TAPTRN - A TAPE UTILITY PROGRAM

by

S. Silna

DEFENCE RESEARCH ESTABLISHMENT OTTAWA
TECHNICAL NOTE 69-38

This document has been approved for public release and sale; its distribution is unlimited.

April 1984
Ottawa
TAPTRN - A TAPE UTILITY PROGRAM

by

S. Slinn
SARSAT Project
Electronics Division
ABSTRACT

The computer program TAPTRN, a utility routine used to read magnetic tapes of unknown content and format, is described for use on an HP 1000 computer. It provides the user ready access to all the capabilities of the tape handling hardware of the HP 1000. TAPTRN is documented in terms of a brief description of the utility, its capabilities, a guide on how to use it, and the source and pseudocode for the routines written.

RÉSUMÉ

Cet ouvrage porte sur le programme d'ordinateur TAPTRN, sous-programme de service utilisé avec l'ordinateur HP 1000 pour lire des bandes magnétiques de présentation et de contenu inconnus. Ce programme permet à l'utilisateur d'avoir facilement accès à toutes les possibilités du dérouleur de bande de l'ordinateur HP 1000. L'auteur fait une brève description du sous-programme de service TAPTRN, explique quelles sont ses possibilités, fournit des instructions sur la façon de l'utiliser et indique la source et le pseudo-code des routines écrites.
**TABLE OF CONTENTS**

<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>INTRODUCTION</td>
<td>1</td>
</tr>
<tr>
<td>2.0</td>
<td>TAPTRN DESCRIPTION</td>
<td>1</td>
</tr>
<tr>
<td>2.1</td>
<td>PROGRAM OVERVIEW</td>
<td>1</td>
</tr>
<tr>
<td>2.2</td>
<td>CALLING SEQUENCE AND PARAMETER LIST</td>
<td>2</td>
</tr>
<tr>
<td>2.3</td>
<td>ERROR CODES</td>
<td>4</td>
</tr>
<tr>
<td>3.0</td>
<td>TSTAPE DESCRIPTION</td>
<td>5</td>
</tr>
<tr>
<td>3.1</td>
<td>PROGRAM OVERVIEW</td>
<td>5</td>
</tr>
<tr>
<td>3.2</td>
<td>Compiling and Running the Program</td>
<td>6</td>
</tr>
<tr>
<td>4.0</td>
<td>SUMMARY COMMENTS</td>
<td>7</td>
</tr>
<tr>
<td>APPENDIX A</td>
<td>TAPTRN SOURCE LISTING</td>
<td>9</td>
</tr>
<tr>
<td>APPENDIX B</td>
<td>TAPTRN PSEUDO CODE</td>
<td>21</td>
</tr>
<tr>
<td>APPENDIX C</td>
<td>TSTAPE SOURCE LISTING</td>
<td>27</td>
</tr>
<tr>
<td>APPENDIX D</td>
<td>TSTAPE PSEUDO CODE</td>
<td>47</td>
</tr>
<tr>
<td>APPENDIX E</td>
<td>TRANSFER FILES TO COMPILE AND LOAD THE PROGRAM</td>
<td>55</td>
</tr>
</tbody>
</table>
1.0 INTRODUCTION

TAPTRN is a utility program to aid in the processing of digital tapes.

This report describes two programs, TAPTRN and TSTAPE. TAPTRN is a FORTRAN callable subroutine designed to facilitate the processing of digital tapes. It contains a variety of functions which makes available all the capabilities of the mag tape hardware. This subroutine is written in FORTRAN 4X and FLECS, a FORTRAN preprocessor.

TSTAPE is a main program, written in FORTRAN 4X, that has been created to facilitate the usage of the TAPTRN program by providing full access to all the TAPTRN functions.

Both of the programs, TAPTRN and TSTAPE, are intended to be used on an HP1000 computer with a RTE IVB operating system.

In the following sections, TAPTRN will be documented under the headings: Program Overview, Calling Sequence and Parameter List, and Error Codes. The source listing for TAPTRN is found in Appendix A and the pseudo code describing the source listing in Appendix B. TSTAPE will be documented under the headings: Program Overview, and Compiling and Running the Program. The source listing for TSTAPE is found in Appendix C and the pseudo code describing the source listing in Appendix D.

2.0 TAPTRN DESCRIPTION

2.1 PROGRAM OVERVIEW

The facilities provided by TAPTRN are as follows:

1) Variable record length.

2) Ability to skip and backspace multiple records or files, thereby allowing the user to:
2

a) skip to subsequent files without having to read through intervening files;
b) position the tape at the logical end of volume;
c) perform a logical rewind to position the tape at the beginning of a file which is not the first file on the reel.

3. Provide the calling program with information on the tape position, i.e. BOT, EOF.

4. Return success or failure on all requested functions along with device status word.

5. Erase four inches of tape.

TAPTRN has ten different functions. With this capability, the user has all the necessary functions to perform any combination of operations on the tape drive. The only restrictions imposed are those of the hardware limitations of the controller and drive.

2.2 CALLING SEQUENCE AND PARAMETER LIST

TAPTRN is called by the FORTRAN statement:

CALL TAPTRN(IUNIT,KODE,KOUNT,IERR,NTRAN,ISTAT,IPOS,IBUFF)

None of these parameters are optional, however, some of the parameters are for input, and some are for output. The input parameters are:

IUNIT - the logical unit number assigned to the tape drive, it has to be zero (not related to the system logical unit numbers).

KODE - the function code.

KOUNT - the count, used in combination with KODE as shown in the table for TAPTRN function codes.

IBUFF - the user buffer name, an integer array to receive the data read from the tape, or it contains the data to be written to the tape. The maximum length allowed for this buffer is 9950 words.

The output parameters are:

IERR - the error code, where:
0 - successful completion of the requested operation
<0 - failure of requested operation.

IPOS - the tape position, where:

=1, BOT
=2, EOF
=3, EOT
=4, EOV

ISTAT - status return information (see table).

NTRAN - for reading/writing records, it contains the actual number of
words transferred (in reading, this may differ from KOUNT if
an EOF is encountered). For the skip/backspace functions,
NTRAN contains the number of files/records skipped.

TABLE 1

TAPTRN Function Codes

<table>
<thead>
<tr>
<th>KODE</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>-2</td>
<td>Unlock logical unit</td>
</tr>
<tr>
<td>-1</td>
<td>Obtain resource number and lock logical unit</td>
</tr>
<tr>
<td>0</td>
<td>Status request</td>
</tr>
<tr>
<td>1</td>
<td>Rewind tape</td>
</tr>
<tr>
<td>2</td>
<td>Write end-of-file</td>
</tr>
<tr>
<td>3</td>
<td>Skip KOUNT records forwards</td>
</tr>
<tr>
<td>4</td>
<td>Backspace KOUNT records</td>
</tr>
<tr>
<td>5</td>
<td>Skip KOUNT files forwards</td>
</tr>
<tr>
<td>6</td>
<td>Backspace KOUNT files</td>
</tr>
<tr>
<td>7</td>
<td>Erase four inches of tape</td>
</tr>
</tbody>
</table>
| 8    | Read a record and transfer up to KOUNT words
      (maximum 500 words) |
| 9    | Write a record KOUNT words long (maximum 500 words) |
### TABLE 2

Status Return Information (ISTAT)

<table>
<thead>
<tr>
<th>ISTAT</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Parity and/or timing error.</td>
</tr>
<tr>
<td>2</td>
<td>Reel does not have Write enable ring.</td>
</tr>
</tbody>
</table>
| 3     | I/O request requested:  
  | a) Tape motion required, but tape is at load point.  
  | b) Backward tape motion required, but tape is at load point.  
  | c) Write request was given, but reel does not have write enable ring. |
| 4     | Timing error on last read/write operation. |
| 5     | End-of-tape marker sensed. |
| 6     | Start-of-tape marker sensed. |
| 7     | End-of-file record encountered when reading, forward spacing, or backward spacing. |
| 8     | Unit available for use. |
| 9     | Unit disabled. |
| 10    | Unit currently in operation. |
| 11    | Unit waiting for an available DMA channel. |

