U.S. DEPARTMENT OF THE INTERIOR

GEOLOGICAL SURVEY

REMAPP Multics Programmer's Guide

by

Don L. Sawatzky

and

Linda M. Crownover

Open-File Report 80-1259

1980

DISCLAIMERS

REMAPP Multics users may consult this document for most commonly used and straightforward image-processing procedures. It is not intended to be an introduction to Multics nor to the scientific use of computers. The REMAPP Multics user is expected to have experience in scientific computer use and to have taken the course Introduction to Multics.

Use of brand names is for descriptive purposes only and does not constitute endorsement by the U.S. Geological Survey.

Contents

ACKNOWLEDGMENTS

A number of ideas that were instrumental in the development of REMAPP came from the U.S. Geological Survey image-processing facility in Flagstaff, Arizona. The subroutine "TAPEIO" was based on Flagstaff's subroutine.

The subroutine used for geometric rectification of Landsat imagery, "FASTGE", was developed by Patrick E. Chavez of the U.S. Geological Survey in Flagstaff. We are indebted to him for his generous help with this subroutine.

Chapter I

INTRODUCTION

REMAPP (REMote sensing Array Processing Procedures) is a set of applications programs and utility subroutines designed to facilitate the processing of remotely sensed image data. REMAPP is currently implemented on Honeywell Multics computers operated by the U.S. Geological Survey in Denver, Colorado and Reston, Virginia.

REMAPP allows processing of image files from data tapes of different formats, such as Landsat. These data are first converted to a single general format, and then stored as disk files. Existing applications programs provide several processing capabilities: arithmetic operations on and between image files; reading Landsat data tapes; and reading and writing Optronics-format data tapes.

This guide introduces the programs OPTRIN and OPTAPE, which provide for the interchange of data between REMAPP-format disk files and Optronics-format magnetic tape files. Also outlined are the subroutines DISKIO and TAPEIO, which are called by these programs. These subroutines perform input and output to disk and tape, as well as packing and unpacking of data to conserve file space.

This guide is intended for users who are interested in writing digital-image processing programs that access the REMAPP Multics data base. For a more detailed explanation of REMAPP, users should consult a REMAPP Multics User's Manual.

Chapter II

PROGRAM OPTAPE

INTRODUCTION

OPTAPE is a program that writes Optronics-format tapes from REMAPP-format disk files. Image annotation is provided, and geometric rectification for Landsat imagery may be performed. In addition, a log of operations is maintained by Remapp_log.

IMAGE-ANNOTATION FACILITIES

Two forms of annotation are provided, 1) the inscription of lines and/or rectangles on the image, and 2) the addition of character strings following the image.

An image is inscribed by assigning all pixels within the designated lines a fixed value, the value of the variable FILL. This results in rectangles or lines being "drawn" on the image. To inscribe an image, the user must set the variable SETS equal to the number of lines and/or rectangles. A maximum of 50 is allowed. The width and density value of the lines must be specified by the variables WIDTH and FILL, respectively. The user must then describe each line or rectangle by giving the coordinates of the upper left corner pixel within the rectangle. A line is drawn by setting the width (in one direction) of the rectangle equal to the line width.

To generate character strings on the image, the user must set the variable TITLES equal to the number of lines of text he wishes to write on the output. He must then enter the lines of text as described below. Character size is controlled by setting the variable SCALE. A mirror image of the text may be produced by setting the variable FLIP equal to one. The 64 character graphic subset of ASCII is supported, that is, all alphabetic characters (A-Z), all numeric symbols (0-9), and the special symbols ! " # $ % & ' ( ) * + , - . / : ; < = > ? @ [¬] ^ _ and space are allowed.

GEOMETRIC RECTIFICATION FOR LANDSAT IMAGERY

OPTAPE performs a line-by-line geometric rectification of the input image according to fixed parameters peculiar to Landsat imagery. Corrections are performed for the non-polar orbit of the satellite, the movement of the Earth under the satellite (skewing), and oversampling in the scan-line direction (aspect-ratio correction). The corrections are performed by deleting pixels from the scan line and offsetting the line to account for the skew. The user must set the variable GCOR equal to one, and the variables LATD, LATM, and LATS equal to the latitude of the upper left corner of the complete Landsat frame in degrees, minutes, and seconds, respectively. The latitude should be specified to the nearest second. Error in ground location may be up to 5% locally if the latitude is within the continental United States, and more if in extreme latitudes.

