UNITED STATES DEPARTMENT OF THE INTERIOR

GEOLOGICAL SURVEY

Programmer's Guide to REMAPP,

REMote sensing Array Processing Procedures

by

D. L. Sawatzky

Open-file Report 85-231

1985

This document is sufficient in its existing form to be consulted by REMAPP image processing software users on the PERKIN-ELMER 3220 minicomputer with the OS32MT VERSION 6.2 operating system and Multiterminal Monitor for most commonly used and straightforward procedures. It is not intended to be an introduction to the operating system or scientific use of computers. The REMAPP user is expected to have experience in scientific use of computers and in the OS32MT operating system.

Table of Contents

Acknowledgements

A number of ideas instrumental in the development of REMAPP came from the U. S. Geological Survey image processing facility in Flagstaff, Azizona. The subroutines DISKIO and TAPEIO were based on their subroutines.

INTRODUCTION

REMAPP (REMote sensing Array Processing Procedures) is a set of applications programs and utility subroutines designed to facilitate the processing of remotely sensed data in digital form. REMAPP is currently implemented on a PERKIN-ELMER 3220 minicomputer in the Remote Sensing Laboratory of the U. S. Geological Survey, Branch of Geophysics, in Denver, Colorado. It has been transported to the DEC VAX 11/780 in Reston, Virginia and the VAX 11/780 and 11/750 in Denver, Colorado. This manual does not address the peculiarities of the VAX operating system, but can be used to some advantage at the REMAPP applications level on those systems.

REMAPP allows processing of image files from computer-compatible magnetic tapes of different formats, such as Landsat and Heat Capacity Mapping Mission, as well as any geophysical, geological or geochemical data that can be converted and handled as image data.

Data formats on tapes are first converted to a single general format, and then stored in REMAPP format disk files. Existing applications programs provide several processing capabilities: arithmetic operations on and between image files and channels or bands, reading Landsat data tapes, and reading and writing Optronics format data tapes.

This guide introduces the subroutines DISKIO and TAPEIO that are called by applications programs written in Fortran. These subroutines efficiently perform input and output to disk files and tapes, and allow the taking of subsets of image files in a standard way for all REMAPP programs.

This guide is intended for users who are interested in writing image processing programs in Fortran that access the REMAPP data base. For more explanation of REMAPP, users should consult the REMAPP user's manual.

Fortran Programming Practices

Logical Units

Some logical units in Fortran programs doing REMAPP processing have special assignments:

4 - REMAPP log
5 - user console or command file for input
6 - user console or batch log for output
7 - magnetic tape device
10 - array processor
11 - array processor

Calling Diskio and Tapeio

The following programming example indicates how to use REMAPP subroutines.

```
                             INTEGER FCBIN(256),FCBOUT(256)
                             LOGICAL*1 BUFFER(4096)
                             .
                             .
    C........OPEN INPUT DISK FILE
                             FCBIN(0)=0
                             CALL DISKIO(9,12,BUFFER,FCBIN)
                             IF(FCBIN(1).LT.0) CALL EXIT(2)
                             .
                             .
    C........OPEN OUTPUT DISK FILE
                             FCBOUT(1)=8
                             FCBOUT(2)=2000
                             CALL DISKIO(0,13,BUFFER,FCBOUT)
                             IF(FCBOUT(1).LT.0) CALL EXIT(3)
                             .
                             .
    C........OPEN INPUT TAPE FILE
                             N=0
                             CALL TAPEIO(0,0,BUFFER,N)
                             IF(N.LT.0) CALL EXIT(2)
                             .
                             .
    C........OPEN OUTPUT TAPE FILE
                             N=0
                             CALL TAPEIO(0,1,BUFFER,N)
                             IF(N.LT.0) CALL EXIT(3)
                             .
                             .
    C........READ A REMAPP FILE RECORD
                             CALL DISKIO(9,12,BUFFER,FCBIN)
                             IF(FCBIN(1).LT.0) THEN
                             IF(FCBIN(1).EQ.-1) GOTO 100
                             GOTO 200
                             ENDIF
                             .
                             .
    C........WRITE A REMAPP FILE RECORD
                             CALL DISKIO(10,13,BUFFER,FCBOUT)
                             IF(FCBOUT(1).LT.0) GOTO 300
                             .
                             .
    C........READ A TAPE RECORD
                             N=4096
                             CALL TAPEIO(9,0,BUFFER,N)
                             IF(N.LT.0) THEN
                             IF(N.EQ.-1) GOTO 100
                             GOTO 200
                             ENDIF
                             .
                             .
```

```
C........CLOSE REMAPP FILES
200                     CALL DISKIO(6,12,BUFFER,FCBIN)
                        .
                        .
                        CALL DISKIO(6,13,BUFFER,FCBOUT)
                        .
                        .
                        CALL TAPEIO(6,0,BUFFER,N)
                        .
                        .
                        CALL TAPEIO(6,1,BUFFER,N)
                        .
                        .
                        STOP
```

SUBROUTINE DISKIO

DISKIO provides a general facility to read and write REMAPP-format disk files on the Perkin-Elmer 3220 minicomputer under operating system OS32MT. This subroutine has the added features of being able to read or create disk files compatible with the Earth Resources Laboratory Applications Software (ELAS) installed on Perkin-Elmer computers.

User programs can read data records sequentially or randomly, as well as skip records forward or backward. The user may specify the first record to be read (the starting scanline in the case of an image file) and the first element of that record (the starting picture element or pixel) as well as the number of lines and pixels.

To access any REMAPP or ELAS disk file the user need only supply the standard OS32 file descriptor. Volume name must be supplied with the filename, but the extension and file class are optional. The file class is defaulted to the private file class signed on under. If the file is an input file and does not already exist, an error message is given and the task calling DISKIO is stopped or otherwise handles the error. If the file is an output file, the filename must not already exist or an error message is given. If the output filename does not exist, the file is created on the given volume and under the given or default file class.

