UTILIZATION OF MAGNETIC TAPES ON ND-560 COMPUTER

BY

R.H.KRIPALANI

AND

S.U.ATHALE

UTILISATION OF MAGNETIC TAPE FILES ON ND-560 COMPUTER SYSTEM

GENERAL INFORMATION ON ND-560 TAPE DRIVES:

(1) ND-560 has two STC (Storage Technology Corporation) tape drives. The STC model 1950 series tape units are tape transports for recording and retrieval of data on ANSI compatible nine track half-inch magnetic tapes. The model 1951 tape unit is a dual-density device capable of writing and reading 6250 bits per inch(bpi) in group coded recording (GCR) format and 1600 bpi in phase encoded(PE) format.

(2) PERFORMANCE SPECIFICATION:

TAPE SPEED		125 INCHES PER SEC
REWIND TIME(MAXIMUM) (2400 FOOT REEL)		60 SEC.
DATA DENSITY	GCR	6250 bpi
	PE	1600 bpi
DATA TRANSFER RATE	GCR	781 kilobyte/sec.
	PE	200 kilobyte/sec.
INTERBLOCK GAP (IBG)		
	GCR	0.3 inches
	PE	0.6 inches
ACCESS TIME (NOMINAL)		
WRITE	GCR	1.2 ms
	PE	1.2 ms
READ	GCR	1.4 ms
	PE	2.0 ms
TAPE START TIME		1.1 ms

(3) The two tape units are referred as M-T-0 and M-T-1. If one is facing the tape units, the left one is M-T-0 and the right one M-T-1.

(4) The logical device numbers(LDN) for the two tape units are as follows:

FILE	LDN(OCTAL)	LDN(DECIMAL)
M-T-0	40B	32
M-T-1	41B	33

NUMERIC AND ALPHANUMERIC CODES

(1) THE EXTENDED BINARY CODED DECIMAL INTERCHANGE CODE (EBCDIC)

EBCDIC employes 8 binary portions (one byte) to represent a single character of information. With this code 256 characters can be represented.

```
bit   bit   bit   bit !  bit   bit   bit   bit
 7     6     5     4  !   3     2     1     0
          -zone part-  !    ---numeric part---
```

EG., CHARACTER A WOULD BE CODED AS

```
1100  !  0001
----  !  ----
12 ZONE !  DIGIT 1
```

(2) THE AMERICAN STANDARD CODE FOR INFORMATION INTERCHANGE (ASCII)

This 7 bit code (4 bits for numeric and 3 bits for zone) developed by a committee of the American National Standards Institute (ANSI), has the advantage over most other codes of being contiguous, in the sense that the binary combinations used to represent alphanumeric information is sequential. Hence alphabetic sorting can be easily accomplished by arithmetic manipulation of the code values.

SINTRAN III uses bits 6-0 in a byte to represent ASCII characters. The 7th bit is the parity bit. It is set if there is an odd number of bits in bits 6-0 (even parity).

SOME ND FORTRAN FEATURES USEFUL FOR TAPE PROCESSING

(1) THE FORTRAN OPEN STATEMENT:

While using magnetic tapes as data files on EC-1040, NEC-1000 and CYBER computer systems certain control cards were needed:

```
EG., EC-1040    // ASSGN SYS006,182
```

```
NEC-1000      $ TAPE9 09,XID,,99999  ...
              $ DDEF  09,FFORM=IBM,RSZ=80 ...

CYBER         REQUEST,TAPE5,VSN= .. ..
              FILE,TAPE5,BT=E .....
```

On ND system the FORTRAN OPEN statement does all the above functions. The OPEN statement can connect an existing tape file to a unit. The OPEN statement has a list of several arguments. We will consider only those required for processing magnetic tape files.

(i) FILE = Character expression, 'M-T-0' OR 'M-T-1'

(ii) ACCESS='SPECIAL' For use of monitor calls MAGTP

(iii) FORM = 'FORMATTED' OR 'UNFORMATTED'

In formatted files data in a record is generally terminated by a CR LF (carriage return , line feed) pair of characters. Tapes created on ND system are treated as FORMATTED while tapes from other systems like VAX, CYBER,NEC are treated as UNFORMATTED.