PROGRAM OPERATION

OPTAPE determines its input and operational parameters by reading control information interactively from the user's terminal. The first record written to tape is a label for the file. This label contains a file identifier that is the first eight characters of the input file name. Data records are then written sequentially. First, a record is read from the input file, then inscription is performed if requested, then geometric correction is performed if requested, and finally, the record is written to tape. The cycle continues until either a user-specified number of records is written, or an end-of-file mark is encountered on the input file. After the image has been written to tape, the character annotation is input from the user's terminal and then written to tape in image format. An end-of-file mark is then written following the file. Data are written on tape as 8-bit bytes. That is, only the rightmost eight bits of the input data are used. OPTAPE does not reposition the tape either before or after writing a file and the EOF mark. If an end of tape occurs during execution of OPTAPE, the program is aborted, but the data already written to tape is usable.

EXECUTION OF OPTAPE

Prior to running OPTAPE, the logical device name "ldvout" must be assigned to a nine-track tape drive. This can be done by using the exec_com tape.ec as follows:

```
ec Remapp>tape mount ldvout [tapeid] [track] [density]
```

The character string "tapeid" is the user's name for the tape reel. OPTAPE is executed by the command:

```
optape
```

After the program is loaded, it begins an interactive dialog with the user. The user is prompted to answer queries and should respond as indicated. To write more than one file to tape, the user must set the variable FILES equal to the number of files he desires to write. The appropriate user responses are detailed on the next page.

DETAILS OF USER RESPONSES

The program queries are listed below with an explanation of the appropriate response.

ENTER NAMELIST PARAMETERS

The user is prompted to enter the namelist parameters described below in the following format:

```
$parms(variable=value, etc.)$
```

The default values are given in parentheses.

```
files = no. of output files to be written on tape          (1)
sets = no. of lines to be inscribed                        (0)
fill = number to annotate inscribed line(s)                (0)
width = width of inscribed line in pixels                  (1)
gcor = 0 = no geometric rectification                      (0)
    >0 = geometric rectification
latd,latm,lats = latitude of northwest corner of
    Landsat image to be corrected; in degrees,
    minutes, seconds                                       (0,0,0)
concat = >1 = no. of files to concatenate                  (1)
    < = 1 = none
flip = >1 = reverse scan line                              (0)
    0 = no reverse
scale = >1 = enlargement factor on text annotation         (3)
    1 = no enlargement
enlarge = >1 = enlargement factor of output image          (1)
    1 = no enlargement
titles = -2 = registration marks, filename, banner
    appended                                               (-1)
    -1 = filename, banner
    0 = no appended information
    n>0 = same as -2 with n lines of caption typed in
    by user
append = >0 = no. of files to append                       (0)
    0 = no appending
appendlength = 0 = appended file length is (no. of
    scanlines of first file opened)*(append + 1)           (0)
    >0 = appended file length
```

```
ENTER SCRIBE PARAMETER SETS, ONE SET PER LINE
NO. OF PIXELS/NO. OF LINES/STARTING PIXEL/STARTING LINE/CLEAR
```

The inscription parameter SETS are entered, one set per line, with the number of lines equal to the value of "SETS". Each set consists of five integers separated by blanks. The integers may be anywhere on the line, but must be in the following order:

1) the number of pixels inside the rectangle
2) the number of lines inside the rectangle
3) the starting pixel number
4) the starting line number
5) zero to inscribe the rectangle, or one to additionally clear the interior

```
ENTER TEXT ANNOTATION
```

The lines of text for character annotation are entered here, with the number of lines equal to the value of "TITLES". Each line may contain up to 35 characters.

EXAMPLE OF OPTAPE EXECUTION

```
ec Remapp>tape mount ldvout pltape 9 800

Tape pltape,9track,blk=2800 will be mounted with a write ring.

Tape pltape,9track,blk=2800 mounted on drive 6 with a write ring.
r REMAPP 12/20/78 1106.3 COST $0.94 $24.34 VCPU 0.917 65.698

optape
$parms files =              1,concat =         1,append =
     0,appendlength =             0,sets =          0,fill =
         0,width =          1,gcor =          0,latd =
    0,latm =                0,lats =      0,scale =               3,
  titles =                 -1,enlarg =    1,flip =                0$

 ENTER NAMELIST PARAMETERS.
$parms  titles=1$

diskio:  ENTER INPUT FILE NAME:  test1
 ENTER 1st scanline, no. scanlines, skips:
0 0 0
 ENTER 1st pixel, no. pixels, skips:
0 0 0
         %Diskio(2):  test1       [Lines   880/Pixels  1230/Bytesize 36]

Diskio:  end-of-information reached

ENTER TEXT ANNOTATION.
 usgs denver colo

STOP
r REMAPP 12/20/78 1134.9 COST $17.98 $42.95 VCPU 37.916 104.236
```

The above example is the terminal log of a user's dialog with OPTAPE. The disk file "test1" was read and transferred to tape "pltape". Default values were used in the namelist parameters, with the exception of TITLES. This was set to 1 to allow character annotation.