#### 2.3 ERROR CODES

TAPTRN carried out a number of checks on the input parameter list to ensure valid requests. The code placed in IERR is generated by TAPTRN and not by the operating system. The error codes which can be set by TAPTRN are listed below.
3.0 TSTAPE DESCRIPTION

3.1 PROGRAM OVERVIEW

TSTAPE has been created to (a) facilitate the usage of the TAPTRN program and (b) be used as an aid in the analysis of computer tapes of unknown format. TSTAPE has many commands which provide full access to all the TAPTRN functions. These are as follows:

COMMAND SUMMARY (X=UNIT, DEFAULT=0)

IN(X) = INITIALIZE THE DRIVE (-****MUST BE FIRST COMMAND INPUT****)
RD(X) = READ A RECORD
WT(X) = WRITE A RECORD
DI = DISPLAY A READ/WRITE BUFFER
ST = DRIVE STATUS (LAST COMMAND)
SL = SET RECORD LENGTH
RW(X) = REWIND TAPE
NR(X) = NUMBER OF RECORDS IN THE FILE
SK(X) = SKIP
RL(X) = FIND THE RECORD LENGTH (WORDS)
EF(X) = WRITE AN EOF
TS(X) = WRITE/READ TEST
SH = SHOW THE FIRST RECORD OF EACH FILE ON THE TAPE
EX = EXIT
HE = HELP

When first using this tape package, the most helpful command is HE, the help command. This command will display the command summary shown above, on the screen.
The commands are self explanatory but a few will be expanded upon here. It should be noted that the first command that should be input is IN(X), initialize the drive. If another command is input first, before this one, an error message will appear stating that the drive has not been initialized. This command has been incorporated to accommodate multiple tape drives on the system, allowing the user to acquire one drive solely for their use.

The command DI, Display the Read/Write Buffer, has the capability to display the buffer in ASCII, integer or octal format, and to convert EBCDIC and BCD data to ASCII before displaying. The user will be queried which of these capabilities is to be utilized at the appropriate time.

For all the commands that have (X) included in them, i.e. IN(X), it is not necessary to include the (X) with the command if the user wishes to use the unit default of 0. For example, instead of entering RD(0), the user can enter RD.

3.2 COMPILING AND RUNNING THE PROGRAM

In order to run this program, enter the following command while in file manager:

:RU,TSTAPE

The initial screen display introducing the program will then appear.

Command files have been created in case it is necessary to recompile and load the program. A listing of these command files is found in Appendix E. The rebuilding of the program is done in two stages. First invoke the transfer file, RTAPE, by entering:

:TR,RTAPE

in order to rebuild the TAPTRN program. Then, invoke the transfer file, CTSSTAP, by entering:

:TR,CTSSTAP

in order to rebuild the TSTAPE program and to link and load the TSTAPE and TAPTRN programs together. The program TSTAPE, is then ready to run in the manner stated above.
4.0 SUMMARY COMMENTS

This report has described the two programs, TSTAPE and TAPTRN. TSTAPE has been described along with a description of how to compile and run the program. TAPTRN has been described along with a description of its parameter list, error codes and status return information. Further documentation of each program is provided in the comments of the program.
APPENDIX A

TAPTRN SOURCE LISTING
This is a FORTRAN callable subroutine that is designed to give
the user the capability to manipulate the tape drive in order
to read any type of tape.
The facilities provided by TAPTN are:
1) Variable record length
2) Ability to skip and backspace multiple records or files
   thus permitting the user to:
   a) skip to subsequent files without having to read through
      intervening files
   b) position the tape at the logical end of volume
   c) perform a logical rewind to position the tape at the
      beginning of a file which is not the first file on the reel
   d) provide the calling program with information on the tape
      position eg. BOT, EOF
   e) allow tape formatting eg. writing EDF, EOF
   f) return success or failure on all requested functions along
      with device status word
   g) erase four inches of tape

The calling sequence to invoke this subroutine is:

CALL TAPTN(IUNIT,KODE,KOUNT,IERR,NTRAN,IPDS,ISTAT,IBUFF)

See the program documentation for a description of the parameters.

The files used with this program are:

&TAPTN - source module
%TAPTN - relocatable module
TAPTN - runnable program
RTAPE - transfer file to compile TAPTN

Note - All these files reside on cartridge 13

To create this subroutine invoke the transfer file, RTAPE by,
TR,RTAPE
C
C Author: Suzanne Slinn
C Date: November 6, 1982
C
C******************************************************************************
C
C
C FILES(15,15)
C
SUBROUTINE TAPTN(IUNIT,KODE,KOUNT,IERR,NTRAN,IPOS,ISTAT,IBUFF)
  INTEGER IERR,ISTAT,KODE,IUNIT,ICON,LURAY(6),NUM,I,ITOPB,1CNWD,
     I
     ISTA1,LUN,IA,IB,IN,OUT,TYPE,IBUFL,ITEST,IBUFF(9950)
  LOGICAL FOUND,EOF,EDT,ILOCK
  DATA IN/1/,OUT/1/,ILOCK/.FALSE./
C
C Initialize error codes
C IERR=-2
C ISTAT=0
C NTRAN=0
C ISTA1=0
C IPPOS=0
C
C Check for valid function code
C IF ((KODE .LT. -2) .OR. (KODE .GT. 9)) RETURN
C
C Check if Magtape unit number valid
C IERR=-92
C IF (IUNIT .NE. 0) RETURN
C LUN=IUNIT+8
C IERR=-37
C IF (KODE .GE. 0)
C   IF (.NOT.(ILOCK)) RETURN
C FIN
C IERR=0
Case Function Code of
SELECT (KODE)
(-2)
Unlock
ICON=00000B
NUM=1
LURAY(1)=LUN
Unlock the LU
CALL LURQ(ICON,LURAY,NUM)
ILOCK=.FALSE.
RETURN
FIN

(-1)
Lock
ICON=00001B
NUM=1
LURAY(1)=LUN
Lock the LU
CALL LURQ(ICON,LURAY,NUM)
ILOCK=.TRUE.
RETURN
FIN

(0)
Status request
ICNWD=0
ICNWD=ICNWD+8
CALL EXEC(13,ICNWD,ISTA1)
CALL STATS(ISTA1,ISTAT)
IERR=0
RETURN
FIN
(1) Rewind tape
CALL MANIP(4,LUN)
CALL EXEC(13,LUN,ISTAT)
CALL SWTAT(ISTAT)
ISTAT=0
IPOS=1
IERR=0
RETURN
FIN