Data records (one scanline per record) are written or read from the file without conversion, packing, or unpacking by DISKIO. The calling program must do packing and unpacking of data, which is advised to make efficient use of disk storage. Special hardware such as array processors can be utilized to do packing and unpacking of data as well as arithmetic processing.

When a program calling DISKIO opens a file for output, the precision in bits per byte of the array elements values (pixel data number) must be specified. When an input file is opened, the data precision is obtained from the REMAPP file header. The precision can be used by the packing and unpacking subroutines.

The calling program must call DISKIO to close the disk file. In the case of an output file, the file header is updated with the total lines written before closing as well as the other information entered at its opening.

Output files are always created with the read-only protection code. This is to prevent inadvertent overwriting or deleting of an existing file. The file must be reprotected to zero in order to delete or rewrite it.

The general form of a Fortran call to DISKIO is as follows:

```
CALL DISKIO(OPERATION,UNIT,IOAREA,FCB)
```

where:

OPERATION is an integer scalar specifying the I/O function to be performed.

UNIT is an integer scalar specifying the Fortran logical unit number to which the file is assigned.

IOAREA is an integer or logical array to read file records into or write file records from.

FCB is a 256-word integer array into which the file header information is read upon opening an input file or from which it is written upon closing an output file. User responses to DISKIO prompts for header information are stored here by the program

Operations

The following operations are valid:

0 = open a file
3 = rewind file, move to first data record
4 = skip forward no. of records specified in FCB(1)
5 = backspace no. of records specified in FCB(1)
6 = close a file
9 = read record into array IOAREA
10 = write record from array IOAREA
11 = read record whose no. is specified in FCB(1)

OPERATION = 0; FCB(1)=0

FCB(1)=0 to open an existing REMAPP or ELAS file for input

The user is prompted to enter the name of the file to be opened for input. The file descriptor must contain the volume name, the file name, and optional extension and optional file class. If FCB(17)=0, the following prompt is given:

```
DISKIO: ENTER INPUT FILE NAME
```

If FCB(17) $\neq$ 0, then the character string in array IOAREA is given.

If the volume name is not included in the response, this message is given:

```
VOLUME NOT SPECIFIED
```

and DISKIO returns with error no. -2 in FCB(1).

If the file named cannot be opened, this message is given:

```
DISKIO: OPEN ERROR = nn
```

where nn is

1	illegal function
2	illegal logical unit number
3	no such volume
4	no such filename on given volume
5	no room on volume for data or index block
6	protection error
7	privilege error
8	not enough system space for buffer
9	logical unit assignment error
11	file descriptor syntax error
128-255	other I/O error detected by OS32

and DISKIO returns with error no. -2 in FCB(1).

If the file opened is an ELAS file, this message is given:

```
ELAS FILE: IL=nnnn, LL=nnnn
           IE=nnnn, LE=nnnn
        NCHAN=nn
   BITS/PIXEL=nn
```

If bits/pixel =0 or $\neq$ 8,16,32, information is requested:

```
ENTER ELAS FILE BITS/PIXEL!
```

If opened file is a REMAPP file, the following message is given:

```
DISKIO: LINES nnnnn PIXELS nnnnn BYTESIZE nn
```

Then REMAPP or ELAS file subset is requested:

```
ENTER 1ST SCANLINE, NO. SCANLINES, SKIPS:

ENTER 1ST PIXEL, NO. PIXELS, SKIPS:
```

Following these requests, this message is given:

```
INPUT FILE filename OPENED:

WINDOW      1ST    NO.   SKIP  ACTUAL
SCANLINES  nnnnn  nnnnn   nn   nnnnn
PIXELS     nnnnn  nnnnn   nn   nnnnn
```

OPERATION=0; FCB(1)≠0

FCB(1)≠0 to create REMAPP or ELAS file for output. The value of FCB(1) is the no. of bits/pixel = 8,16, or 32 for the new file. The calling program program must also set the no. of pixels per scanline in FCB(2). If the new file is an ELAS file then FCB(3) must contain the no. of scanlines per channel and FCB(13) must contain the no. channels (bands).

The user is prompted to enter the name of the file to be created for output. The file descriptor must contain the volume name, the file name, and optional extension and optional file class. To create an ELAS file append the % character to the file descriptor. If FCB(17)=0, then the following prompt is given:

```
DISKIO: ENTER OUTPUT FILE NAME
```

If FCB(17) ≠ 0, then the character string in array IOAREA is given.

If the volume name is not included in the response, this message is given:

```
VOLUME NOT SPECIFIED
```

and DISKIO returns with error no. -2 in FCB(1).

If the file named cannot be created this message is given:

```
DISKIO: CREATE ERROR = nn
```

where nn is

1 =	illegal function or illegal file type was specified
3 =	no such volume name
4 =	filename already exists on volume
5 =	insufficient space to allocate file on volume
10 =	the volume is not a direct access device
11 =	file descriptor syntax error

and DISKIO returns with error no. -2 in FCB(1).

If the new file cannot be opened this message is given:

```
DISKIO: OPEN ERROR = nn (see OPEN error nos. above)
```

and DISKIO returns with error no. -2 in FCB(1).

OPERATION=3

Operation=3 causes rewinding of the file or moving to the first data record. See OPERATION=9 for error messages.

OPERATION=4

Operation=4 causes the no. of data records specified in FCB(1) to be skipped. See OPERATION=9 for error messages.

OPERATION=5

Operation=5 causes the no. of data records specified in FCB(1) to be backspaced. See OPERATION=9 for error messages.

OPERATION=6

Operation=6 closes the disk file. If the file is a REMAPP output file, the file control block, FCB, which now contains the no. of data records written, is written to the file. The following message is given for output files:

```
OUTPUT FILE CLOSED:
DISKIO: LINES nnnnn PIXELS nnnnn BYTESIZE nn
```

No message is given after the closing of an input file. See OPERATION=9 for error messages.

