-format are supported but key files will not be permitted. Records are unblocked for MTS line files and unit-record devices such as a printer. If the input was originally a partitioned data set, a comment is printed for each member with an associated line number or count depending on whether the output is an MTS line file. If the output happens to be a magnetic tape, each member will be followed by a tape mark (end-of-file).

*LABELSNIFF

Purpose: To print OS compatible tape labels in intelligible format.

Prototype: $RUN *LABELSNIFF PAR=TAPE=*pseudodevicename*

Usage: This is a PL/1 program. Hence, the parameter specification TAPE=*pseudodevicename* for the tape to be sniffed at must be made. Do not use on brand-new blank tapes.

Example: $RUN *LABELSNIFF PAR=TAPE=*TAPE*

Logical I/O Units Referenced:
SPRINT - tape label and diagnostic output.

Description: The program rewinds the specified tape and determines whether the tape is properly labeled (OS compatible). It then scans the tape, printing all of the volume, header, and trailer labels on the tape. When the end of the tape is detected, the tape is rewound and the program terminates.

*LIST

Contents: The object module of a program which lists blocked records (usually from a magnetic tape).

Purpose: To enable the user to list 121-column assembler output lines from a magnetic tape. For 80-column card images, see the *UPDATE public file description. For general record unblocking, see the *DEB description.

Usage: The program in invoked by a $RUN command specifying *LIST as the location of the object module.

Parameters: Two parameters separated by blanks must be specified:

```
PAR=*pseudodevicename*  blockingfactor
```

Logical I/O Units Referenced:

SPRINT - the listing output.

Example: To list assembler output blocked 40 logical records per physical record on the tape *T*

```
$RUN  *LIST  SPRINT=*SINK*@MCC  PAR=*T* 40
```

*TAPECOPY

Contents: The object module of the MTS tape copying program.

Purpose: To copy tapes using overlapped I/O.

Usage: 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 his own program(s). 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 adcon. This adcon is the location of a half-word character count which is immediately followed by the EBCDIC characters of the parameter. If no parameter is to be passed, then the half-word count should be zero.

Logical I/O Units Referenced:

SPRINT - the listing of the number of records in each file.
SERCOM - error comments.
0 - the input tape pseudo-device name.
1 - the output tape pseudo-device name.

Parameters:

FILES=n — The number of files to be copied is specified by n, where n is a one to six digit decimal integer.

RECORDS=m — The number of records 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 description 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 record in error will be skipped and copying will continue with the next record.

COPYERRORS — This parameter causes the same error comment printing as OMITERRORS. However, any records which are in error <u>are</u> copied.

Examples:

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

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

```
$RUN *TAPECOPY 0=*X* 1=*Y* PAR=NOREW,FILES=163

$RUN *TAPECOPY 0=*MASTER 1=*COPY* PAR=OMITERRORS
```

Description: The tape copying program copies records and filemarks from logical device 0 to logical device 1. If no FILES=n and RECORDS=m parameters are given, then the copying continues until two successive file marks are read (this is taken as an indication of the logical end of the input tape). Five file marks are written on the output tape. If FILES=n and-or RECORDS=m parameters are given, then n files and the first m records of file n+1 will be copied before the five file mark termination is written on the output tape. If n and m are zero, then all files are copied as if no FILES and RECORDS parameters had been given.

If the NOREW parameter is <u>not</u> given, then both tapes are rewound before and after copying. If it is given, then <u>no</u> rewinding takes place either before or after copying and the output tape is left positioned after the first of the five file marks used for termination.

If neither of the parameters OMITERRORS or COPYERRORS was given and an unrecoverable tape 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 user must mount the tapes using *MOUNT, specifying that the file protect ring is in for the output tape, that a SIZE parameter equal to or greater than the maximum record length on the input tape be given for <u>both</u> tapes, and that the mode be correct for both tapes.

This program may be used to copy a 7 or 9 track tape of any density and mode to a 7 or 9 track output tape of any mode and density.

Warning: This program will not function properly if line number ranges are applied to the <u>input</u> tape(s). However, explicit concatenation of tape names is permitted on both input and output.

*TAPEDUMP

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

Prototype: $RUN *TAPEDUMP 0=*pseudodevicename* [PAR=parameters]

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

Logical I/O Units Referenced:
SPRINT - dump output.
0 - magnetic tape or file to be dumped.

Parameters:
FILES=n
RECORDS=m
EBCDIC
BCD
NODUMP

Description: *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 parameter 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 tape will be dumped until two consequtive filemarks are encountered.