(2) Write end of file
CALL MANIP(1,LUN)
CALL EXEC(13,LUN,ISTAT)
CALL SWTAT(ISTAT)
ISTAT=0
IERR=0
IPOS=2
RETURN
FIN

(3) Skip KOUNT records forwards
I=1
EOF=.FALSE.
WHILE ((I .LE. KOUNT) .AND. (.NOT. (EOF)))
  CALL MANIP(3,LUN)
  ICNWD=8
  CALL EXC(13,ICNWD,ISTA1)
  CALL SIWAT(ISTA1)
  ITEST=IAND(ISTA1,(2**7))
  WHEN (ITEST .EQ. 2**7)
    EOF=.TRUE.
    ISTAT=7
    IPOS=2
    I=I-1
    FIN
  ELSE
    ITEST=IAND(ISTA1,(2**5))
    IF (ITEST .EQ. (2**5))
      EOF=.TRUE.
      ISTAT=5
      IPOS=3
      I=I-1
      FIN
    FIN
  I=I+1
FIN
WHEN (I .GT. KOUNT)
  NTRAN=I-1
FIN
ELSE
  NTRAN=1
FIN
IERR=0
RETURN
FIN
(4)
C
Skip KOUNT records backwards
I=1
EOF=.FALSE.
WHILE (( I .LE. KOUNT ) .AND. (.NOT. (EOF)))
C
Backward space 1 record
CALL MANIP(2,LUN)
ICNWD=8
CALL EXEC(13,ICNWD,ISTA1)
CALL STWAT(ISTA1)
ITEST=IAND(ISTA1,(2**7))
WHEN ( ITEST .EQ. 2**7 )
   EOF=.TRUE.
   ISTAT=7
   IPOS=2
   I=I-1
FIN
ELSE
   ITEST=IAND(ISTA1,(2**6))
   IF ( ITEST .EQ. (2**6) )
      EOF=.TRUE.
      ISTAT=6
      IPOS=1
      I=I-1
   FIN
FIN
I=I+1
FIN
WHEN ( I .GT. KOUNT )
   NTRAN=I-1
FIN
ELSE
   NTRAN=I
FIN
IERR=0
RETURN
FIN
C  Skip KOUNT files forwards
   I=1
   EOT=.FALSE.
   WHILE ((I .LE. KOUNT) .AND. (.NOT. (EOT)))
     Forward space a file
     CALL MANIP(I3R,LUN)
     ICNWD=8
     CALL EXEC(13,ICNWD,ISTA1)
     CALL SWAT(ISTA1)
     ITEST=IAND(ISTA1,(2**5))
     IF (IEST .EQ. 2**5) THEN
       IPOS=3
       EOT=.TRUE.
       ISTAT=5
       I=I-1
     END IF
     I=I+1
   END WHILE
   IERR=0
   WHEN (I .GT. KOUNT)
     NTRAN=I-1
   END WHEN
   ELSE
     NTRAN=1
   END ELSE
RETURN
(6)
C Skip KOUNT files backwards
I=1
EOT=.FALSE.
WHILE ((I .LE. KOUNT) .AND. (.NOT. (EOT)))
   Backward space a file
   CALL MANIP(14B,LUN)
   ICNWD=8
   CALL EXEC(13,ICNWD,ISTA1)
   CALL STWAT(ISTA1)
   ITEST=IAND(ISTA1,(2**6))
   IF (ITEST .EQ. 2**6)
      IPOS=1
      EOT=.TRUE.
      ISTAT=6
      I=I-1
   FIN
   I=I+1
FIN
IERR=0
WHEN (I .GT. KOUNT)
   NTRAN=I-1
FIN
ELSE
   NTRAN=I
FIN
RETURN
FIN

C
(7)
C Erase four inches of tape
CALL MANIP(12B,LUN)
CALL EXEC(13,LUN,ISTAT)
CALL STWAT(ISTA1)
IERR=0
RETURN
FIN
C
(U)

C Read a record
ICODE=1
ICNWD=8
IBUFL=KOUNT
CALL EXEC(ICODE,ICNWD,IBUFL,IBUFL)
CALL ABREG(IA,IB)
NTRAN=IB
CALL STATS(IA,ISTAT)
IF (ISTAT .EQ. 7) IPOS=2
IERR=0
RETURN
FIN

C

(9)

C Write a record
ICODE=2
ICNWD=8
IBUFL=KOUNT
CALL EXEC(ICODE,ICNWD,IBUFL,IBUFL)
CALL ABREG(IA,IB)
CALL STATS(IA,ISTAT)
IERR=0
RETURN
FIN
FIN
RETURN
END

C

C******************************************************************************

C This subroutine continually polls the status of the operation
C until the current operation is complete.
C******************************************************************************

C
SUBROUTINE STWA1(ISTAT)
  INTEGER ISTAT
  CONTINUE
    ITEST=(ISTAT .AND. 32768)
    IF (ITEST .EQ. 0) RETURN
    CALL EXEC(13,0,ISTAT)
    GO TO 15
END

C
C******************************************************************************
C  This subroutine performs the I/O control calls such as rewind, write end of file.
C******************************************************************************
C
SUBROUTINE MANIP(FUNC,LUN)
  INTEGER FUNC,LUN,ICNW
  ICNW=0
  Shift the function code six bits to the left in ICNW
  ICNW=ICNW+ISHFT(FUNC,6)
  ICNW=ICNW+LUN
  CALL EXEC(3,ICNW)
  RETURN
END

C
C******************************************************************************
C  This subroutine sets the appropriate bits in ISTAT for status return information.
C******************************************************************************
C
SUBROUTINE STATS(ISTA1,ISTAT)
  INTEGER CHECK,ISTAT,I,ISTA1
  LOGICAL FOUND
  CHECK=0
  Set the 14 and 15 bits to 1 in CHECK
CHECK=CHECK+16384
CHECK=CHECK+32768
C
Save bits 14 and 15
CHECK=IAND(CHECK,ISTA1)
C
no bits set
IF (CHECK .EQ. 0) ISTAT=8
C
14 bit set
IF (CHECK .EQ. 16384) ISTAT=9
C
15 bit set
IF (CHECK .EQ. 32768) ISTAT=10
C
14 and 15 bit set
IF (CHECK .EQ. 49152) ISTAT=11
I=1
FOUND=.FALSE.
C
Find the STATUS bit set for bits 1-7
WHILE (( I.LE. 7) .AND. (.NOT. (FOUND)))
    CHECK=0
    CHECK=IAND(ISTA1,(2**I))
    IF (CHECK .EQ. 2**I)
        ISTAT=I
        FOUND=.TRUE.
FIN
    I=I+1
FIN
RETURN
END
APPENDIX B

PSEUDOCODE

Subroutine TAPTRN

Initialize error codes and other variables
If (Kode < -2 or Kode > 9)
then
    Return indicating KODE range error
endif
If (Iunit ≠ 0)
then
    Return indicating invalid logical unit number error
endif
If (Kode ≥ 0 and unit not locked)
then
    Return indicating resource not locked error
endif
Case Function Code of
   -2: Unlock the logical unit
        Set locked flag to false
        Return
   -1: Lock the logical unit
        Set locked flag to true
        Return
0: Obtain present drive status
    Set ISTAT to reflect status
    Return

1: Manipulate the bits for rewind code
    Obtain present driver status
    Wait until the driver is available for use
    Return

2: Manipulate the bits for end-of-file code
    Obtain present driver status
    Wait until the driver is available for use
    Return

3: Initialize loop counters
    While (record counter < KOUNT and NOT(EOF))
        Manipulate bits to space 1 record forwards
        Obtain present driver status
        Wait until device is ready for use
        If end of file record bit is set then
            EOF = True
            Set ISTAT to reflect EOF
            Set IPOS to reflect EOF
            Decrement record counter
        Else
            If end-of-tape marker bit is set then
                EOT = True
                Set ISTAT to reflect EOT
                Set IPOS to reflect EOT
                Decrement record counter
            endif
            Increment record counter
        Endif
    Endwhile
    If (Record counter > KOUNT)
        NTRAN = record counter - 1
    else
        NTRAN = record counter
    endif
    Return
4: Initialize loop counters
   While (Record counter ≤ KOUNT and NOT(EOF))
   Manipulate bits to backspace one record
   Obtain present driver status
   Wait until device is ready for use
   If end of file record bit is set
      then
         EOF = True
         Set ISTAT to reflect EOF
         Set IPOS to reflect EOF
         Decrement record counter
      else
      If start of tape marker bit is set
      then
         EOF = True
         Set ISTAT to reflect BOT
         Set IPOS to reflect BOT
         Decrement record counter
      endif
   endif
Endwhile
If (Record counter > KOUNT)
then
   NTRAN = record counter - 1
else
   NTRAN = record counter
endif
Return

5: Initialize loop counters
   While (Record counter ≤ KOUNT and NOT (EOT))
   Manipulate bits for forward space one file
   Obtain present driver status
   Wait until device is ready for use
   If end-of-tape marker bit is set
   then
      EOT = True
      Set ISTAT to reflect EOT
      Set IPOS to reflect EOT
      Decrement record counter
   endif
   Increment record counter
Endwhile
If (Record counter > KOUNT)
then
   NTRAN = record counter - 1
else
    NTRAN = record counter
endif
Return