EXAMPLE OF OPTAPE LOG

```
                 OPTAPE - REMAPP Multics 12/20/78  1109.2 mst Wed

11:10

    Input segment test1                opened:
         Window          1st     no. skip  actual
         scanlines         1     880    0   880
         pixels            1    1230    0  1230

11:34

    Tape file test1      contains  980 scanlines by 1230 pixels

REMAPP Multics Remapp_off - 12/20/78   1136.4 mst Wed

11:36  Tape dismounted:  Tape_id= pltape, Switch_name= ldvout
```

The above is an example of the log generated by REMAPP when program OPTAPE is executed.

CHAPTER III

PROGRAM OPTRIN

INTRODUCTION

Program OPTRIN writes REMAPP-format disk files from Optronics-format tape files. The user enters an interactive dialog with the program and specifies reading the entire file, or a subset of the file, and the filename for the output disk file. When OPTRIN finishes reading the file, the user may then read the next tape file, or terminate the program.

PROGRAM OPERATION

When OPTRIN begins execution, it opens the current tape file; therefore, the user must space to the first tape file he wishes to read before executing the program. The user is then prompted to determine if the entire file, or only a subset of the file, is to be read. The first 54 bytes of the first record of the file are then read. If the first byte has a value of 255, this record is assumed to be a header record. If the first byte does not have a value of 255, then the program backspaces a record to again point to the beginning of the file. If the user specified a starting line other than 1 (default), OPTRIN skips to the specified record. The starting line is then read, the number of pixels per line is determined, and the output file is opened. The file identifier, or "NO HEADER" if there was no header record, and the number of pixels per line of the first line are then displayed on the user's terminal. The user is then prompted to enter the output file name: that is, the name of the disk file to be created. Following this, the program enters a write-read cycle until either an end-of-file mark is reached on the tape, or a user-specified number of records has been read. When records are written to disk, the X and Y offset numbers are stripped from the beginning of the tape record, and the data are stored with 8-bit precision. If, on completion of reading the file, no end-of-file mark is reached, OPTRIN spaces to point to the next file. The user may then terminate the program or read the next file.

EXECUTION OF OPTRIN

Prior to running OPTRIN, the logical device name "ldvin" must be assigned to a nine-track tape drive. This can be done by using the exec_com tape.ec as follows:

```
ec Remapp>tape mount ldvin [tapeid] [track] [density]
```

The character string "tapeid" specifies the user's name for the tape reel. OPTRIN is executed by the command:

```
optrin
```

The program begins an interactive dialog with the user. The user is prompted to answer queries and should respond as indicated. The queries are repeated for each file processed. The appropriate responses to program messages are detailed below.

DETAILS OF USER RESPONSES

The following is a list of messages issued by OPTRIN. Messages not appearing in this list but occurring during the execution of OPTRIN are issued by REMAPP subroutines. The character string "fileid" used below specifies the user's name for the tape file and, if the tape file has a header, it is the tape file identifier character string.

```
TAPE FILE SUBSET:  FIRST LINE/# OF LINES/FIRST PIXEL/# OF PIXELS?
```

The user is prompted to enter 1) the number of the first line in the file to be read, 2) the total number of lines to be read, 3) the number of the first pixel in the first line to be read, and 4) the total number of pixels per line to be read. The four numbers are entered as integers separated by blanks, and the response is terminated by a carriage return. If zeros are entered, the defaults are in effect; namely, all the pixels of all the lines are read.

```
READ NEXT FILE (Y,N)?
```

The user is prompted to enter a "y" if he desires to read the next tape file, or an "n" to terminate execution of the program. The response is terminated with a carriage return.

```
NO HEADER or ID=FILEID
PIXELS/LINE=N
```

The above messages are issued by OPTRIN prior to opening the output file, and after reading the tape file header, if one existed. The tape file header appears as "FILEID", or "NO HEADER" appears if the file did not contain a header. The number of pixels per line of the starting scanline is given as "N".

ERROR MESSAGES

```
LOGICAL END OF TAPE
ERROR READING TAPE FILE HEADER
ERROR OPENING TAPE FILE
```

These messages indicate that a call to the subroutine TAPEIO has failed with regard to the input file. The program is aborted.