OPERATION=9

Operation=9 causes the next data record to be read from an input file into array IOAREA. The no. of pixels transferred is the no. in FCB(2) of the file header at file opening. If an end-of-file is read, this message is given:

```
DISKIO: END OF FILE ON filename
```

and DISKIO returns with error no. -1 in FCB(1). If an attempt to read a record before the beginning of the file is made, DISKIO returns with error no. -2 in FCB(1). The first pixel in the array will be the first requested from the data record. Skipped pixels will be removed from the array. On I/O error a diagnostic message will be given for the Fortran VII Run Time Library:

```
ERR n (mmmmmm) LU # NN:
message
```

where:

NN = Fortran logical unit no.

n=-1	HARDWARE EOF DETECTED
n=26	PARITY OR RECOVERABLE ERROR
n=28	END OF MEDIUM

DISKIO returns with the error no. -2 in FCB(1).

OPERATION=10

Operation=10 causes the contents of array IOAREA to be written to the next data record in an output file. The no. of pixels transferred is the no. set in FCB(2) at file creation. On I/O error a diagnostic message will be given for the Fortran VII Run Time Library:

```
ERR n (mmmmmm) LU # NN:
message
```

where:

NN = Fortran logical unit no.

```
n=-1  HARDWARE EOF DETECTED
n=26  PARITY OR RECOVERABLE ERROR
n=28  END OF MEDIUM
```

DISKIO returns with error no. -2 in FCB(1).

OPERATION=11

Operation=11 causes a specified data record to be read into array IOAREA as in Operation=9. The record no. is specified in FCB(1). I/O errors and errors in record no. are handled as in Operation=9.

Unit

UNIT is the logical unit to which the REMAPP disk file is assigned by DISKIO during the opening. All application programs should use logical unit nos. 12 and greater for REMAPP disk files. Logical unit 4 is used for the REMAPP log, unit 5 is for user replies or command files, unit 6 is for program interactive prompts to users, unit 7 is used by subroutine TAPEIO for the tape drive, and units 10 and 11 are used by the array processor.

Ioarea

IOAREA is the array into which data records are read or assembled for output. The type declaration of IOAREA array must be consistent with the use of the data in the calling program. If the data is 16-bit integer data, then IOAREA is declared type INTEGER*2 and packing and unpacking is unnecessary. Likewise, for 32-bit integer data, IOAREA is declared type INTEGER*4, and no packing and unpacking is needed. If the data is unsigned 8-bit integer data,it must be packed or unpacked in the array by the calling program before I/O to the disk file. If a REAL array is used in the program, a output file must be opened for 32-bit data. IOAREA must be dimensioned large enough to contain all the bytes in a single data record of the file.

REMAPP File Control Block

File control block (FCB) format is as follows:

```
FCB(1) - ON OPER=0, 0 = OPEN INPUT FILE
                   => 8 = OPEN OUTPUT FILE WITH BITS/PIXEL
         ON OPER=4,5 HAS NO. OF RECORDS TO SPACE
         ON OPER=11, NO. OF RECORD TO READ
         ON RETURN FROM DISKIO HAS STATUS:
                  STATUS=0 - NO ERROR
                  STATUS=-1 - END OF FILE OR MEDIA
                  STATUS=-2 - I/O ERROR
FCB(2) - ON OPEN FOR OUTPUT FILE HAS NO. OF PIXELS PER SCANLINE
         ON OPEN FOR INPUT FILE RETURNS PIXELS/SCANLINE WINDOW
FCB(3) - ON OPEN FOR OUTPUT FILE HAS NO. OF SCANLINES PER
         CHANNEL. NECESSARY IF OUTPUT FILE IS ELAS TYPE. ON OPEN INPUT
         RETURNS NO. OF SCANLINES IN WINDOW.
```

FCB(4) - FOR INPUT FILE CONTAINS FIRST PIXEL TO READ
FCB(5) - FOR INPUT FILE CONTAINS FIRST SCANLINE TO READ
FCB(6) - FOR INPUT FILE CONTAINS NO. OF PIXELS TO SKIP
FCB(7) - FOR INPUT FILE CONTAINS NO. OF SCANLINES TO SKIP
FCB(8)-FCB(11) - FILENAME IN ASCII
FCB(12) - RECORD LENGTH IN BYTES
FCB(13) - NO. OF CHANNELS
FCB(14) - CURRENT RECORD NO.
FCB(15) - NO. SCANLINES IN WINDOW
FCB(16) - NO. PIXELS PER SCANLINE IN WINDOW
FCB(17) - =0 USE STANDARD OPEN PROMPT
≠0 USE PROMPT STRING IN IOAREA
FCB(18) - FIRST PHYSICAL RECORD NO.
FCB(19) - PHYSICAL RECORD INCREMENT PER SCANLINE
FCB(21) - 0 FOR INPUT, 1 FOR OUTPUT
FCB(23) - NO. OF PIXELS/SCANLINE
FCB(24) - NO. OF RECORDS/FILE
FCB(25) - BITS/PIXEL
FCB(26) - LAST SCANLINE IN WINDOW
FCB(27) - LAST PIXEL IN WINDOW
FCB(30)-FCB(256) - FILE HISTORY IN ASCII

ELAS Header

The ELAS file header is stored in FCB array as follows:

FCB(1) - NO. OF BYTES IN HEADER
FCB(2) - NO. BYTES/DATA RECORD
FCB(3) - INITIAL LINE OF DATA
FCB(4) - LAST LINE
FCB(5) - INITIAL ELEMENT OF DATA LINE
FCB(6) - LAST ELEMENT
FCB(7) - NO. OF CHANNELS
FCB(8) - 4321
FCB(19) - BITS/PIXEL, IF=0 THEN =8
FCB(57)-FCB(60) - ASCII FILE NAME

Subroutines Called

The following Fortran Run Time Library subroutines are called by DISKIO: CFILW, OPENW, SYSIO, IOERR, CLOSE.