6: Initialize loop counters
  While (Record counter ≤ KOUNT and NOT (EOT))
    Manipulate bits to backspace one file
    Obtain present driver status
    Wait until device is ready for use
    If start-of-tape marker bit is set then
      EOT = True
      Set ISTAT to reflect EOT
      Set IPOS to reflect EOT
      Decrement record counter
    endif
    Increment record counter
Endwhile
If (Record counter > KOUNT) then
    NTRAN = record counter - 1
else
    NTRAN = record counter
endif
Return

7: Manipulate bits to erase four inches of tape
  Obtain present driver status
  Wait until device is ready for use
  Return

8: Read the record
  Determine the present driver status
  Return

9: Write the record
  Determine the present driver status
  Return

End Case on Function Code
Return
End Subroutine TAPTRN
Subroutine STWAT

  c Subroutine to wait for driver to be ready for use
  While driver not available for use
    Obtain present driver status
  Endwhile
  Return
End Subroutine STWAT

Subroutine MANIP

  c Subroutine to manipulate bits for desired function code
  Shift function code six bits to the left in ICNWD
  Add the logical unit number to ICNWD
  Perform the function
  Return
End Subroutine MANIP

Subroutine STATS

  c Subroutine to set appropriate bits in ISTAT to reflect
  c present driver status
  If neither the 14th nor 15th bits in driver status are set then
    ISTAT=8
  If 14th bit set in driver status then
    ISTAT=9
  If 15th bit set in driver status then
    ISTAT=10
  If 14th and 15th bits are set in driver status then
    ISTAT=11
Initialize loop counters
While (Present bit ≤ 7 and NOT FOUND)
   If Present bit in driver status set then
      ISTAT = Present bit
      FOUND = True
   endif
   Present bit = Present bit +1
Endwhile
Return
End Subroutine STATS
This is a FORTRAN program that has been created to facilitate the usage of the TAPTRN program. TSTAPE has several commands which provide full access to all the TAPTRN functions. To obtain a list of the commands, type HE in response to the INPUT COMMAND prompt when the program runs.

To invoke this program, type RU,TSTAPE.

The files used with this program are:

&TSTAPE - source program
XSTAPE - relocatable module
LTSTAPE - transfer file for LOADR
CTSTAPE - transfer file to compile and load TSTAPE
Note - all these files reside on cartridge 13

To create this program invoke the transfer file, CTSTAP by,
:TR,CTSTAPE
AFTER building the subroutine TAPTRN.

To run this program after invoking the transfer file, CTSTAPE, type,
:RU,TSTAF

Author: Suzanne Y. Slinn
Date: November 6, 1982
$FILES(15,15)

PROGRAM TSTAPE

INTEGER BCM(2),FM,A10,I10,O10,EA0,RA0,IR0,YES,ICOMD,IR,IT
INTEGER BF1(120),BF2(120),BF3(240),VT(6)
INTEGER NAME(40),IDIT(2),ILOCK(2),VOL
DIMENSION IB1(9950),INIT(2),POSN(5),ICMD(19)
EQUIVALENCE (BF3,BF1),(BF2(121),BF2),(IB1(1),IDIT(1))
DATA ICM/"IN","SH","RD","WT","DI","SI","SL","CO",
1 'RW','NR','SK','RL','EF','TS','EX','HE','DU','LE',
2 'SA'/
DATA INIT/2*0/
DATA A10/’A’/,I10/’I’/,O10/’O’/,EA0/’E’/,RA0/’R’/,IR0/’I’/,
1 YES/’Y’/,IU,K1,IR,NT,IP,IS/6*0/
DATA VT/015452R,0621068,0155108,015512R,015452R,0621058/
DATA POSN/’,’/B0T’,’EOF’,’EDT’,’EOV’/
DATA ILOCK/015554R,015555B/

C Clear the screen, home the cursor, display initial messages.
WRITE(1,490) (VT(I),I=1,6)

490 FORMAT(6A2,/,’TSTAPE: THIS PROGRAM IS INTENDED AS ’,/, 1 ’A) A TEST PROGRAM FOR THE SUBROUTINE TAPTRN AND ’,/, 2 ‘B) A UTILITY PROGRAM TO AID IN THE ANALYSIS OF ’,/, 3 ’COMPUTER TAPES OF UNKNOWN FORMAT.’,/, 4 ‘**TYPE (HE)LP TO DISPLAY THE AVAILABLE COMMANDS.’)
IC=1
ID=0

C Issue the desired command
C Move cursor to display line
110 CALL KLOAD(22,0)
WRITE(1,500)
READ(1,A10)ICM,IN'M(?)
485 FORMAT(A2,A1)
ICL=IC
IUL=ID
DO 100 I=1,17
100 CONTINUE
120   CALL KLOAD(22,0)
125   WRITE(1,510)
126   CALL TIME
127   GO TO 110

C A valid command has been identified
105   IB=BCM(2)
106       ID=0
107       IF (IB .EQ. 2H0 ) ID=0
108       IF (IB .EQ. 2H1 ) ID=1
109       IF (( ID .LT. 0 ) .OR. (ID .GT.1)) GO TO 120
110       IF (ID .EQ. 0) IU=0
111       IF (ID .EQ. 1) IU=1
112       I *I

C Determine if it is the initialization command or a command
C that does not determine a previous initialization.
113       IF ((IC .EQ. 1) .OR. (IC .GT. 14)) GO TO 125
114       IF (INIT(IU+1) .EQ. 1) GO TO 125
115       CALL KLOAD(22,0)
116       WRITE(1,515)
117       CALL TIME
118       GO TO 110

C Case command of:
C
125   CALL KLOAD(22,0)
126   WRITE(1,1111)ILock(1)
127   GO TO (5,10,15,20,25,30,35,40,45,50,55,60,65,70,75,80,85
1     ,90,95),IC

C **IN**, Initialize tape command
S   CALL TAPTN(IU,-1,KT,IR,NT,IP,IS,IB1)
S   INIT(IU+1)=1
S   ICL=IC
S   IUL=ID
S   GO TO 30
C           *SH*, Show the contents of the first record of each file
10  KT=9950
    VOL=0
    CALL KLOAD(22,0)
    WRITE(1,541)
    READ(1,505)FM
    CALL TAPTN(IU,1,KT,IR,NT,IP,IS,IB1)
11  CONTINUE
C            Repeat until EOF is true
    IF (VOL .EQ. 1) GO TO 30
    IP=0
    CALL TAPTN(IU,8,KT,IR,NT,IP,IS,IB1)
    IF ((IP .EQ. 2) .OR. (IP .EQ. 3)) VOL=1
    IF (VOL .EQ. 1) GO TO 30
C            If not end of tape, write out first record of file
    WRITE(18,500)NT
    IF (FM .EQ. A10) WRITE(18,545)(IB1(I,1),I=1,NT)
    IF (FM .EQ. I10) WRITE(18,550)(IB1(I,1),I=1,NT)
    IF (FM .EQ. O10) WRITE(18,555)(IB1(I,1),I=1,NT)
C            IF (FM .EQ. EAO) CALL EBCDIC(IB1,NT)
    IF (FM .EQ. BAO) CALL BCD(IB1,NT)
    IP=0
12  CONTINUE
13  IF (IP .EQ. 2) GO TO 13
    CALL TAPTN(IU,8,KT,IR,NT,IP,IS,IB1)
    GO TO 12

*RD*, Read a record

CALL KLOAD(22,0)
WRITE(1,530)

Get the count words to be read
READ(1,525)KT
IF ((KT .EQ. 0) .OR. (KT .GT. 9950)) KT=9950
DO 16 I=1,KT
16  IBI(I)=0