50 TAPE ERRORS

This message indicates that 50 tape errors have been found while reading the input file. The number of errors indicate that the image data are poor, and once 50 errors are found, OPTRIN skips to the end of the file. The program terminates in the normal manner.

EXAMPLE OF OPTRIN EXECUTION

```
ec Remapp<tape mount ldvin rms500 9 900

Tape rms500,9track,blk=2800 will be mounted with no write ring.
Tape rms500,9track,blk=2800 mounted on drive 5 with no write ring.
r REMAPP 12/20/78 1035.2 COST $ 0.82 $2.86 VCPU 1.524 7.544
optrin

Tape file subset:
FIRST LINE/ # OF LINES/ FIRST PIXEL/ # OF PIXELS ?:
0 0 0 0

ID=6eSFRCOJ

PIXELS/LINE=1230

Diskio:  ENTER OUTPUT FILE NAME:  test1
Diskio:  Enter output file bits/byte:  36
TAPEIO:  End of information reached.
            %Diskio(2):  test1    [Lines   880/Pixels 1230/Bytesize 36]

READ NEXT FILE (y,n) ?
n

STOP
```

The above example is the terminal log of a user's dialog with OPTRIN. The tape file "6eSFRCOJ" on rms500 was read and transferred to the disk file "test1".

EXAMPLE OF OPTRIN LOG

```
               OPTRIN - REMAPP Multics 12/20/78  1035.5 mst Wed

10:35
   Opened tape input file:  6eSFRCOJ
   Window(1st line,no lines,1st pixel,no pixels):    1    0    1  1230

10:58
   Output segment closed:
   %Diskio(2):  test1            [Lines   880/Pixels 1230/Bytesize 36]
```

```
REMAPP Multics Remapp_off - 12/20/78   1100.4 mst Wed

11:00  Tape dismounted:  Tape_id= rms500,  Switch_name=  ldvin
```

The above is an example of the log generated by REMAPP when program OPTRIN is executed.

CHAPTER IV

SUBROUTINE DISKIO

INTRODUCTION

DISKIO provides a general facility to read and write REMAPP-format disk files. User programs may 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 line) and the first element of that record (the starting pixel) as well as the numbers of lines and pixels.

To access any REMAPP disk file the user need only supply the segments entry name in standard Multics format. In the case of an input file, the user's directories are searched according to his search rules. If the file is not found, an error message will be displayed on the terminal and the program terminated. In the case of an output file, the file will be written to a segment in the user's current working directory.

Data records are read and written without any conversion or manipulation of the data, other than packing or unpacking. When the user opens a file for output he must specify the precision of the array elements in bits, where 9=packed disk file and 36=unpacked disk file. When a file is opened for input, information on how the data are packed is read from the label (the last record of the file). This information is used to unpack the data when the file is read.

The user establishes access to disk files by calling DISKIO to open a file. It is at this time that the file name, packing precision, and other parameters are specified. The user may then read or write the file, or position to a specific record. The user terminates disk operations by calling DISKIO to close the file. In the case of an output file, this operation writes information to the file label.

DISKIO USAGE

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

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

where:

OPERATION is an integer specifying the function to be performed.

UNIT is an integer specifying the Fortran I/O unit number to be associated with the file.

IOAREA is an integer array to read data into or write data from.

FCB is a 30 element integer array containing information necessary to process the file.

The following operations are valid:

OPERATION =

0	open a file
3	rewind - position to first data record
4	skip fcb(1) records
5	backspace fcb(1) records
6,7,8	close a file
9	read a record
10	write a record
11	position to data record given by fcb(1) - next I/O operation will read/write that record

SUBROUTINES REQUIRED
disk_open - gets the switch_name, attach_description, and address pointers of the arrays

FILE DESCRIPTION
stream input/stream output - Remapp header record is the last record of 120 bytes

OPERATIONS

OPERATION = 0   OPEN AN INPUT FILE

REQUIRED PARAMETER FCB(1) = 0

The user is prompted to enter the name of the file to be opened. Once the filename has been determined, the last 120 bytes of the file are read and this information is placed in the file's associated FCB. If the number of words per record, FCB(12), is designated to be less than 1 or greater than 32767, the number is ignored and the user is prompted to enter a valid number of words per record.