DISKIO Fortran Listing

```
      SUBROUTINE DISKIO(OPER,UNIT,IOAREA,FCB)
$INCLUDE REMAPP.HDR
C    ***************************************************************
C
C    REMAPP P-E/3220 1985
C    REMote sensing Array Processing Procedures
C    U. S. GEOLOGICAL SURVEY, DENVER, COLORADO
C    BRANCH OF GEOPHYSICS, REMOTE SENSING LABORATORY
C    DON L. SAWATZKY
C
```

```
C   ***************************************************************
C..................
C
C     DISKIO WILL CREATE OR READ REMAPP AND ELAS TYPE DISK FILES
C     REMAPP DISK FILES ARE INDEXED FILES UNDER PERKIN-ELMER OS32
C     ELAS DISK FILES ARE CONTIGUOUS FILES UNDER OS32
C
C..........
C     OPERATIONS
C
C     OPER=0 - OPEN A FILE
C          3 - REWIND, POINT TO FIRST DATA RECORD
C          4 - SKIP NO. RECORDS SPECIFIED IN FCB(1)
C          5 - BACKSPACE NO. RECORDS SPECIFIED IN FCB(1)
C          6 - CLOSE A FILE
C          9 - READ RECORD INTO ARRAY IOAREA
C          10 - WRITE RECORD FROM ARRAY IOAREA
C          11 - READ RECORD NO. SPECIFIED IN FCB(1) INTO ARRAY
C               IOAREA
C
C...................
C
C     UNIT = INTEGER FORTRAN LOGICAL UNIT NO. ASSIGNED TO FILE
C
C...................
C
C     IOAREA = INTEGER OR LOGICAL ARRAY DIMENSIONED TO CONTAIN
C            ENOUGH BYTES TO HOLD ONE FILE RECORD
C
C............................
C     REMAPP FILE CONTROL BLOCK DEFINITIONS
C
C     FCB = INTEGER ARRAY DIMENSIONED TO 256 (1024 BYTES)

C     WORD 1 - ON OPER=0, 0= OPEN INPUT FILE
C                        => 8 OPEN OUTPUT FILE WITH BITS/PIXEL
C              ON OPER=4,5 HAS NO. OF RECORDS TO SPACE
C              ON OPER=11, NO. OF RECORD TO READ
C              UPON RETURN HAS STATUS
C                    STATUS=0 - NO ERROR
C                    STATUS=-1 - END OF FILE OR MEDIA
C                    STATUS=-2 - I/O ERROR
C     WORD 2 - ON OPEN FOR OUTPUT FILE HAS NO. OF PIXELS/SCANLINE
C              ON OPEN INPUT FILE RETURNS PIXELS/SCANLINE WINDOW
C     WORD 3 - ON OPEN OUTPUT FILE HAS NO. OF SCANLINES PER
C              CHANNEL
C              NECESSARY IF OUTPUT FILE IS ELAS TYPE.
C              ON OPEN INPUT RETURNS NO. OF SCANLINES WINDOW.
C     WORD 4 - FOR INPUT FILE CONTAINS FIRST PIXEL TO READ
C     WORD 5 - FOR INPUT FILE CONTAINS FIRST SCANLINE TO READ
C     WORD 6 - FOR INPUT FILE CONTAINS PIXEL SKIP FACTOR
C     WORD 7 - FOR INPUT FILE CONTAINS SCANLINE SKIP FACTOR
C     WORDS 8,11 - FILENAME IN ASCII
C     WORD 12 - RECORD LENGTH IN BYTES
```

```
C     WORD 13 - NO. OF CHANNELS
C     WORD 14 - CURRENT RECORD NO.
C     WORD 15 - NO. SCANLINES IN WINDOW
C     WORD 16 - NO. PIXELS PER SCANLINE IN WINDOW
C     WORD 17 - =0 USE STANDARD OPEN PROMPT
C               <>0 USE PROMPT STRING IN IOAREA
C     WORD 18 - RECORD NO. OF FIRST SCANLINE
C     WORD 19 - RECORD INCREMENT
C     WORD 21 - 0 FOR INPUT, 1 FOR OUTPUT
C     WORD 23 - NO. OF PIXELS/SCANLINE
C     WORD 24 - NO. OF RECORDS/FILE
C     WORD 25 - BITS/PIXEL
C     WORD 26 - NO. OF LAST SCANLINE IN WINDOW
C     WORD 27 - NO. OF LAST PIXEL IN WINDOW
C     WORD 30-256 - FILE HISTORY IN ASCII
C.......................................................
C     TO OPEN A REMAPP OUTPUT FILE MUST PASS
C     FCB 1 AND 2
C
C     TO OPEN AN ELAS OUTPUT FILE MUST PASS
C     FCB 1,2,3,13
C
C     AND DIMENSION AND INITIALIZE ALL FCBS:
C     INTEGER*4 FCB(256)/256*0/
C..........................................................
C     ELAS FILE HEADER
C
C     WORD 1 - NO. BYTES IN HEADER
C     WORD 2 - NO. BYTES/DATA RECORD
C     WORD 3 - INITIAL LINE OF DATA