Read one record of the desired length
CALL TAPTN(IU,8,KT,IR,NT,IP,IS,IB1)
IF (IP .EQ. 0) GO TO 14
CALL KLOAD(22,0)
WRITE(1,535)
CALL TIME
14 CALL KLOAD(22,0)
WRITE(1,635)

FORMAT(' DUMP TO LINE PRINTER? (Y)=YES _')
IO=16

KT=NT

Determine if dump to line printer desired.
READ(1,505)BF1(I)
IF (BF1(I) .EQ. YES) GO TO 24
GO TO 30

*WT* Write a record

CALL KLOAD(22,0)
WRITE(1,530)

Get number of words to write
READ(1,525)KT
IF ((KT .EQ. 0) .OR. (KT .GT. 9950)) KT=9950
CALL KLOAD(22,0)
WRITE(1,532)

FORMAT(' ENTER POSITION IN BUFFER OF WHERE TO START WRITING _')

Get position in buffer of where to start writing data to
READ(1,525)ISTRT
I**=KT=ISTRT-1
DO 22 I=1,KT
22   IB(I)=ISTRI+I
C   Write one record of the desired length
   CALL TAPIN(10,9,KT,IR,N1(IP,IS,IK1))
   GO TO 30
C
C......*DI* Display the read buffer
25   IO=1
24   CALL KLOAD(22,0)
   WRITE(1,541)
   READ(1,505)FM
   WRITE(1,1111)ILOCK(2)
   WRITE(1,640) (VT(I),I=1,6)
C
C   Determine which form of output is desired.
   IF (FM .EQ. AIO) GO TO 26
   IF (FM .EQ. IIO) GO TO 27
   IF (FM .EQ. OIO) GO TO 28
   IF (FM .EQ. EAO) GO TO 29
   IF (FM .EQ. BAO) GO TO 31
   IF (FM .EQ. TR0) GO TO 32
   GO TO 120
C
C   Ascii data
26   CALL KLOAD(22,0)
   WRITE(1,650)
   READ(1,505)ICOMD
   IF (ICOMD .NE. YES) GO TO 37
   WRITE(1,640) (VT(I),I=1,6)
   WRITE(IO,545) (IB(I),I=1,KT)
   IF (IO .EQ. 18) GO TO 30
   GO TO 110
37   LW=40
   IF (ICOMD .NE. YES)LW=66
38   WRITE(1,640) (VT(I),I=1,6)
   WRITE(IO,545) (IB(I),I=1,NT)
   IF (IO .EQ. 18) GO TO 30
   GO TO 110
C      Integer data
27    WRITE(IO,550) (IB1(I),I=1,NT)
     IF (IO .EQ. 18) GO TO 30
     GO TO 110
C      C*al data
26    WRITE(IO,555) (IB1(I),I=1,NT)
     IF (IO .EQ. 18) GO TO 30
     GO TO 110
C      Ebcidic data
29    CONTINUE
     CALL EBCDC(IB1,KT)
     GO TO 26
C      Bcd data
31    CONTINUE
     CALL BCD(IB1,KT)
     GO TO 26
C      Convert to octal data
32    CONTINUE
     CALL CONVRT(IB1,KT)
C      WRITE(IO,555)(IB1(I),I=1,KT)
     IF (IO .EQ. 18) GO TO 30
     GO TO 110
C       *ST* Status of the drive
30    WRITE(1,1111)ILOCK(2)
     WRITE(1,560)(VT(I),I=1,6),INL,ICMD(INL),KCONFI,KT,NT,
1     POSN(IP+1),IN,15
     GO TO 110
C       *SL* Set record length
35    CALL KLOAD(22,0)
     WRITE(1,565)
     READ(1,525)KT
     IF ((KT .EQ. 0) OR (KT .GT. 9950)) KT=9950
     GO TO 30
C
40    GO TO 110
C......*RW* Rewind tape
45 CALL TAPTNT(IU,1,KT,IR,NT,IP,IS,IB1)
   GO TO 30

C......*NR* Get the number of records in a file
50 IF ((IP .EQ. 1) .OR. (IP .EQ. 2)) GO TO 51
   CALL KLOAD(22,0)
   WRITE(1,575)
   CALL TIME
   GO TO 30

C
51 K=0
   KS=KT
   KT=9950
C Read one record
54 CALL TAPTNT(IU,B,KT,IR,NT,IP,IS,IB1)
   IF ((K .EQ. 0) .AND. (IP .EQ. 2)) GO TO 52
   IF (IP .EQ. 2) GO TO 53
   K=K+1
   NTL=NT
   GO TO 54
52 CALL KLOAD(22,0)
   WRITE(1,580)
   CALL TIME
   GO TO 30
53 WRITE(1,1111) ILOCK(2)
   CALL KLOAD(21,0)
C Write out the number of records found
   WRITE(1,1111) ILOCK(1)
   WRITE(1,585) K,NTL
   CALL TIME
   KT=KS
   GO TO 30
C C.... *SK* Skip to end of file
55 CALL KLOAD(22,0)
   WRITE(1,625)
   READ(1,505)ICOMD
   IK=3
   IF (ICOMD .NE. YES) IK=5
   CALL KLOAD(22,0)
   WRITE(1,630)
   READ(1,525)KT
   IF (KT .GE. 0) GO TO 56
   IK=IK+1
   KT=-KT
56 CALL TAPTN(IU,IK,KT,IR,NT,IP,IS,IB1)
   GO TO 30
C C.... *RL* Find record length
60 KT=9950
   CALL TAPTN(IU,8,KT,IR,NT,IP,IS,IB1)
   CALL KLOAD(22,0)
   WRITE(1,590)NT
   CALL TIME
   GO TO 30
C C.... *EF* Write end of file
65 CALL TAPTN(IU,2,KT,IR,NT,IP,IS,IB1)
   GO TO 30
C C.... *TS* TEST READ WRITE/READ PATTERN
70 CALL KLOAD(22,0)
   WRITE(1,530)
   READ(1,525)KT
   CALL KLOAD(22,0)
   WRITE(1,600)
   READ(1,605)NW,INC
   CALL KLOAD(22,0)
WRITE(1,620)
READ(1,505)ICOMD
IF(ICOMD .NE. YES) GO TO 71
CALL KLOAD(22,0)
WRITE(1,540)
READ(1,505)FM

C
71 KT1=KT
DO 81 I=1,NW
   DO 82 J=1,KT1
      IB1(J)=J
      IF (ICOMD .NE. YES) GO TO 72
      IF (FM .EQ. AID) WRITE(18,545)(IB1(L),L=1,KT1)
      IF (FM .EQ. IIO) WRITE(18,550)(IB1(L),L=1,KT1)
      IF (FM .EQ. OID) WRITE(18,555)(IB1(L),L=1,KT1)
   C Write a record
   72 CALL TAPTN(IU,9,KT1,IR,NT,IP,IS,IB1)
      IF (IR .EQ. 0) GO TO 83
      WRITE(1,1111)ILOCK(2)
      WRITE(1,560)(VT(J),J=1,6),IUL,ICMD(ICL),KCONF,KT1,NT,
      POSN(IP+1),IR,IS
   C KT1=KT1+INC
      IF (((KT1 .LE. 0) .OR. (KT1 .GT. 9950)) GO TO 84
   81 CONTINUE
C
C Write 2 end of files/rewind and read
C
84 CALL TAPTN(IU,2,KT,IR,NT,IP,IS,IB1)
   CALL TAPTN(IU,2,KT,IR,NT,IP,IS,IB1)
   CALL TAPTN(IU,1,KT,IR,NT,IP,IS,IB1)
C
   IUL=IU
   KT1=KT
   DO 86 I=1,NW
      DO 89 J=1,KT1
         IB1(J)=0
      C
C Read a record
CALL TAPTN(IU,B,KT1,IR,NT,IP,IS,IB1)
IF (ICMD.NE.YES) GO TO 73
IF (FM.EQ.AI0) WRITE(10,545)(IB1(L),L=I,KT1)
IF (FM.EQ.1IO) WRITE(10,550)(IB1(L),L=I,KT1)
IF (FM.EQ.010) WRITE(10,555)(IB1(L),L=I,KT1)