The user is then prompted to enter the first scanline, the number of scanlines and skips, and the first pixel, the number of pixels and skips. If these are zero, as set by the user, then default values are used and FCB(2) and FCB(3) are calculated as follows:

FCB(2)=(NO. OF PIXELS PER LINE-STARTING PIXEL)/(PIXEL SKIP+1)+1

FCB(3)=(NO. OF SCANLINES-STARTING LINE)/(LINE SKIP+1)+1

The record pointer, FCB(14), is set to the starting record, FCB(5), minus one. The default is zero.

Concluding the opening procedure, file information is displayed on the user terminal and control returns to the user's program.

OPERATION=0   OPEN AN OUTPUT FILE

REQUIRED PARAMETERS FCB(1) = 1
                    FCB(2) = SCANLINE LENGTH

The filename is determined as in opening an input file. The user is prompted to enter the output file bits per byte. Concluding the opening procedure, control returns to the user's program.

OPERATION=3   REWIND, POSITION TO FIRST DATA RECORD

The value of FCB(14) is set to zero and the next read operation will input the first data record.

OPERATION=4   SKIP RECORDS FORWARD

REQUIRED PARAMETER FCB(1)=NUMBER OF RECORDS TO SKIP

The value of FCB(1) is added to the record pointer, FCB(14). The next read operation will read the record now pointed to by FCB(14). Note that after a read operation the record pointer is incremented by one.

OPERATION=5 BACKSPACE RECORDS

REQUIRED PARAMETER FCB(1)=NUMBER OF RECORDS TO BACKSPACE

The value of FCB(1) is subtracted from the record pointer, (FCB(14). The next read operation will read the record now pointed to by FCB(14).

OPERATION=6,7,8 CLOSE A FILE

If the file is an input file it is closed and control returns to the user's program. If the file is an output file, DISKIO writes, as the last record of the file, the contents of the FCB. The access code is set to 'r' (read). A message is displayed listing the name of the file closed, the number of lines in the file, the number of pixels per line, and the bytesize. This information will also be stored in a Multics segment remapp_log. Control then returns to the user's program.

OPERATION=9 READ A RECORD

DISKIO reads the record from the file specified by the current record number, FCB(14), into the user's buffer idarea. DISKIO then calls the subroutine CHARPACK to unpack data in the buffer when the bytesize, FCB(25), is less than 36. The record pointer is incremented by one plus the record skip, FCB(7).

The starting pixel, FCB(4), the starting line, FCB(5), the pixel skip, FCB(6), and the line skip, FCB(7), parameters are used to subset input data. These values are specified when the file is opened. The first pixel returned in the array ioarea is the FCB(4)th pixel of the input file. The next pixel returned is from FCB(6) + 1 beyond the FCB(4)th pixel in the input file. Therefore, if FCB(6)=0, then starting with pixel FCB(4), all pixels are returned. If FCB(6)=1, then every other pixel is returned. Similarly with the line skip, FCB(7). After windowing is performed, control returns to the user's program.

If an end-of-information mark occurs when DISKIO attempts to read a file, a message is displayed, FCB(1) is set to -1, and control returns to the user's program.

OPERATION=10 WRITE A RECORD

If FCB(25) is less than 36, DISKIO calls subroutine CHARPACK to pack the data in the buffer in 9-bit bytes for the number of bytes per record, FCB(23). The buffer is then written to disk as record number FCB(14). The record number, FCB(14), and the record count (FCB(24), are incremented by one, and control returns to the user's program. The contents of IOAREA are destroyed if packing was performed. A subset of an output file cannot be taken.

OPERATION=11 POSITION TO A RECORD

REQUIRED PARAMETER FCB(1)=NUMBER OF THE RECORD TO BE POSITIONED TO

The record pointer, FCB(14), is set equal to the value of FCB(1). The NEXT read/write operation will read/write that record. Control returns to the user's program.

DISKIO MESSAGES

The following is a list of messages issued by DISKIO. The message prefix, "?", indicates a fatal error, and FCB(1) is set to -2; a "%" indicates the message is informatory; and a "$" indicates the user is expected to respond to the message.

?DISKIO(1): INVALID OPERATION CODE

The first argument to DISKIO, the operation code, OPERATION, was not one of the allowed operations. The program is aborted.

%DISKIO(2): FILENAME[LINES I/PIXELS J/BYTESIZE K]

The file "FILENAME" has been opened for input or closed on output. The file contains I lines, with J pixels per line. K=9 (packed data) or K=36 (unpacked data).

$DISKIO: ENTER INPUT FILE NAME

The user is prompted to give the entry name of the segment to be read (not greater than 16 characters). The response is terminated by a carriage return.

$DISKIO: ENTER OUTPUT FILE NAME

The user is prompted to enter the name of the segment to be written. The response is terminated by a carriage return.