C     WORD 4 - LAST LINE
C     WORD 5 - INITIAL ELEMENT OF DATA LINE
C     WORD 6 - LAST ELEMENT
C     WORD 7 - NO. OF CHANNELS
C     WORD 8 - =4321
C     WORD 19 - BITS/PIXEL, IF=0 THEN =8
C     WORDS 57,60 - ASCII FILE NAME
C.......................................................
      LOGICAL*1 IOAREA(1)
      INTEGER*4 FCB(256),OPER,UNIT,TTYIN,TTYOUT,ISIZE,NFILE(5)
      CHARACTER FILEDESC*20,FILENM*16,STRING*64
      INTEGER*4 CODE,PBLK(6),ELAS(256)
      COMMON/DISCOM/ CODE,FILENM,I,IFC,ISIZE,ISTAT,J,JUMP,
     & K,KEND,KINC,KMUL,KST,L,LL,NBYTES,NFILE,PBLK,TTYIN,TTYOUT
     & ,FILEDESC,STRING,ELAS,M,N,IFSIZE,IBSIZE
C
C  ENTRY POINT AND BRANCH VECTOR TO OPERATION ROUTINES
C
      JUMP=OPER+1
      IF(JUMP.GT.14.OR.JUMP.LT.1) GOTO 5
      GO TO (30,5,5,100,110,120,200,5,5,140,190,130,5,5)JUMP
5     WRITE(6,10)
10    FORMAT(1X,'DISKIO:INVALID OPERATION CODE')
      STOP
```

```
C
C  RETURN
20     RETURN
C
C  FILE OPENING PROCEDURES
C
30     IF(FCB(17).EQ.0) THEN
        IF(FCB(1).EQ.0)WRITE(6,11)
11      FORMAT(1X,'DISKIO:ENTER INPUT FILENAME')
        IF(FCB(1).GT.0)WRITE(6,12)
12      FORMAT(1X,'DISKIO:ENTER OUTPUT FILENAME')
       ELSE
       WRITE(STRING,'(64A1)') (IOAREA(I),I=1,64)
       WRITE (6,*) 'DISKIO:'//STRING(1:INDEX(STRING,'$')-1)
       FCB(17)=0
       ENDIF
        READ(5,'(A20)')FILEDESC
        IF(INDEX(FILEDESC,':').EQ.0) THEN
        WRITE(6,14)
14      FORMAT(1X,'VOLUME NOT SPECIFIED')
       GO TO 300
       ENDIF
C.....GET FILENAME FROM FILEDESC
       IF(FCB(1).NE.0) THEN
       N=INDEX(FILEDESC,':')
       M=INDEX(FILEDESC,'/')
       L=INDEX(FILEDESC,'%')
C.....IF OUTPUT FILE DESCRIPTOR CONTAINS % CREATE AN ELAS FILE
       IF(L.NE.0) THEN
       FCB(15)=4321
C.....GET ELAS FILE DESCRLIPTOR
       FILEDESC=FILEDESC(1:L-1)
       ENDIF
C.....REMOVE VOLUME NAME FROM FILE DESCRIPTOR TO GET FILENAME
       IF(M.EQ.0) THEN
       FILENM=FILEDESC(N+1:)
       ELSE
C.....REMOVE FILE CLASS FROM FILENAME, IF ANY
       FILENM=FILEDESC(N+1:M-1)
       ENDIF
C.....SET DEFAULT NO. OF CHANNELS TO 1
       IF(FCB(13).EQ.0) FCB(13)=1
C.....MOVE FILENAME TO FCB
       READ(FILENM,'(4A4)') (FCB(I),I=8,11)
       ENDIF
       IF(FCB(1).NE.0)GO TO 80
C
C  OPEN INPUT FILE
C
       CALL OPENW(UNIT,FILEDESC,0,0,255,CODE)
       IF(CODE.NE.0) WRITE(6,399) CODE
399    FORMAT(/1X,'DISKIO: OPEN ERROR = ',I3)
       IF(CODE.NE.0)GO TO 300
       IFC=77;READ,RANDOM,IMAGE,BINARY,WAIT
```

```
      CALL SYSIO(PBLK,IFC,UNIT,FCB,1024,0)
      CALL IOERR(PBLK,ISTAT)
      IF(ISTAT.NE.0) GOTO 300
C.....CHECK IF REMAPP OR ELAS FILE
      IF(FCB(8).EQ.4321) THEN
C.....ELAS FILE
      WRITE(6,*) 'ELAS FILE: IL=',FCB(3),' LL=',FCB(4)
      WRITE(6,*) '           IE=',FCB(5),' LE=',FCB(6)
      WRITE(6,*) '        NCHAN=',FCB(7)
      WRITE(6,*) '   BITS/PIXEL=',FCB(19)
      IF(FCB(19).EQ.0.OR.(FCB(19).NE.8.AND.
     & FCB(19).NE.16.AND.FCB(19).NE.32)) THEN
      WRITE(6,*) 'ENTER ELAS FILE BITS/PIXEL!'
      READ(5,*) FCB(19)
      ENDIF
C.....MOVE ELAS STUFF UP
      DO 400 I=1,8
      FCB(60+I)=FCB(I)
400   FCB(I)=0
C.....LOAD FCB TO LOOK LIKE REMAPP FCB
      FCB(1)=0
      FCB(8)=FCB(57)
      FCB(9)=FCB(58)
      FCB(10)=FCB(59)
      FCB(11)=FCB(60)
      FCB(12)=FCB(62)
      FCB(23)=FCB(66)-FCB(65)+1
      FCB(24)=FCB(67)*(FCB(64)-FCB(63)+1)
      FCB(25)=FCB(19)
      FCB(13)=FCB(67)
C.....GET RANDOM READ PARMS
C     FCB(18) IS FIRST RECORD; FCB(19) IS RECORD INCR
      FCB(18)=4
      FCB(19)=(FCB(62)+255)/256
      ELSE
C.....REMAPP FILE