(iv) Factor = Legal values 1,2 or 4 specifies the modification to the 'amount' factor in the monitor call MAGTP. MAGTP has as argument the length of the area read or written, where the amount is the exact number of bytes. The given amount parameters (or return parameters) in these monitor calls are adjusted by the value of factor before the monitor call is executed.

FACTOR=1 Indicates that the amount parameter is to be interpreted as number of bytes.(1 byte word)

FACTOR=2 Indicates the number of 16 bit words (2 byte word)

FACTOR=4 Indicates the number of 32 bit words (4 byte word)

ND-100 is used for I/O operations on tapes. Default for ND-100 is FACTOR=2, for ND-500 FACTOR=4.

(v) Unit = A means of referring to a file. If the word unit is omitted then the unit specifier must be first item LDN 1 implies terminal. 2-127 for open in FORTRAN.

(2) EQUIVALENCE STATEMENT

It is used to specify that storage is shared by two or more variables, arrays or character substring in ND-FORTRAN the restriction on the equivalencing character only with character is lifted. However an arithmatic or logical item may not begin on an odd byte boundary on ND-100 but the following is acceptable:

```
INTEGER K
```

```
CHARACTER *10 C
EQUIVALENCE (K,C(2:3))
```

Since C can start at an odd byte so that K will start at an even byte. However

```
INTEGER K,N
CHARACTER *10 C
EQUIVALENCE (K,C(1:2)),(N,C(2:3))
```

Is not allowed,since there is no way of avoiding one of either K or N starting at an odd byte. On ND-500 this situation produces an extension message not an error.

(3) CHARACTER ALIGNMENT ND-100
==========

Some monitor calls (eg MAGTP) in Sintran require that data areas begin on a word boundary. A character variable can be forced on to a word boundary by using an equivalence to an integer variable eg.,

```
CHARACTER *400 C
INTEGER *2 IC(200)
EQUIVALENCE (C,IC)
```

will force the variable C to be word aligned.

(4) In ND FORTRAN, records on a formatted external (not a print file) file are delimeted (terminated) by ASCII carriage return character (octal 15) line-feeds (octal 12) which immediately follow a carriage return are ignored on input hence ND FORTRAN for formatted files recl must be specified two character longer for cr,lf characters.

(5) CHARACTER SUBSTRING:
==========

Suppose you have a variable A, 1000 characters long i.e., CHARACTER*1000 A it is possible to extract character substrings from A as follows A(300:400) specifies characters in position 300 through 400 of the character variable A. With this it is possible to break long records and make them less than 256 characters each.

(6) ENDFILE STATEMENT
==========

This statement will write an endfile record. An ENDFILE statement will not automatically be performed before rewinding like on other systems.
An endfile record can also be put through the DEVICE-FUNCTION.

MONITOR CALL - MAGTP (NO.144B)
==========

(1) Monitor calls are used when a programming language does not provide a paricular function.

(2) Monitor call MAGTP reads from, writes to, or performs a variety of control functions for magnetic tape devices.It may also be used with other devices with similar characteristic to magnetic tape devices eg., versatec printer /plotter or floppy disk.

(3) The monitor call MAGTP is provided in the FORTRAN library.This may be called from a FORTRAN program as either a subroutine or a function subprogram. The main difference is that is using a monitor call as a function, a value is returned indicating the result of carrying out the request. If a function returns a status code, it is strongly recommended that this status is tested. If a monitor call is called as a subroutine, then status eg., error condition must be detected in a different way to a function. The system variable errorcode which is an ND FORTRAN extension, may be used with many of the monitor calls (both functions & subroutines) to detect errors. If errcode is used to detect errors from monitor calls the program must not be compiled in standard-check-on mode.

(4) MAGTP reuires five parameters. All the five parameters are of type INTEGER.

i. IFUNC ==> Function to be performed (eg.,READ,WRITE or REWIND)
Each function has got an access code eg., READ ==> 0,
WRITE ==> 1 etc.

ii. MARRY ==> Memory area (buffer) used for data transfer

iii. LDN ==> Logical device number

iv. IPAR1 !

v. IPAR2 ! Device dependent parameters.