73 IEC=0
DO 87 J=1,KT1
   IF (IB1(J).NE.J) IEC=IEC+1

87 CONTINUE
C
IF (IR.EQ.0) GO TO 88
WRITE(1,1111)ILOCK(2)
WRITE(1,560)(VT(J),J=1,6),IUL,ICMD(ICL),KCODE,KT1,
1 NT,POSN(IP+1),IR,IS

88 IF (IEC.EQ.0) GO TO 92
   CALL KLOAD(22,0)
   WRITE(1,610)IEC

92 KTI=KTI+1
   IF ((KTI.LE.0).OR.(KTI.GT.975)) GO TO 110

86 CONTINUE
CALL TAPTN(IU,1,KT,IR,NT,IP,IS,IB1)
GO TO 110
C
C.....*EX* Exit
75 WRITE(1,1111)ILOCK(2)
   CALL TAPTN(IU,-2,KT,IR,NT,IP,IS,IB1)
   STOP
C
C.....*HE* Help facility
80 WRITE(1,1111)ILOCK(2)
1111 FORMAT(A2)
   WRITE(1,640)(VT(I),I=1,6)
   OPEN(33,FILE='HELINS',IOSTAT=IOS)

998 CONTINUE
READ(33,222,ERR=111,END=111)(BF1(I),I=1,80)
222 FORMAT(80A1)
223 FORMAT(1X,72A1)
GO TO 998
111 CONTINUE
110 OPEN(33)
GO TO 110
C
85 GO TO 110
90 GO TO 110
95 GO TO 110
C
C......Format statements.
500 FORMAT( 'INPUT COMMAND ',_')
505 FORMAT(2A1)
510 FORMAT( 'INVALID COMMAND*')
515 FORMAT( 'INITIALIZE DRIVE*')
520 FORMAT( 'INPUT PARITY (O)DD,(E)VEN _')
525 FORMAT(I7)
530 FORMAT( 'INPUT RECORD SIZE (DEFAULT, KT=9950 ) _')
535 FORMAT( 'EOF MARKER READ*')
540 FORMAT( 'FORMAT? (A)SCII,(I)NTEG,(O)CTAL,(E)BC OR (B)CD TO ,
1 ASCII _')
541 FORMAT( 'FORMAT? (A)SCII,(I)NTEG,(O)CTAL,(E)BC OR (B)CD TO ,
1 ASCII _')
545 FORMAT(2X,30A2)
550 FORMAT(12X,8I7)
555 FORMAT(12X,807)
560 FORMAT(6A2,/,35X,' TAPE STATUS',//,' UNIT',7X,I6,11X,
1 ' LAST COMMAND',5X,A2,/, ' PAR/DENS ',I6,11X,
2 'KOUNT',7X,I7,11X,'TRAN',7X,I7,/, 'POSN',9X,A4,11X,
3 'ERR',8X,I7,11X,'ISTAT',7X,I7)
565 FORMAT( 'INPUT KOUNT SIZE(DEFAULT KT=9950 ) _')
570 FORMAT( 'ROUTINE NOT WRITTEN*')
575 FORMAT( 'TAPE IS NOT AT BOT OR EOF*')
580 FORMAT(' *END OF TAPE*')
585 FORMAT(' NO. OF RECORDS (EOF EXCLUDED)',/,'1X',
1 ' SIZE LAST DATA RECORD',/,'1X')
590 FORMAT(' NO. OF WORDS READ=',/,'1X')
600 FORMAT(' INPUT NO OF RECORDS AND INC/DEC (7 DIGITS) _')
605 FORMAT('I7,1X,17')
610 FORMAT(' RECORDS NO=',/,'1X', 'NO OF READ ERRORS=',/,'1X')
620 FORMAT(' DO YOU WANT BUFFER DUMPS? (Y)=YES _')
625 FORMAT(' RECORD SKIP? (Y)=YES _')
630 FORMAT(' INPUT KOUNT,NEGATIVE MEANS BACKSKIP _')
640 FORMAT('6A2')
650 FORMAT(' WITH CARRIAGE CONTROL? (Y)=YES _')
655 FORMAT(' TWO LINE WIDTH NOT AVAILABLE _')
660 FORMAT('A2')
665 FORMAT(' TAPE IS MOUNTED AND DRIVE INITIALIZED? (Y)=YES _')
670 FORMAT(' IS SOURCE TAPE ON DRIVE ZERO? (Y)=YES _')
675 FORMAT(' DO YOU WANT A CODE CONVERSION? (Y)=YES _')
680 FORMAT(' WHICH ONE? (E)BC OR (B)CD TO ASCII? _')
685 FORMAT(' DO YOU WANT A MAP DUMP OF THE TAPE? (Y)=YES _')
690 FORMAT('616')
695 FORMAT(' DEFAULT MAP IS TO SPOOL? (Y)=YES _')
7000 FORMAT(' FILENAME? _')
7005 FORMAT('40A1')
7010 FORMAT(' RECORD SIZE ON OUTPUT (MAX=132)? _')
7015 FORMAT('I6')
7020 FORMAT('132A1')
END
C THIS SUBROUTINE GETS THE CURRENT SYSTEM TIME

C

SUBROUTINE TIME
INTEGER ITIME(5),SEC,DIFF
CALL EXEC(11,ITIME)
SEC=ITIME(2)
DIFF=0
1 CONTINUE
IF (DIFF .GT. 2) GO TO 2
   CALL EXEC(11,ITIME)
   DIFF=ITIME(2)-SEC
GO TO 1
2 CONTINUE
RETURN
END
C
C
C
C
C
KLOAD

SUBROUTINE KLOAD(IRL, ICL)
IMPLICIT REAL*8(A-H,O-Z)
INTEGER ICA(2), ICB(I), ICC(2), ARAY(24), IC(24)
DATA ICA/0154468,0804008/
DATA ICB/0710008/
DATA ICC/0414008,0155128/
DO 1 I=1,7
  IC(I)=2H
  ENCODE(24,500,IC) (ICA(I),I=1,2), IRL, ICR, ICL, (ICC(I),I=1,2)
  FORMAT(262,12,A2,12,A2)
  WRITE(1,510)(IC(I),I=1,7)
  FORMAT(1X,7A2)
1  RETURN
END
& EBCDIC

FTN4

C THE FOLLOWING SUBROUTINE ACCEPTS AN ARRAY OF WORDS (MAXIMUM SIZE 9950)
C Coded in EBCDIC and converts each word to ASCII code. The size of the
C ARRAY IS PASSED THROUGH PARAMETER "NO" AND THE RESULTANT ASCII CODE IS
C PASSED BACK THROUGH THE PARAMETER "BUFF".