$DISKIO(11): NUMBER OF 9-BIT BYTES PER RECORD?

On opening the input file, DISKIO has found the specification in the label for the number of words per record, FCB(12), to be invalid. The user is prompted to enter the correct value. The response is terminated by a carriage return.

$DISKIO: ENTER OUTPUT FILE BITS/BYTE

The user is prompted to specify the precision of the array elements in bits, where 9=packed disk file, and 36=unpacked disk file. The response is terminated by a carriage return.

$DISKIO: END OF INFORMATION REACHED

While reading the file, an end-of-information mark was reached. The value of -1 is returned in FCB(1), and control returns to the user's program.

?DISKIO: ENTRY ERROR

An entry error has been detected, and the value of -2 is returned in FCB(1). Control returns to the user's program.

$DISKIO: [SYSTEM_MESSAGE USER_MESSAGE]

An error has been encountered by the system during a DISKIO procedure. A system code is displayed and a corresponding message to the user, indicating the nature of the error. The value of -2 is returned in FCB(1), and control returns to the user's program.

FILE CONTROL BLOCK (FCB) FORMAT

TABLE 1

ARRAY ELEMENT	MNEMONIC	USAGE
1	PARM	PASS PARAMETERS TO DISKIO
2	PIXELS	# OF BYTES/RECORD OF FILE SUBSET
3	LINES	# OF RECORDS/FILE OF FILE SUBSET
4	SBYTE	STARTING BYTE NUMBER
5	SREC	STARTING RECORD NUMBER
6	BYTSKP	BYTE SKIP FACTOR
7	RECSKP	RECORD SKIP FACTOR
8-11	NAME	FILENAME IN 9-BIT ASCII
12	WORDS	# of 9-bit bytes PER DATA RECORD
13		unused
14		current record number
15		no. pixels in window
16		no. scanlines in window
17-19		unused
20	IOUNIT	FORTRAN I/O UNIT NUMBER
21	IO	0 FOR INPUT, 1 FOR OUTPUT FILE
22		unused
23	BYTES	NUMBER OF BYTES PER RECORD
24	DARECS	NUMBER OF LINES IN THE FILE
25	BITS	NUMBER OF BITS PER BYTE
26		last record no. of window
27		last pixel no. of window
28		unused
29-30		i/o switch pointer

CHAPTER V

SUBROUTINE TAPEIO

INTRODUCTION

TAPEIO provides a general facility for reading and writing magnetic tapes. Other operations allow user programs to space records or files forward or backward, and to rewind the tape.

Data tapes are assumed to be industry compatible: that is, data exists as eight-bit bytes. After a read operation, TAPEIO unpacks the data from a buffer into an array, one data byte per word. When data are written, they are packed into the same array, four eight-bit bytes per word, and then written to tape. No conversion or reformatting of the data, other than packing or unpacking, occurs.

The exec_com tape.ec should be invoked prior to running any programs that call subroutine TAPEIO. Tape.ec handles the attachments and detachments of the I/O tape modules, as well as positioning of the tape. The user may also specify whether a 7- or 9-track tape is to be used.

USAGE

```
ec Remapp>tape arg1 arg2 arg3 arg4 arg5
```

arg1 =	tape operations: mount, dismount, rewind, skip__file, backspace_file, skip_record, backspace_record
arg2 =	"ldvin" = switch name for input tape file "ldvout" = switch name for output tape file
arg3 =	n>0 number of files/records to skip/backspace n=0 backspace to beginning of file tapeid = for tape mounts
arg4 =	tape tracks (7 or 9); for initial tape mounts to set or reset, optional if device is already assigned
arg5 =	density; for initial tape mounts to set or reset, optional if device is already assigned

TAPEIO USAGE

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

```
call TAPEIO (OPERATION, IO, IOAREA,N)
```

where:

OPERATION is an integer specifying the function to be performed.

IO is an integer with a value of 0 to indicate an operation on an input file, and 1 to indicate operation on an output file.

IOAREA is an array to read data into or write data from, and may be dimensioned to be as large as the magnetic-tape buffer, (8192 words).

N is an integer used to specify parameters to TAPEIO and on return from the subroutine is an error code.