(5) STATUS TESTING
==========

If MAGTP is used as a function eg., ISTAT=MAGTP(0,MA,40B,40,IDUM) then status is checked as follows:

```
IF(ISTAT.NE.0) CALL ERMSG(ISTAT)
```

If an error occurs, it will be displayed on the print file.If MAGTP is used as a subroutine eg.,call MAGTP(0,MA,40B,40,IDUM) then the error condition is tested as follows:

```
IF (ERRCODE.NE.0) CALL ERMSG(ERRCODE)
```

(6) CALL FORMATS :
==========

FORMAT (I) ISTAT=MAGTP(NF,ID,LN,ID,ID)

(II) ISTAT=MAGTP(NF,NA,LN,MW,MR)
(III) ISTAT=MAGTP(NF,NA,LN,NW,ID)
(IV) ISTAT=MAGTP(NF,ID,LN,ND,ID)

NF=Function number : ID=Dummy parameter=0 : LN=Logical device number
NA=Array name (buffer arrea) : MW=Maximum words/bytes (0/26B)
MR=Words/bytes read (0/26B) : MW=Words to be written : ND=Select density
ND=0 for 1600 BPI ; ND=1 for 6250 BPI

(I) In all the above formats ISTAT will receive error status on return. If ISTAT=0 the call terminated correctly. If ISTAT>0 it contains the FILE SYSTEM error number.
(Refer SINTRAN III Ref.Manual APPENDIX C error messages pages C5 TO C9)
EG., IF ISTAT=3 END OF FILE, ISTAT=165B BAD DATA BLOCK ETC

(II) Number terminated by a letter B means it is an octal number.

(III) If LN=40B for M-T-0 or 41B for M-T-1 then a FORTRAN open statement need not be there.

FUNCTION CODE (NF)	FORMAT NO.	FUNCTION
0	II	READ RECORD
1	III	WRITE RECORD
2	II	READ ODD NUMBER OF BYTES
4	II	READ ONE RECORD BACKWARDS
10B	I	ADVANCE THROUGH EOF. TAPE IS POSTIONED IMMEDIATELY AFTER EOF
11B	I	REVERSE THROUGH EOF. TAPE IS POSITIONED IMMEDIATELY IN FRONT OF EOF
12B	I	WRITE EOF
13B	I	REWIND
15B	I	BACKSPACE RECORD
16B	I	ADVANCE RECORD
17B	I	UNLOAD
23B	IV	SELECT DENSITY
26B	II	READ BYTE RECORD
27B	III	WRITE BYTE RECORD

EXAMPLE 1
=========

A tape contains monthly rainfall data for JANUARY thru DECEMBER 1871-1984

for 306 stations. For each station the first record contains the station name etc. followed by 114 records (1871-1984) of data. The format of the first record is (80A1)and for the remaining 114 records the format is (10X,12F5.2)
The tape was created on NEC system with recl=80 bytes blocking factor=1 and density=1600 bpi mode ASCII.
This program will read the data for the 11th station.

```
      DIMENSION R(12),STN(80)                 !R==>Rainfall for 12 months
                                              !STN==>Station name etc.
      INTEGER *2 IA(40)   !FOR                 IA = Temporary storage area(buffer)
      CHARACTER *80 A     !WORD                Where the contents of the records
      EQUIVALENCE (IA,A) !ALIGNMENT            Will be transfered

      OPEN(9,FILE='M-T-0',FORM='UNFORMATTED',FACTOR=2,ACCESS='SPECIAL')
      IDUM=0
      ISTAT=MAGTP(23B,IDUM,9,0,IDUM)          !0 means 1600bpi
      ISTAT=MAGTP(13B,IDUM,9,IDUM,IDUM)       !Rewind tape
      IF(ISTAT.NE.0)GO TO 20                  !If ISTAT=0 everything ok.

      DO 10 I=1,10
      DO 10 J=1,115                           !Since we want to skip the data for
                                              !first 10 stations
      ISTAT=MAGTP(16B,IDUM,9,IDUM,IDUM)       !Advance (16B) by 10*115 recs.
      IF(ISTAT.NE.0)GO TO 20
10    CONTINUE

      READ DATA FOR 11TH STATION
      ISTAT=MAGTP(0,IA,9,100,MWR)
```