      WRITE(6,70) (FCB(I),I=8,11),FCB(24),FCB(23),FCB(25)
      FCB(18)=1
      FCB(19)=1
      ENDIF
      WRITE(6,61)
61      FORMAT(1X,'ENTER 1ST SCANLINE, NO. SCANLINES, SKIPS: ')
      READ(5,*) FCB(5),FCB(16),FCB(7)
      WRITE(6,62)
62      FORMAT(1X,'ENTER 1ST PIXEL, NO. PIXELS, SKIPS: ')
      READ(5,*) FCB(4),FCB(15),FCB(6)
C.....DEFAULT STARTING VALUES
      IF(FCB(5).LE.0) FCB(5)=1
      IF(FCB(4).LE.0) FCB(4)=1
C.....STARTING VALUES TOO LARGE?
      IF(FCB(4).GT.FCB(23)) FCB(4)=FCB(23)
      IF(FCB(5).GT.FCB(24)) FCB(5)=FCB(24)
C.....NOS. TOO LARGE?
      IF(FCB(4)+FCB(15)-1.GT.FCB(23)) FCB(15)=FCB(23)-FCB(4)+1
      IF(FCB(5)+FCB(16)-1.GT.FCB(24)) FCB(16)=FCB(24)-FCB(5)+1
C.....DEFAULT WINDOW LENGTHS
      IF(FCB(15).LE.0) FCB(15)=FCB(23)-FCB(4)+1
      IF(FCB(16).LE.0) FCB(16)=FCB(24)-FCB(5)+1
C.....LAST RECORD
      FCB(26)=FCB(5)+FCB(16)-1
      IF(FCB(26).GT.FCB(24)) FCB(26)= FCB(24)
C.....FIRST RECORD
      FCB(14)=FCB(19)*(FCB(5)-1)+FCB(18)
C.....LAST PIXEL
      FCB(27)=FCB(4)+FCB(15)-1
      IF(FCB(27).GT.FCB(23)) FCB(27)=FCB(23)
C.....ACTUAL NO. PIXELS AND SCANLINES
      FCB(2)=(FCB(15)-1)/(FCB(6)+1)+1
      FCB(3)=(FCB(16)-1)/(FCB(7)+1)+1
C
70    FORMAT(1X,T14,'DISKIO(2): ',4A4,'[LINES ',I5,
     *'PIXELS ',I4,'BYTESIZE ',I2']')
      FCB(21)=0
      WRITE(4,72) (FCB(I),I=8,11)
72    FORMAT(1X,T15,'INPUT FILE ',4A4,' OPENED:')
      WRITE(4,71)FCB(5),FCB(16),FCB(7),FCB(3),
     * FCB(4),FCB(15),FCB(6),FCB(2)
71    FORMAT(1X,T20,'WINDOW',T32,'1ST',T37,'NO.',T41,'SKIP',
     *T46,'ACTUAL',/,1X,T20,'SCANLINES',T30,4I5,/
     *,1X,T20,'PIXELS',T30,4I5)
      FCB(21)=0
      GO TO 20
C
C OPEN AN OUTPUT FILE
C
80    CONTINUE
      FCB(25)=FCB(1)
      FCB(23)=FCB(2)
      FCB(12)=FCB(23)*FCB(25)/8
      FCB(14)=1
      FCB(24)=FCB(3)
      FCB(21)=1
      IF(FCB(15).NE.4321) THEN
C.....CREATE REMAPP FILE
      IBSIZE=32
      NBYTES=MAX(256,FCB(12))
      CALL CFILW(FILEDESC,2,NBYTES,IBSIZE,3,0,10,CODE)
      ELSE
C.....CREATE ELAS FILE
      IFSIZE=(((FCB(2)*FCB(1)/8)+255)/256)*FCB(3)*FCB(13)+4
      CALL CFILW(FILEDESC,0,0,IFSIZE,0,0,0,CODE)
      ENDIF
      IF(CODE.NE.0) WRITE(6,398) CODE
398   FORMAT(1X,'DISKIO: CREATE ERROR = ',I3)
      IF(CODE.NE.0)GO TO 300
C.....OPEN CREATED FILE, ASSIGN TO LOGICAL UNIT
      CALL OPENW(UNIT,FILEDESC,5,0,10,CODE)
      IF(CODE.NE.0) WRITE(6,399) CODE
      IF(CODE.NE.0)GO TO 300
C.....WRITE FILE HEADER
C.....REMAPP HEADER
      IF(FCB(15).NE.4321) THEN
      IFC=41
      CALL SYSIO(PBLK,IFC,UNIT,FCB,1024,0)
      ELSE
C.....ELAS HEADER STUFF
      ELAS(57)=FCB(8)
      ELAS(58)=FCB(9)
      ELAS(59)=FCB(10)
      ELAS(60)=FCB(11)
      ELAS(19)=FCB(1)
      ELAS(8)=4321
      ELAS(7)=FCB(13)
      ELAS(6)=FCB(2)
      ELAS(5)=1
      ELAS(4)=FCB(3)/FCB(13)
      ELAS(3)=1
      ELAS(2)=256*((FCB(12)+255)/256)
      ELAS(1)=1024
      IFC=41
      CALL SYSIO(PBLK,IFC,UNIT,ELAS,1024,0)
      ENDIF
      CALL IOERR(PBLK,ISTAT)
      IF(ISTAT.NE.0) GOTO 300
      GO TO 20
C
C  REWIND - POINT TO FIRST DATA RECORD
C
100   FCB(14)=(FCB(5)-1)*FCB(19) + FCB(18)
      GO TO 20
C
C  SKIP SPECIFIED NUMBER OF RECORD
110   FCB(14)=FCB(14)+FCB(1)*FCB(19)
      GO TO 20
C
C  BACKSPACE SPECIFIED NUMBER OF RECORDS
C
120   FCB(14)=FCB(14)-FCB(1)*FCB(19)
      GO TO 20
C
C  READ TO A SPECIFIC RRECORD
C
130   FCB(14)=(FCB(1)-1)*FCB(19) + FCB(18)
      GO TO 140
C
C READ A NEXT RECORD
C
140   FCB(1)=0
      IF(FCB(14).LT.(FCB(5)-1)*FCB(19) + FCB(18)) GOTO 300
      IF(FCB(14).LE.(FCB(26)-1)*FCB(19) + FCB(18)) GO TO 142
      WRITE(6,180)(FCB(I),I=8,11)
```