The following operations are valid:

OPERATION =

0	open a tape file
1 or 3	rewind tape to load point
4	skip record(s)
5	backspace records
6	close a tape file
7	skip file(s)
8	backspace file(s)
9	read a record
10	write a record

OPERATIONS

```
OPERATION=0 OPEN AN INPUT FILE
            REQUIRED PARAMETER IO=0
            CALL TAPEIO (0,0,0,0)
```

TAPEIO accesses the file and control is returned to the calling problem. A file must be opened before a read operation.

```
            OPEN AN OUTPUT FILE
            REQUIRED PARAMETER IO=1
            CALL TAPEIO(0,1,0,0)
```

TAPEIO accesses the file and control is returned to the calling program. A file must be opened before a write operation.

```
OPERATION=1,3 REWIND TAPE TO LOAD POINT
                REQUIRED PARAMETERS IO=0 (INPUT FILE) 1 (OUTPUT FILE)
                CALL TAPEIO (3,IO,0,0)
```

TAPEIO rewinds the specified tape to the load point, and control is returned to the calling program.

```
OPERATION=4 SKIP RECORD(S)
            REQUIRED PARAMETER N=NUMBER OF RECORDS TO SKIP
            CALL TAPEIO(4,0,0,N)
```

TAPEIO skips n records forward and control is returned to the calling program. If n is 0, one record is skipped.

```
OPERATION=5 BACKSPACE RECORD(S)
            REQUIRED PARAMETER N=NUMBER OF RECORDS TO BACKSPACE
            CALL TAPEIO (5,0,0,N)
```

TAPEIO backspaces n records and control is returned to the calling program. If n is 0, one record is backspaced.

```
OPERATION=6 CLOSE A FILE
            REQUIRED PARAMETER IO=0 (INPUT FILE) 1 (OUTPUT FILE)
            CALL TAPEIO (6,IO,0,N)
```

TAPEIO closes the specified file and control returns to the calling program. An end-of-file mark is written on an output file. A file should be closed upon completion of reading or writing the file.

```
OPERATION=7 SKIP FILE(S)
            REQUIRED PARAMETER N=NUMBER OF FILES TO SKIP
            CALL TAPEIO (7,0,0,N)
```

TAPEIO skips forward n files, and control returns to the calling program. If N is 0, one file is skipped.

```
OPERATION=8 BACKSPACE FILE(S)
            REQUIRED PARAMETER N=NUMBER OF FILES TO BACKSPACE
            CALL TAPEIO (8,0,0,N)
```

TAPEIO backspaces n files, and control is returned to the calling program. If N is 0, one file is backspaced.

```
OPERATION=9 READ A RECORD
            REQUIRED PARAMETERS IBUF=AN ARRAY TO READ DATA INTO
            CALL TAPEIO (9,0,IBUF,N)
```

TAPEIO reads the data into IBUF, unpacking it one byte per array element. On return, N is the number of bytes read (record length). If an end-of-file occurred, the value of -1 is returned in N, and if other tape errors occurred, the value of -2 is returned. Control returns to the calling program.

```
OPERATION=10 WRITE A RECORD
             REQUIRED PARAMETERS IBUF=AN ARRAY TO WRITE DATA TO
                                 N=NUMBER OF BYTES TO WRITE
             CALL TAPEIO (10,1,IBUF,N)
```

TAPEIO packs the data in the array and into four eight-bit bytes per work. TAPEIO then writes the data from the buffer to tape. If an error occurs during a write operation, the value of -2 it returned in N. Control returns to the calling program.

TAPEIO ERROR MESSAGES

TAPEIO: ILLEGAL OPERATION=

TAPEIO encountered an illegal operation, and the value of -2 was returned in N. The operation failed, and control returns to the calling program.

[SYSTEM_MESSAGE USER_MESSAGE]

An error has been encountered by the system during a TAPEIO procedure. A system code is displayed and a corresponding message to the user, indicating the nature of the error. The value of -2 is returned in N, and control returns to the calling program. If an end-of-information mark was reached, the value of -1 is returned in N.

CHAPTER VI

REMAPP SUBROUTINES

This chapter presents user information on other REMAPP subroutines and function subprograms.

SUBROUTINE MASK

Purpose

Add character annotation to the end of an Optronics-format image file. The entire 64 character graphic subset of ASCII is available.

Usage

```
COMMON/MASKC/TITLES,FLIP,SCALE,IXOF1,IXOF2

CALL MASK(IBUF2)
```

Description of parameters

TITLES	= number of character strings
FLIP	= 1 to reverse the line
SCALE	= size of character matrix = 6 * scale
IXOF1	= X-offset, left 8 bits
IXOF2	= X-offset, right 8 bits
IBUF2	= output buffer, starting at IBUF2(7), IBUF2(1-6) are used as position data by the Optronics system

All parameters are integers greater than or equal to zero.