This statement will read a record (code 0) and transfer the record to a buffer area IA or A. 100 denotes the maximum words to be read and MWR will give us the actual words read. If IA happens to be longer than 100,then system will read only 100 words.

```
      IF(ISTAT.NE.0)GO TO 20
      WRITE(1,*)ISTAT,MWR
      We can now read the first record from the buffer area A.
      READ(A,2)STN
2     FORMAT(80A1)
      WRITE(1,2)STN
      DO 30 I=1,114
      ISTAT=MAGTP(0,IA,9,100,MWR)
      IF(ISTAT.NE.0)GO TO 20
      READ(A,3)R
```

```
 3   FORMAT(10X,12F5.2)
     WRITE(1,3)R
30   CONTINUE
     GO TO 21
20   CALL ERMSG(ISTAT)
21   ISTAT=MAGTP(17B,IDUM,9,IDUM,IDUM)    !REWIND & UNLOAD TAPE
     STOP
     END
```

EXAMPLE 2
=========

A tape contains SST data with recl=80 bytes,but blocking factor=50 i.e., block contains 4000 characters. The first record of each block has format (4I20) and the remaining 49 have format (16F5.1),tape 1600 ASCII.

```
     DIMENSION SST(16)
     INTEGER*2 IA(2000)
     CHARACTER*4000 A
     CHARACTER*80 R(50)                  !50 Recs of length 80 characters each.
     EQUIVALENCE (A,IA)
     EQUIVALENCE (R,A)
     DO 10 IBLK=1,2
     ISTAT=MAGTP(26B,IA,40B,5000,MB) !Unit=M-T-0,MB=No. of bytes read,26B is
                                     !Read byte by byte
                                     !Since R & IA is equivalenced,
                                     !Array R can be directly used further

     READ(R(1),1)IM,IR,IX,IY         !Reading from first record
   1 FORMAT(4I20)
     DO 30 K=2,50                    !Reading from the 2nd till the 50 record
     READ(R(K),2)SST
   2 FORMAT(16F5.1)
  30 CONTINUE
  10 CONTINUE
     STOP
     END
```

EXAMPLE 3

A tape contains OLR data. Each record contains

```
IDATE , ILAT , (NOLR(I),I=1,144)
  I6     I4        144I4
```

i.e. (I6,145I4) Logical record length 586 characters

(Starting from 0 longitude there are 144 grid points at intervals of 2.5 Deg. Round the globe)

Data starts from 40n and goes to 40s at intervals of 2.5 deg.
(33 LATITUDES)

Block contains = 586 x 33 =19338 characters
Density=6250 bpi mode = EBCDIC

STEP 1. The maximum length of a formatted record is 256 characters in ND FORTRAN so first we will have to break not only the physical block, But also the logical records so that it can be read through FORTRAN. Each logical record (586) is brocken into 3 records as follows;

```
I6,I4,48I4,  48I4,  48I4
 1          2      3
202        192    192 CHARS.
```

THEREFORE ONE BLOCK WILL BE BROKEN INTO 33*3=99 RECS.

```
INTEGER *2 IA(9699)
CHARACTER *19338 A
EQUIVALENCE (IA,A)
OPEN (8,FILE='EB586',STATUS='UNKNOWN')
ISTAT=MAGTP(26B,IA,40B,20000,MB)
DO 10 I=1,33
I1=(I-1)*586+1
I2=I1+201
WRITE(8,1) A(I1:I2)
FORMAT(8X,A202)          !RECL=210 CHARACTERS
I1=I2+1
I2=I1+191
WRITE(8,2) A(I1:I2)
FORMAT(18X,A192)  !RECL=210
I1=I2+1
I2=I1+191
WRITE(8,2) A(I1:I2)
CONTINUE
```

In this way each physical block is brocken into 99 logical records of length 210 characters each. but the data is in EBCDIC so it has to converted to ASCII.