SUBROUTINE TAPEIO

TAPEIO provides a facility for reading and writing tapes. Other operations allow user programs to space records and files forward and backward and to rewind the tape. Computer-compatible data tapes are assumed to contain one eight-bit data byte per one tape byte. Both 800 and 1600 Bpi tape recording densities can be read or written, but this depends upon the system software that operates the tape drives and hardware settings at time of operation and not upon TAPEIO. TAPEIO does no conversion, reformatting, packing or unpacking of data. The calling program must do this.

The general form of a call to TAPEIO is as follows:

```
CALL TAPEIO(OPERATION,IOMODE,IOAREA,N)
```

where:

OPERATION is an integer specifying the function to be performed. All operations are on logical unit 7. A REMAPP program is not expected to both tape input and output.

IOMODE is an integer specifying input or output operating mode. The operating mode cannot be changed after the file is opened, and only after the file is closed.

```
IOMODE=0 - input file mode
      =1 - output file mode
```

IOAREA is an integer or logical array into which a tape data record is read or from which a tape data record is to be written.

N is an integer used to specify parameters to TAPEIO and on return to the calling program contains the operation status.

Operations

The following operations are valid:

0 = open a tape file on logical unit 7 for mode set in IOMODE
1,3 = rewind tape to load point
4 = skip no. of records specified in N
5 = backspace no. of records specified in N
6 = close the tape file
7 = skip no. of files specified in N
8 = backspace no. of files specified in N
9 = read tape physical record into IOAREA
10 = write tape physical record from IOAREA

OPERATION=0; IOMODE=0

OPERATION=0 and IOMODE=0 opens a tape file for input. The following request is made:

TYPE TAPE DEVICE NAME:

The physical device name, M0: or M1:, is assigned to logical unit 7. Tape-to-tape operations requiring both reading a tape and writing another tape within a program are not possible with TAPEIO. If the tape device cannot be opened, TAPEIO returns with error no. -2 in N. If the opening is successful, the following is written on logical unit 4:

TAPEIO: FILE OPENED; MODE=0

and TAPEIO returns with N=0.

OPERATION=0; IOMODE=1

OPERATION=0 and IOMODE=1 opens a tape file for output. The following request is made:

TYPE TAPE DEVICE NAME:

The physical device name, M0: or M1:, is assigned to logical unit 7. Tape-to-tape operations requiring both reading a tape and writing another tape within a program are not possible with TAPEIO. If the tape device cannot be opened, TAPEIO returns with error no. -2 in N. If the opening is successful, the following is written on logical unit 4: The tape is not checked for a write ring.

TAPEIO: FILE OPENED; MODE=1

and TAPEIO returns with N=0.

OPERATION=1,3

OPERATION=3 rewinds the tape to the load point. If an error, a message is given and TAPEIO returns with error no. -2 in N.

OPERATION=4

OPERATION=4 causes forward spacing the no. of records specified in N. Upon return, if no error N = 0, else N is the negative of the no. of records actually spaced.

OPERATION=5

OPERATION=5 causes backspacing of the no. of records specified in N. Upon return, if no error N = 0, else N is the negative of the no. of records actually spaced.

OPERATION=6

OPERATION=6 closes the tape file and leaves the tape positioned and not rewound. If the file is an output file, two end-of-file marks are written and the tape is positioned between them.

OPERATION=7

OPERATION=7 causes forward spacing the no. of files specified in N. and leaves the tape position at the beginning of a file. Upon return, if no error N = 0, else N is the negative of the no. of files actually spaced.

OPERATION=8

OPERATION=8 causes backspacing the no. of files specified in N and leaves the tape positioned at the beginning of a file. Upon return, if no error N = 0, else N is the negative of the no. of files actually spaced.

OPERATION=9

OPERATION=9 reads from a physical tape record the no. of bytes specified in N into array IOAREA. No conversion or unpacking of the data is done. On return N contains the actual no. of bytes read which is never greater than the no. requested. If an end-of-file is read, then N = -1 on return. If I/O errors occur, then N = -2 on return. If this operation is attempted when IOMODE = 1, the message is given:

```
READ/WRITE TO WRONG TAPE DEVICE
```

and TAPEIO returns with N = -2. If an attempt is made to read past the physical end of tape, i. e., the file spans two or more tape volumes, then the message is given:

```
MOUNT NEXT TAPE, TYPE CONTINUE
```

OPERATION=10

OPERATION=10 writes from the array IOAREA the number of bytes specified in N to a physical tape record. No conversion or packing of the data is done. On return N contains the no. of bytes written. If an I/O error occurs, then N = -2 on return. If this operation is attempted when IOMODE = 0, the message is given:

```
READ/WRITE TO WRONG TAPE DEVICE
```

and TAPEIO return with N = -2. If an attempt is made to write past the physical end of tape, i. e., the file spans two or more tape volumes, then the message is given:

```
MOUNT NEXT TAPE, TYPE CONTINUE
```

System Error Messages

On tape spacing or I/O error a diagnostic message will be given for the Fortran VII Run Time Library:

```
ERR n (mmmmmm) LU # 7:
message
```

where:

```
         n=-1  HARDWARE EOF DETECTED  (usually not an error)
         n=26  PARITY OR RECOVERABLE ERROR  (no write ring?)
         n=27  UNRECOVERABLE ERROR
         n=28  END OF MEDIUM  (physical end of tape?)
         n=29  DEVICE UNAVAILABLE  (tape drive offline?)
```

TAPEIO returns with the error no. -2 in N.

Subroutines Called

The following Fortran Run Time Library subroutines are called by TAPEIO: OPENW, SYSIO, IOERR, CLOSE.