"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 in the dump. All non-printing graphics in the EBCDIC (or BCD) portion of the dump will print as the character period(.).

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 and mode will continue to print, but the data dump will not.

If the FDname specified as logical device zero is a 7-track tape, the default is BCD. If the FDname refers to a file or a 9-track tape, the default is EBCDIC.

The BCD character set uses the IBM scientific (ALTERNATE) BCD graphics. Note that this character set is neither the IBM STANDARD (BUSINESS) BCD, nor the University of Michigan (AUGMENTED) BCD.

*UPDATE

Purpose: This program will copy tapes (or files) containing card images, making insertions and deletions, as well as blocking and unblocking.

Logical I/O Units Referenced:

SPRINT - printed output.
SPUNCH - output from %PUNCH.
SERCOM - error messages.

Commands and insertions are expected to come from the source stream (*SOURCE*). If another source of commands is wished, its FDname should be specified following the "PAR=" on the $RUN command. Commands and insertions must be less than or equal to 80 bytes in length.

Examples:

```
$RUN  *UPDATE
$RUN  *UPDATE PAR=PIL.UPDATE
```

Usage: The update input tape must consist of 80 column card images which may be blocked to any factor desired. The blocking factor, if greater than 1, must be stated on the %INPUT command. The update output tape will consist of 80 column card images blocked as specified (except for the last record, and other records which may be truncated by a %CLOSE command). [Space for the specified input and output buffering is obtained dynamically when %INPUT and %OUTPUT command are encountered; released when %CLOSE is encountered.]

All commands take the following form: column 1 must contain a percent-sign ("%") which must be immediately followed by the command (only the first three letters need be given, and they must be upper-case. Most devices are in upper case mode unless commanded otherwise). Parameters for the command are separated from the command and from each other by one or more blanks (or commas, which are treated identically with blanks). Lines which are not recognized as commands are treated as insertion lines and are copied immediately to the update output tape.

There are four different types of parameters used in the commands: numeric, filemark, character, and keyword. Numeric parameters are used for tape operation counts, deletion counts, and so forth. They consist of one to twenty digits which represent an unsigned decimal integer. A filemark parameter is used to refer to a filemark and consists of the characters FILEMARK or FILEMK. Character parameters are used for FDnames, pseudo-device names, I.D.'s, etc. There are two forms of character parameters. The first form consists of

one to eighty characters with the restrictions that the first character cannot be a digit or an apostrophe (') and neither blanks nor commas (,) can be a part of the character parameter. The second form of a character parameter consists of from one to eighty characters enclosed in apostrophes, with an apostrophe within the character parameter represented by two adjacent apostrophes. The second form does not restrict the use of a digit or apostrophe as the first character nor the use of blanks and commas within the parameter. Note that the outer apostrophes act only as delimiters and are not considered a part of the character parameter. Keyword parameters are simply keywords which are specified for a specific command, such as ON or OFF.

Examples:

```
     Numeric Parameters        1    2     15    123
     Filemark Parameters       FILEMARK   FILEMK
     Character Parameters      PIL6215     *T*  SEQ.0001
                               '12340001' 'PIL 00001'
                               '''TS"0001' '*T*(1,100)'
     Keyword Parameters        ON   OFF *
```

Tape positioning and reading across a file mark while the tape is still open is considered an error. Therefore %CLOSE commands should precede positioning and the update should finish copying of a file with %BEFORE FILEMARK, not %AFTER FILEMARK.

Tape Attachment and Manipulation Commands

%INPUT INTAPE [N]

INTAPE is the pseudo-device name (or FDname) of the file or device to be established as the update input tape. N is an integer specifying the blocking factor of that tape. If omitted, a blocking factor of 1 card/record (i.e., unblocked) is assumed. This command causes the tape to be opened.

Example: %INPUT *IN* 50

%OUTPUT OUTTAPE [N]

OUTTAPE is the pseudo-device name (or FDname) of the file or device to be established as the update output tape. N is an integer specifying the blocking factor desired on that tape. If omitted a value of 1 (unblocked) is assumed.

```
Example: %OUTPUT   *OUT*   20
         %OUTPUT   FILE1   1
```

%REWIND T

Tape T is rewound. T must not currently be open as

input or output. The rewind operation is performed by the calling the subroutine REWIND#. All files or devices which this subroutine can rewind may be specified. See the writeup on REWIND# in Volume 3 of the MTS Manual.

Example: %REWIND *OUT*

%RUN T
%UNLOAD T
Tape T is rewound and unloaded. T must not currently be open as input or output.