STEP 2. TO CONVERT EBCDIC DATA TO ASCII.

```
IITM>CONVERT
 SOURCE EBCDIC/ASCII/BCD ? (E/A/B):E
 :::::::
 .
 .
```

```
DESTINATION EBCDIC/ASCII/BCD ? (E/A/B):A
:::::::
.
.
INPUT IS MAG TAPE? (Y/N):N
INPUT FILE NAME:EB586
OUTPUT FILE NAME:AS586
RECORD LENGTH:212  (GIVE 2 CHARS. MORE)
OUTPUT FILE IS TEXT FILE?(Y/N):Y
STOP 0
IITM>
```

FILE AS586 will contain data in ASCII characters. If through PED you See EB586 it will contain some junk but through PED you can see AS586 file which contain actual data. This file 'AS586' can be directly read in your FORTRAN program.

If to the question INPUT IS MAG TAPE?(Y/N) answer is 'Y' then system will prompt with ANY HEADER?(Y/N): say 'N' rest will be the same.

STEP 3. It is now possible to transfer the entire contents of the file 'AS586' to a tape

```
IITM>COPY
  TO DEVICE:'M-T-1' OR M-T-0'
  FROM DEVICE:AS586:SYMB
IITM>
```

STEP 4. This tape generated on ND can directly be read in your FORTRAN program. It does not require the MAGTP function. eg.,

```
      OPEN(9,FILE='M-T-1',FORM='FORMATTED')
      READ (9,1) IDATE,ILAT,(NOLR(K),K=1,48)
1     FORMAT(8X,I6,49I4)
      READ(9,2) (NOLR(K),K=49,96)
      READ(9,2) (NOLR(K),K=97,144)
2     FORMAT(18X,48I4)
```

OPERATIONS ON MAGNETIC TAPES THROUGH DEVICE-FUNCTION

There are various functions on tape which can be used directly at the iitm prompt.
List of these functions can be seen by the command

```
*     IITM>LIST-DEVICE-FUNCTIONS
```

When the system responds with

IITM> Type DEVICE-FUNCTION press return key

The first response of the system will always be

FILE-NAME: TYPE M-T-0 OR M-T-1

The second response of the system will always be

FUNCTION:

The next response will depend on the function to be performed. For the following functions the system will always respond with

NO. OF OPERATIONS(DECIMAL):

ADVANCE-RECORDS
ADVANCE-TO-EOF
BACKSPACE-RECORDS
REVERSE-TO-EOF
WRITE-EOF

After the no. of operations are performed the system will respond with

IITM>

For the following functions

REWIND
UNLOAD

The system will perform the operation and respond with

IITM>

For the following functions

READ-BYTE-RECORD	(BYTES)
READ-RECORD	(WORDS)
READ-BACKWARDS	(WORDS)
READ-ODD-NO-OF-BYTES	(BYTES)

The system will respond with

BUFFER ADDRESS(OCT): TYPE 0
NO. OF BYTES/WORDS(OCT): 100000B (GIVE A LARGE NO)
NO. OF BYTES/WORDS READ: 4000B/2000B

To read the above record do the following

```
ITTM>DEV-FUN
FILE NAME:"TEST"
FUNCTION:WRITE-REC
BUFFER-ADDRESS(OCT):0
NO.OF WORDS(OCT):2000B
```

One can see the file TEST through PED .
(Here WRITE-BYTE-REC will give

ILLEGAL FUNCTION CODE

Since this function is meant to write on tapes only)

If the function is SELECT-DENSITY
DENSITY(BPI):1600

IF the function happens to be

READ-STATUS
READ-TAPE-STATUS
READ-LAST-STATUS

THE SYSTEM WILL RESPOND WITH

STATUS: (SOME OCTAL NUMBER)

EXAMPLE

Suppose while performing the function READ-REC the system responds with following message

DEVICE ERROR(DEV-FUN READ-LAST-STATUS TO GET STATUS)

or the value of ISTAT=171B (121D) while reading tape through MAGTP function,it means an error has occurred on the tape.To find the cause of the error do the following

```
ITTM>DEV-FUN
FILE NAME:M-T-0
FUNCTION:READ-LAST-STATUS
STATUS:100421B        ( This number is given by the system )
```

Each of the octal digit is to be broken into 3 binary digits

(Since to represent any octal no 3 bits are enough)

```
!-----------------------------------------------------------------!
!                                                                 !
!     1         0         0         4         2         1         !
!                                                                 !
!-----------------------------------------------------------------!
!                                                                 !
!     1     0   0   0     0   0   0     1   0   0     0   1   0     0   0   1   !
!                                                                 !
!-----------------------------------------------------------------!
! BIT POSITION
!    15    14  13  12    11  10  9    8   7   6    5   4   3    2   1   0   !
!-----------------------------------------------------------------!
```

From this we learn that bits in position 0,4,8 and 15 are on and the rest are off. Referring to appendix G1 of the SINTRAN reference manual, the error can be detected.