SUBROUTINE EBCDIC(BUFF,NO)
INTEGER CD1,CD2
INTEGER ASC(255),NO,I,TEMP1,BUFF(9950),HIGH,LOW
DATA ASC/1B,2B,3B,0B,11B,0B,177B,0B,0D,0R,13B,14B,15B,16B,17B,
  20B,21B,22B,0R,0B,10B,30B,31B,0R,0B,0B,0B,0B,0B,
  0B,0B,34B,0B,0B,12B,27B,33B,0R,0B,0R,0B,0B,5B,6B,7B,
  0B,0R,26B,0B,0R,36B,0R,4B,0R,0B,0R,24B,25B,0R,32B,
  40B,40B,40B,40B,40B,40B,40B,40B,40B,40B,40B,40B,56B,74B,50B,
  53B,174B,46B,40B,40B,40B,40B,40B,40B,40B,40B,
  41B,44B,52B,51B,73B,40B,55B,57B,40B,40B,40B,
  40B,40B,40B,40B,40B,40B,40B,40B,40B,140B,72B,43B,100B,47B,
  75B,42B,40B,141B,142B,143B,144B,145B,146B,147B,
  150B,151B,40B,40B,40B,40B,40B,40B,40B,152B,153B,154B,
  155B,156B,157B,160B,161B,162B,40B,40B,40B,40B,40B,
  40B,40B,40B,133B,40B,40B,40B,40B,40B,40B,40B,
  40B,40B,40B,40B,135B,40B,40B,173B,101B,102B,103B,
  104B,105B,106B,107B,110B,111B,40B,40B,40B,40B,40B,
  40B,40B,40B,40B,40B,134B,135B,123B,124B,125B,126B,
  127B,130B,131B,132B,40B,40B,40B,40B,40B,60B,61B,
IF (NO.EQ.0) GOTO 6


DO 8 I=1,NO
  TEMP1=BUFF(I)
  HIGH = ISHIFT(TEMP1, -8)
  LOW = ISHIFT(HIGH, 8)
  LOW = TEMP1 - LOW
  CALL QTOD(HIGH)
  CALL QTOD(Low)
  CD1=ASC(HIGH)
  CD2=ASC(LOW)
  BUFF(I)=ISHFT(CD1,8)+CD2
8 CONTINUE
6 RETURN
END

C THE FOLLOWING SUBROUTINE CONVERTS AN OCTAL NUMBER TO A DECIMAL NUMBER

SUBROUTINE QTOD(NUM)
  INTEGER NUM,J,N,R
  J = NUM
  J = J/100B
  N = J*64
  R = NUM - J*100B
  J = R/10B
  N = N + J*8
  R = R - J*10B
  NUM = N + R
RETURN
END
FTN4

SUBROUTINE BCD(HBUFF,NO)

C THE FOLLOWING SUBROUTINE ACCEPTS AN ARRAY OF WORDS STORED IN "BUFF" AND AN
C INTEGER "NO" REPRESENTING THE NUMBER OF WORDS IN BUFF. EACH WORD CONTAINS
C TWO CHARACTERS IN BCD CODE REPRESENTED IN STANDARD FORMAT WHICH ARE BROKEN
C DOWN INTO SEPARATE BCD CODES, CONVERTED TO ASCII CODE BY MEANS OF A TABLE,
C AND THEN STORED TWO CHARACTERS PER WORD IN THE STANDARD FORMAT BACK IN THE
C THE ARRAY BUFF FROM WHERE IT WAS TAKEN.

INTEGER BCI(63),BUFF(9950),NO,I,NEW,TEMP1,TEST

C THE LARGEST ARRAY ACCEPTED BY THE SUBROUTINE BCD IS 9950

INTEGER CD1,CD2

DATA BCT/061B,062B,063B,064B,065B,066B,067B,070B,
1 071B,072B,073B,100B,072B,076B,075B,040B,
1 057B,123B,124B,125B,126B,127B,130B,131B,
1 132B,137B,054B,050B,047B,134B,042B,055B,
1 112B,113B,114B,115B,116B,117B,120B,121B,
1 122B,041B,044B,052B,135B,073B,045B,053B,
1 101B,102B,103B,104B,105B,106B,107B,110B,
1 111B,077B,056B,051B,133B,074B,136B/

IF (NO .EQ. 0) GOTO 6

DU 8 I=1,NO

TEMP1=BUFF(I)

NEW=40000B

5 NEW=NEW-40000B

TEST=TEMP1-NEW

IF (TEST.GE.00000B) GOTO 3

GOTO 5
C TEST IF THE VALUE OF THE SECOND BCD CODE NUMBER
C NEW IS THE FIRST BCD CODE NUMBER TO BE SHIFTED RIGHT 8 BITS
3   NEW=ISHFT(NEW,-8)
    CD1=BCT(NEW)
    CD2=BCT(NEW)
C CD1 AND CD2 CONTAIN THE ASCII EQUIVALENTS OF NEW AND TEST
     BUFF(I)=ISHFT(CD1,8)+CD2
8   CONTINUE
6   RETURN
END
APPENDIX D

PROGRAM TSTAPE

Display start up message on the console
Repeat
   Move the cursor to line 22
   Read in the desired command
   If not a valid command
      then
         Display error message
   else
      If the drive has not yet been initialized
         then
            Display error message
      else
         Move cursor to line 22
         Turn memory lock on
         Case Command of
            In: Initialize the tape drive
               Call TAPTRN (KODE = -1)
               Display Status
            SH: Show the contents of the first record of each file
               End-of-volume = False
               Move cursor to line 22
               Determine the format the record is to be displayed in
               Rewind the tape
               Repeat
                  Read a record
                  If EOV encountered
                     then
                        End-of-volume = True
                  endif
                  If not EOV
                     then
Write out number of words transferred
Display record of format desired
While not EOF
    Read next record
Endwhile
endif
Until EOV
Display status

RD: Read a record
Move cursor to line 22
Read in the number of words to be read
    (Default is 9950 words)
Initialize buffer to zeroes
Call TAPTRN (Kode = 8)
Move cursor to line 22
If end of file marker read then
    Display appropriate message
else
    Determine if buffer read is to be dumped to the line printer
    If buffer is to be dumped then
        Move cursor to line 22
        Turn off memory lock
        If ASCII output desired then
            Output data
            Display status
        endif
        If Integer output desired then
            Output data
            Display status
        endif
        If Octal output desired then
            Output data
            Display status
        endif
        If EBCDIC output desired then
            Call EBCDIC
            Output converted ASCII data
            Display status
        endif
    endif
Endif
If BCD output desired then
  Call BCD
  Output converted ASCII data
  Display status
endif
else
  Display status
endif

WT: Write a record
  Move cursor to line 22
  Determine number of words in the record that are to be written
  Determine starting place in the write buffer
  Call TAPTRN (KODE = 9)

DI: Display the read buffer
  Move cursor to line 22
  Determine which form of output is desired
  If ASCII output desired then
    Output data
    Display status
  endif
  If Integer output desired then
    Output data
    Display status
  endif
  If Octal output desired then
    Output data
    Display status
  endif
  If EBCDIC data desired then
    Call EBCDIC
    Output converted ASCII data
    Display status
  endif
  If BCD output desired then
    Call BCD
    Output converted ASCII data
    Display status
  endif
ST: Status of the drive
   Turn off memory lock
   Display status of the last command

SL: Set record length
   Move cursor to line 22
   Read in desired record length
   Display status

RW: Rewind tape
   Call TAPTRN (Kode = 1)
   Display status

NR: Number of records in a file
   If not (BOT) or
      not (EOF) display
      error message
   endif
   Record counter = 0, EOF = False
   Repeat
      Call TAPTRN (Kode = 8)
      If end of file has been reached
      then
          Display the number of records read and the
          length of the last data record read
          EOF = True
      else
      Increment record counter
      endif
   Until EOF = True
   Display status

SK: Skip to end of file
   Move cursor to line 22
   Determine if record or file skip
   Determine number of records/files to skip
   If record skip
   then
      Kode = 3
   else
      Kode = 5
   endif
   If number of records/files to skip < 0
   then
      Kode = Kode +1
      Number of records/files to skip =
      -(Number of records/files to skip)
endif
Call TAPTRN (Kode)
Display status

RL: Record length
Call TAPTRN (Kode = 8)
Move cursor to line 22
Display the number of words read
Display status

EF: Write End of File marker
Call TAPTRN (2)
Display status

TS: Test Read/Write Pattern
Move cursor to line 22
Determine length of record to write
Determine the number of records to be written
Determine if buffer dumps desired
For I=1 to Number of records to be written
  If buffer dumps desired
  then
    Output buffer in appropriate form
    endif
  endif
  Call TAPTRN (Kode = 9)
  Increment length of record to write
Endfor
Call TAPTRN (Kode = 2)
Call TAPTRN (Kode = 2)
Call TAPTRN (Kode = 1)
For I=1 to Number of records to be read
  Call TAPTRN (Kode = 8)
  If buffer dumps desired
  then
    Output buffer in appropriate form
    endif
  endif
  Increment length of record to read
Endfor
Call TAPTRN (Kode = 1)

EX: Exit
Turn off memory lock
Call TAPTRN (Kode = -2)
Halt

HE: Help facility
Open external file
Read command summary for external file
Close file
End Case (of commands)
endif (valid command)
Until Exit command chosen

End Program TSTAPE
SUBROUTINE TIME

The following subroutine gets the current system time.
This subroutine is used in order that all messages displayed on the
console to the user are present for 2 seconds.