TAPEIO Fortran Listing

```
      SUBROUTINE TAPEIO(IOPR,MODE,IBUF,N)
$INCLUDE REMAPP.HDR
C   *****************************************************************
C
C   REMAPP P-E/3220 1985
C   REMote sensing Array Processing Procedures
C   U. S. GEOLOGICAL SURVEY, DENVER, COLORADO
C   BRANCH OF GEOPHYSICS, REMOTE SENSING LABORATORY
C   DON L. SAWATZKY
C
C   *****************************************************************
C.....................
C     OPERATIONS
C
C     OPER=0 - OPEN THE TAPE FILE ON LOGICAL UNIT 7
C          3 - REWIND, SET AT TAPE LOAD POINT
C          4 - SKIP NO. RECORDS SPECIFIED IN ARGUMENT N
C          5 - BACKSPACE NO. RECORDS SPECIFIED IN ARGUMENT N
C          6 - CLOSE THE TAPE FILE, WRITE 2 FILEMARKS, BACKSPACE
C              1 FILEMARK
C          7 - FORWARD SPACE PAST ONE FILE MARK
C          8 - BACKSPACE TO BEFORE ONE FILE MARK
C          9 - READ RECORD INTO ARRAY IOAREA
C          10 - WRITE RECORD FROM ARRAY IOAREA
C          11 - READ RECORD NO. SPECIFIED IN ARGUMENT N INTO C
ARRAY IOAREA
C
C....................
C
C     MODE = I/O TRANSFER MODE
C     MODE=0 TAPE FILE IS OPEN FOR INPUT ONLY
C         =1 TAPE FILE IS OPEN FOR OUTPUT ONLY
C....................
C
C     IBUF = INTEGER OR LOGICAL ARRAY DIMENSIONED TO CONTAIN
C            ENOUGH BYTES TO HOLD ONE TAPE PHYSICAL RECORD
C
```

```
C.........................
C
C     N = NO. OF BYTES TO READ/WRITE OR NO. OF SPACING OPERATIONS
C         RETURNS NO. OF BYTES READ OR NO. OF SPACING OPERATIONS
C         RETURNS -1 FOR END OF FILE OR -2 FOR I/O ERROR.
C..........................
      INTEGER*4 PBLK(5),IBUF(1)
      DATA ILU/7/
C
C....BRANCH TO OPERATION
      NBYTES=N
      GOTO(20,1,1,1,4,5,6,7,8,9,10) IOPR+1
C
C....ASSIGN FUNCTION CODES
1     IFC=192;REWIND
      GOTO 40
4     IFC=144;FORWARD RECORD
      NOPS=N
      GOTO 30
5     IFC=160;BACKWARD RECORD
      NOPS=N
      GOTO 30
6     IF(MODE.NE.1) GOTO 61
      NOPS=2
      IFC=136;WRITE FILEMARK
      GOTO 30
7     IFC=132;FORWARD FILE
      NOPS=N
      GOTO 30
8     IFC=130;BACKWARD FILE
      NOPS=N
      GOTO 30
9     IFC=91;READ,BINARY,IMAGE
      IF(MODE.NE.0) GOTO 50
      GOTO 40
10    IFC=59;WRITE,BINARY,IMAGE
      IF(MODE.NE.1) GOTO 50
      GOTO 40
20    MODE=MOD(MODE,2)
      WRITE(6,*)'TYPE TAPE DEVICE NAME:'
      READ(5,FMT='(A4)') MAG
      CALL OPENW(7,MAG,2*MODE+1,0,0,ISTAT)
      N=0
      IF(ISTAT.NE.0) N=-2
      IF(ISTAT.EQ.0) WRITE(4,99) MODE
99    FORMAT(/,1X,'TAPEIO: FILE OPENED; MODE=',I2)
      RETURN
C....REPEATED OPERATIONS
30    CONTINUE
      DO 31 I=1,NOPS
      CALL SYSIO(PBLK,IFC,ILU,IBUF,NBYTES,0)
      CALL IOERR(PBLK,ISTAT)
31    IF(ISTAT.NE.0) GOTO 32
      IF(IOPR.NE.6) RETURN
```

```
C....BACKSPACE OVER LAST WRITTEN FILEMARK
      IFC=130
      GOTO 40
C....ERROR HANDLER
32    IF(IAND(ISTAT,Y'8').EQ.0) GOTO 33
C....END OF FILE
      IF(IOPR.EQ.9) THEN
      N=-1
      ELSE
      N=-(I-3)
      ENDIF
      RETURN
C....UNRECOVERABLE ERROR
33    N=-2
      RETURN
C....SINGLE OPERATION
40    CALL SYSIO(PBLK,IFC,ILU,IBUF,NBYTES,0)
      CALL IOERR(PBLK,ISTAT)
      IF(ISTAT.NE.0) THEN
      IF(ISTAT.EQ.X'90'.AND.(IOPR.EQ.10.OR.IOPR.EQ.9)) THEN
      CALL SYSIO(PBLK,192,ILU,IBUF,NBYTES,0)
      PAUSE 'MOUNT NEXT OUTPUT TAPE; TYPE CONTINUE'
      GOTO 40
      ELSE
      GOTO 32
      ENDIF
      ENDIF
      N=PBLK(5)
      IF(IOPR.NE.6) RETURN
C.....CLOSE TAPE FILE
61    CALL CLOSE(7,ISTAT)
      N=0
      IF(ISTAT.NE.0) N=-2
      IF(ISTAT.EQ.0) WRITE(4,97) MODE
97    FORMAT(/,1X,'TAPEIO: FILE CLOSED: MODE=',I2)
      RETURN
C....ERROR IN R/W TO WRONG MODE
50    WRITE(4,98)
98    FORMAT(/,1X,'READ/WRITE TO WRONG TAPE DEVICE')
      N=-2
      RETURN
      END
```