BIT	0	TAPE ON LINE
	1	WRITE ENABLE RING PRESENT
	2	TAPE STANDING ON LOAD POINT
	3	CRC ERROR/FATAL ERROR
	4	SET IF ANY OF BITS 5,6,7,8,9,11 OR 12 IS SET
	5	TRYING TO WRITE ON UNPROTECTED TAPE
	6	BAD DATA BLOCK
	7	END OF FILE DETECTED
	8	SEARCH CHARACTER DETECTED
	9	END OF TAPE DETECTED
	10	WORD COUNTER NOT ZERO
	11	DMA ERROR
	12	OVERFLOW
	13	TAPE BUSY OR FORMATTER BUSY
	14	LRC ERROR/SOFTWARE ERROR
	15	INTERRUPT WHEN FORMATTER IS READY

The above is different from error codes returned from monitor calls
(SINTRAN III FILE SYSTEM) APPENDIX C.2

EG

```
ISTAT=3     MEANS    END OF FILE
     =2     MEANS    BAD FILE NUMBER                    ETC
```

Suppose at any stage you wish to know the read/write errors your tape has encountered
Do the following

```
FUNCTION:READ-TAPE-STATUS
```

If there are no errors system responds

```
READ-ERROR STATUS:0B
NO.OF READ-ERRORS:0B
WRITE-ERROR STATUS:0B
NO. OF WRITE-ERRORS:0B
```

If there are errors a message like this may appear

```
READ-ERROR STATUS:12121B
NO. OF READ ERRORS:13B
WRITE-ERROR STATUS:0B
NO.OF WRITE-ERRORS:0B
```

For the function

```
FUNCTION:READ-STATUS
STATUS:5B

    1  0  1
    2  1  0    BIT
   ON OF ON
```

```
IITM>TAPE
(M A G N E T   VERSION 2.15)
```

Following are the commands

```
*ASSIGN N          ASSIGNS TAPE UNIT NO N
*BACKSPACE N       BACKSPACES BY N PHYSICAL RECORDS
*BLOCK N           SETS BLOCKLENGTH TO N
*DENSITY N         SETS DENSITY TO N
```

```
*DUMP N                 DUMPS CURRENT TAPEFILE TO TAPEUNIT N
*EXIT                   EXITS
*HELP                   TYPES  THIS TEXT
*MODE A                 SETS MODE TO ASCII OR EBCDIC
*PRINT                  PRINTS CURRENT TAPEFILE ON LINE-PRINTER
*READ  FILENAME         READS TAPEFILE TO DISC
*RECORD N               SETS RECL TO N  (MAX 136 CHARACTERS)  DEFAULT 80
*REWIND                 REWINDS TAPE
*SKIPFILE N             SKIPS N FILES FROM CURRENT TAPE POSITION
*SKIPRECORD N           SKIPS N PHYSICAL RECS FROM CURRENT POSITION
*WRITE  FILENAME        FROM DISC TO TAPE
*UNLOAD                 UNLOADS TAPE
*VERIFY N               DISPLAYS N LINES ON TERMINAL FROM CURRENT
                        PHYSICAL RECORD
*EXIT
```