Example: %RUN *OUT*

%FSF T [N]
Tape T is spaced forward N files. If N is omitted a value of 1 is assumed. T must not currently be open as input or output.

Example: %FSF *IN* 3

%BSF T [N]
Tape T is spaced backwards N files. If N is omitted a value of 1 is assumed. T must not currently be open as input or output.

Example: %BSF *IN*

%WTM T [N]
%WEF T [N]
%EOF T [N]
Tape T has N tape marks (end-of-file marks) written on it. If N is missing a value of 1 is assumed. T must not currently be open as input or output.

Example: %WTM *OUT*

%FSR T [N]
Tape T is spaced forwards N records. If N is omitted, a value of 1 is assumed. T must not currently be open as input or output.

%BSR T [N]
Tape T is spaced backwards N records. If N is omitted, a value of 1 is assumed. T must not currently be open as input or output.

%CLOSE [T]
Tape T is closed. If the update output tape, the last buffer (possibly truncated) is written out. If T is omitted, the update output tape is assumed.

Update Feature Control Commands

%NEWID H

Cards written onto the update output tape following this command will have new ID's (columns 73-80). The first card written will have the ID specified in this command. Succeeding cards will have ID incremented in steps of one. The ID given in this command should consist of 8 characters. Only the numeric portion of the ID is incremented.

Example: %NEWID PIL00001

%OLDID

Suspends the re-IDing of the cards as described under %NEWID.

%LIST ON

Starts listing of deleted and inserted cards. (Initially on)

OFF

Stops listing of deleted and inserted cards [This output goes onto *SINK*]

%PUNCH ON

Starts putting all card images sent to the update output tape on SPUNCH (presumably for punching).

OFF

Turns off the "punching" described above. (Initially off)

Card Location, Copying and Deletion Commands

In the execution of these commands, the 360 collating sequence is used for comparisons.

%AFTER ID

Copies all cards having ID's less than or equal to ID from the update input tape to the update output tape

N

Copies the next N cards from the update input tape to the update output tape.

Examples:
```
%AFTER   PIL03789
%AFTER   '04780000'
%AFTER   2
```

%BEFORE ID

Copies all cards having ID's less than ID from the update input tape to the update output tape

FILEMARK

FILEMK
Copies the rest of the file. It leaves the tape positioned after the file mark.

Examples: %BEFORE PIL07892
%BEFORE FILEMARK

%DELETE ID
Copies all cards having ID's less than ID from update input to update output, then deletes the card (or cards) having ID of ID (if any).
N
Deletes the next N cards on the update input tape
ID1 ID2
Copies all cards having ID's less than ID1 from update input to update output, then deletes all cards having ID's ID1 through ID2 inclusive from the input.
ID1 N
Copies all cards having ID's less than ID1 from update input to update output, then deletes the next N cards on the input tape.
* ID2
Deletes all cards on the update input tape from the current position up through ID2.

Examples %DELETE PIL00016
%DELETE 2
%DELETE PIL00378,PIL00379
%DELETE PIL00378,2
%DELETE * PIL00736

%FIND ID
The update input tape is searched for a card with ID equal to ID. Order of the ID's is ignored. Card passed over are not copied to the update output tape.

Example: %FIND QQSV0395

%UNTIL ID
The update input tape is searched for a card with ID equal to ID. Order of the ID's is ignored. Cards passed over are copied to the update output tape.

Return Command

%END
This command or an end-of-file encountered in the command stream causes execution of the update program to terminate. All buffers are closed.

Sample Command Stream

```
%INPUT     *IN*  50
%OUTPUT    *OUT*   1
%DELETE    PIL00016
%AFTER     PIL00079
             GETSPACE  8193,T=3
             LR     15,1
             L      14,=F'8192'
%DELETE    PIL00830,2
%DELETE    PIL07044,PIL07049
             PUTLINE
             GETLINE
%AFTER     PIL09711
SAVR#        DS   6F
SAVREG#      DS  18F
PARREG#      DS   6F
%BEFORE    FILEMARK
%CLOSE
%CLOSE     *IN*
%WTM       *OUT*
%REW       *OUT*
%RUN       *IN*
%END
```

*VARRLIST

Contents: The object module of the V-format output lister.

Purpose: To list V-format records on a printer.

Usage: The program is invoked by an appropriate $RUN command specifying *VARRLIST as the source file.

Logical I/O Units Referenced:
SCARDS - source input with variable format
SPRINT - unblocked V-format records.

Example: $RUN *VARRLIST SCARDS=*TAPE* SPRINT=*SINK*

Description: The program serves the same purpose as *LIST except that it processes V-format records.