Subroutine TIME

Obtain the current system time
Sec = current second
Difference = 0
Repeat

Obtain current system time
Difference = current second - Sec

Until difference > 2
Return

End Subroutine TIME

SUBROUTINE BCD (BUFF, NO)

The following subroutine converts an array of BCD characters to their
corresponding ASCII characters.
The characters to be converted are input in BUFF and the number of
characters to convert are input in NO.
If there are characters to convert then

For I=1 to NO

Temp = BUFF(I)
Get the upper 8 bits and shift left
Look up this number in the table for ASCII equivalent
Get lower 8 bits and look up the ASCII equivalent
Store the ASCII equivalent back into BUFF(I)

Endfor
endif
End Subroutine BCD

SUBROUTINE &EBCDIC (BUFF, NUMBER)

The following subroutine accepts an array of words, BUFF, (maximum size
9950) coded in EBCDIC and converts each word to ASCII code. The size
of the array is passed through the parameter "NUMBER" and the resultant
ASCII code is passed back through the parameter "BUFF".
If NUMBER > 0 then

For I=1 TO NUMBER DO
TEMP = BUFF(I)
HIGH = TEMP SHIFTED RIGHT 8 BITS
LOW = HIGH SHIFTED LEFT 8 BITS
LOW = TEMP-LOW
CONVERT HIGH TO A DECIMAL NUMBER
CONVERT LOW TO A DECIMAL NUMBER
CONVERT1 = ASCII EQUIVALENT OF EBCDIC HIGH
CONVERT2 = ASCII EQUIVALENT OF EBCDIC LOW
BUFF(I) = CONVERT1 SHIFTED LEFT 8 BITS + CONVERT2
Endfor
endif
Return
End Subroutine

SUBROUTINE &KLOAD (Row,Column)
c This subroutine moves the cursor to the desired row and column.  
c These values are given in the input parameters Row and Column.
   Clear Buffer
   Encode the octal commands for moving the cursor to the
   desired position into Buffer
   Write out Buffer to the screen
   Return
End Subroutine
RTAPE T=00004 IS ON CR SY USING 00002 BLKS R=0000

0001 **This transfer file compiles the TAPTRN program.
0002 :RU,FLECS,&TAPTRN,&TAPE:13,TAPE:13
0003 :RU,FTN4X,&TAPE:13,1,%TAPTRN:13

CTSTAP T=00004 IS ON CR00013 USING 00002 BLKS R=0000

0001 **This transfer file compiles the TSTAPE program and the BCD and
0002 **EBCDIC subroutines. Then it links all together with the TAPTRN routine.
0003 :OF,TSTAE,0
0004 :RU,FTN4X,&BCD::13,1,%BCD:13
0005 :RU,FTN4X,&EBCDIC::13,1,%EBCDIC:13
0006 :RU,FTN4X,&TSTAPE,1,%TSTAPE:13
0007 :RU,LOADR,LTSTAPE

LTSTAP T=00004 IS ON CR00013 USING 00002 BLKS R=0000

0001 RE,%TSTAPE:13
0002 RE,%LOAD:13
0003 RE,%TAPTRN:13
0004 RE,%BCD:13
0005 RE,%EBCDIC:13
0006 EN

APPENDIX E
TRANSFER FILES TO COMPILE AND LOAD THE PROGRAM
**Document**: TAPTRN - A TAPE UTILITY PROGRAM

**Description**: The computer program TAPTRN, a utility routine used to read magnetic tapes of unknown content and format, is described for use on the HP 1000 computer. It provides the user ready access to all the capabilities of the tape handling hardware of the HP 1000. TAPTRN is documented in terms of a brief description of the utility, its capabilities, a guide on how to use it, and the source and pseudocode for the routine written.
<table>
<thead>
<tr>
<th>KEY WORDS</th>
</tr>
</thead>
<tbody>
<tr>
<td>SOFTWARE</td>
</tr>
<tr>
<td>MAGNETIC TAPE</td>
</tr>
<tr>
<td>FORTRAN</td>
</tr>
<tr>
<td>HP 1000 COMPUTER</td>
</tr>
<tr>
<td>FLECS</td>
</tr>
</tbody>
</table>

**INSTRUCTIONS**

1. ORIGINATING ACTIVITY: Enter the name and address of the organization issuing the document.

2a. DOCUMENT SECURITY CLASSIFICATION: Enter the overall security classification of the document including special warning terms whenever applicable.

2b. GROUP: Enter security reclassification group number. The three groups are defined in Appendix M of the DOD Security Regulations.

3. DOCUMENT TITLE: Enter the complete document title in all capital letters. Titles in all cases should be unclassified. If a sufficiently descriptive title cannot be selected without classification, show title classification with the usual one-capital-letter abbreviation in parentheses immediately following the title.

4. DESCRIPTIVE NOTES: Enter the category of document, e.g., technical report, technical note or technical letter. If appropriate, enter the type of document, e.g., interim, progress, summary, annual or final. Give the inclusive dates when a specific reporting period is covered.

5. AUTHOR(S): Enter the name(s) of author(s) as shown on or in the document. Enter last name, first name, middle initial. If military, show rank. The name of the principal author is an absolute minimum requirement.

6. DOCUMENT DATE: Enter the date (month, year) of Establishment approval for publication of the document.

7a. TOTAL NUMBER OF PAGES: The total page count should follow normal pagination procedures, i.e., enter the number of pages containing information.

7b. NUMBER OF REFERENCES: Enter the total number of references cited in the document.

8a. PROJECT OR GRANT NUMBER: If appropriate, enter the applicable research and development project or grant number under which the document was written.

8b. CONTRACT NUMBER: If appropriate, enter the applicable number under which the document was written.

8c. ORIGINATOR'S DOCUMENT NUMBER(S): Enter the official document number by which the document will be identified and controlled by the originating activity. This number must be unique to this document.

9a. OTHER DOCUMENT NUMBER(S): If the document has been assigned any other document numbers (either by the originator or by the sponsor), also enter this number(s).

10. DISTRIBUTION STATEMENT: Enter any limitations on further dissemination of the document, other than those imposed by security classification, using standard statements such as:

   (1) “Qualified requesters may obtain copies of this document from their defence documentation center.”

   (2) “Announcement and dissemination of this document is not authorized without prior approval from originating activity.”

11. SUPPLEMENTARY NOTES: Use for additional explanatory notes.

12. SPONSORING ACTIVITY: Enter the name of the departmental project office or laboratory sponsoring the research and development. Include address.

13. ABSTRACT: Enter an abstract giving a brief and factual summary of the document, even though it may also appear elsewhere in the body of the document itself. It is highly desirable that the abstract of classified documents be unclassified. Each paragraph of the abstract shall end with an indication of the security classification of the information in the paragraph (unless the document itself is unclassified) represented as (TS), (SI), (CI), (N), or (U).

   The length of the abstract should be limited to 20 single-spaced standard typewritten lines. 7.5 inches long.

14. KEY WORDS: Key words are technically meaningful terms or short phrases that characterize a document and could be helpful in cataloging the document. Key words should be selected so that no security classification is required. Identifiers, such as equipment model designation, trade name, military project code name, geographic location, may be used as key words but will be followed by an indication of technical context.
END

FILMED

4-85

DTIC