In ND-FORTRAN new version, it is possible to WRITE/READ a logical record size upto 60,000 bytes long by using FORTRAN READ/WRITE statements. If your record size exceeds 255 bytes, additional commands SET-IO-BUFFERS (1 buffer for each file) and LOAD FORTRAN-LIBRARY have to used in Linkage-Loader. Following is an example of writing record size of 40,000 bytes through write statement.

```
@DEL-FI YYY:NRF
@F-5
ND-500 ANSI 77 FORTRAN COMPILER - 203054J03
FTN: COMP TEST-SIZE,1,"YYY"

ND-500 ANSI 77 FORTRAN COMPILER - 203054J03      10:42  19 AUG 1991
SOURCE FILE:   TEST-SIZE:SYMB

     1*            PROGRAM PROG1
     2*      C PROGRAM TO TEST RECORD SIZE OF 40000 BYTES
     3*      C PROGRAM GENERATES DATA IN ARRAY A AND WRITES IT ON TAPE
     4*            DIMENSION A(8000)
     5*            OPEN(6,FILE='M-T-1',FORM='FORMATTED')
     6*            REWIND(6)
     7*            DO 10 I=1,8000
     8*         10 A(I)=I
     9*            WRITE(6,'(5000F8.2)')A
    10*            A=A*2
    11*            WRITE(6,'(5000F8.2)')A
    12*            ENDFILE(6)
```

```
13*            STOP
14*            END

- CPU TIME USED: 0.4 SECONDS.  13 LINES COMPILED.
- NO MESSAGES
- PROGRAM SIZE=228 DATA SIZE=32196 COMMON SIZE=0
FTN: EX

@L-L

ND-Linkage-Loader -  H.00        3. April       1986 Time:  0:00
N11 entered:                    19. August      1991 Time: 10:42
N11: DEL-DOM YYY
N11: SET-IO-BUFFER 1
N11: SET-DOM "YYY"
N11: LOAD YYY,FORT-LIB
Program:.........350 P01   Data:..........76710 D01
FORTRAN-LIB-203101-J02
FORTRAN-LIB-203101-J02
FORTRAN-LIB-203101-J02
Program:.......66455 P01   Data:.........315104 D01
N11: EX
IO-buffer:....320000 D01   ..............323777 D01
Segment no........30        is linked

@ND YYY

STOP 0
```

The data is written this way gives logical record lengh as in the program(40000 bytes) but the physical record length(which actualy on the tape) is 2048 bytes.

Following program reads the data written on tape using the above program and writes the same in output-file TEST1.

```
@DEL-FI YYY:NRF
@F-5
ND-500 ANSI 77 FORTRAN COMPILER - 203054J03
FTN: COMP READ-SIZE,1,"YYY"

ND-500 ANSI 77 FORTRAN COMPILER - 203054J03      12:08  19 AUG 1991
SOURCE FILE:    READ-SIZE:SYMB

1*             PROGRAM PROG1
```

```
  2*            DIMENSION A(8000)
  3*            OPEN(6,FILE='M-T-1',FORM='FORMATTED')
  4*            OPEN(7,FILE='TEST1',FORM='FORMATTED',STATUS='UNKNOWN')
  5*            REWIND(6)
  6*            READ(6,'(5000F8.2)',END=40)A
  7*            WRITE(7,'(5000F8.2)')A
  8*            READ(6,'(5000F8.2)',END=40)A
  9*            WRITE(7,'(5000F8.2)')A
 10*         40 STOP
 11*            END

- CPU TIME USED: 0.3 SECONDS.  11 LINES COMPILED.
- NO MESSAGES
- PROGRAM SIZE=351 DATA SIZE=32280 COMMON SIZE=0
FTN: EX

@L-L

ND-Linkage-Loader -  H.00        3. April       1986 Time:  0:00
N11 entered:                    19. August      1991 Time: 12: 8
N11: DEL-DOM YYY
N11: SET-IO-BUFFER 2
N11: SET-DOM "YYY"
N11: LOAD YYY,FORT-LIB
Program:.........543 P01   Data:..........77034 D01
FORTRAN-LIB-203101-J02
FORTRAN-LIB-203101-J02
Program:.......66200 P01   Data:.........315524 D01
N11: EX
IO-buffer:....320000 D01   ..............327777 D01
Segment no........30       is linked

@ND

STOP 0
```

The output file TEST1 can be seen in PED. In PED if record length is exceeding 255,it will read the file and say 'LINES EXCEEDED MAXIMUM LINE LENGTH' and a character $ will come at the beginning of each line. But this character is actualy not there in the file.It is just the indication of record length is greater than 255, since the limitation for a line in PED is 255. Records will continue on the next lines. The exact record length can be calculated by multipling the lines with $ by 255.