Subroutines and function programs required

TAPEIO

Date

Time

Remarks

Each character that can be drawn is represented as a 6 x 6 bit array. The program determines the placement of the character and looks up the bit pattern of each line in the array. Integer scaling is accomplished by repeating each bit a certain number of times (scale). For example, if SCALE=5, each character will be represented on output as a 30 x 30 bit array. A maximum of 2048 bits per line is allowed.

A closing line is printed with a logo and the date and time.

SUBROUTINE FASTGE

Purpose

Geometric rectification for Landsat imagery is performed on a line-by-line basis for over-sampling in the scan-line direction (aspect-ratio correction), and the nonpolar orbit of the satellite and the rotation of the Earth under the satellite (skew correction).

Usage

```
COMMON/GEOMC/LATD,LATM,LATS,PIXELS,NL,KOUNT,NSGM

Call FASTGE(IBUF)
```

Description of parameters

LATD	= degrees of latitude, NW corner of frame
LATM	= minutes of latitude
LATS	= seconds of latitude
PIXELS	= original number of pixels per scan line
NO	= number of lines in the image
KOUNT	= program call counter
NSGM	= corrected number of pixels per line
IBUF	= buffer containing line to be corrected

All parameters are integers greater than or equal to zero.

Subroutines and function subprograms required

FILBUF

COS

Method

Skewing of the image is performed by offsetting the scan lines an integer amount determined from the latitude. Aspect ratio is corrected by deleting pixels from the line.

REMLOG

PROGRAM remlog

A log of REMAPP operations can be maintained by using the following subroutine calls in REMAPP programs. All REMAPP programs should call remlog$header("name").

Calls to remlog$header and remlog$open must be followed by a call to $close, but may have any number of write(6,n), print, or call ioa_ statements in between.

For example:

```
call remlog$header("OPTRIN")

print, "this is going to be fun"

call remlog$close

call remlog$open

write(6,99)

99 format (1x, "this is going to fun")

call remlog$close
```

All output from these calls will be stored in segment remapp_log.

CHAPTER VII

OPTRONICS TAPE FORMAT

This chapter describes the tape format required by the Optronics P-1700 photomation. The tape must be 9-track with a density of 800 BPI. A tape may contain as many data files as space will allow. Each file consists of one header record followed by an arbitrary number of data records and is terminated by an end-of-file mark.

The header record contains information useful to the Optronics operator but only the header identifier and the file identifier are used by the Optronics system. This information is written as ASCII characters, right-justified in 8-bit bytes. The Optronics system, however, uses only the right-most 6 bits.

The header record is 54 bytes long and is formatted as follows:

Byte 1. Header Identifier.

All bits are set to one to indicate a header. Otherwise, this record is taken as a data record.

Bytes 2-9. Tape File Identifier.

Eight characters to be used as file identification.

Bytes 10-15. Date.

Six characters for the date that the file was created, two characters each for month, day, and year.

Bytes 16-19. Time.

Four characters for the time that the file was created, two characters each for hours and minutes.

Bytes 20-23. Aperture Setting.

Four characters for the aperture setting of the Optronics expressed in micrometers times ten. This is either 0250, 0500, 0100.

Bytes 24-27. Raster Setting.

Four characters for the raster setting of the Optronics expressed in micrometers times ten. This is either 0250, 0500, or 1000.

Bytes 28-31. X-START Location.

Four characters expressing the X-START location of the image in millimeters. The first character is always ASCII zero.

Bytes 32-35. X-END Location.

Four characters expressing the X-END location of the image in millimeters. The first character is always ASCII zero.

Bytes 36-39. Y-START Location.

The same for Y-START as for X-START.

Bytes 40-43. Y-END Location.

The same for Y-END as for X-END.

Byte 44. Transfer.

The character "+" for a direct Optronics transfer (i.e., data value 0 is output as 0), and "-" for complement Optronics transfer (i.e., data value 0 is output as 255).

Bytes 45-54. Binary Zeroes.

The data records consist of image position information and density data values. A record may contain a maximum of 9144 data values and all data records will usually be of the same length. The data is formatted in 8-bit bytes as follows:

Bytes 1-3. X-Position Data.

The rightmost 18 bits (the left 6 bits are ignored) represent an unsigned integer offset in the X direction from the beginning scan line. The first scan line has an offset of one.

Bytes 4-6. Y-Position Data.

The rightmost 16 bits (the left 8 bits are ignored) represent an unsigned integer offset in the Y direction from the beginning of that scan line. This will be zero for scan lines with less than 9145 data points.

Bytes 7 and beyond. Density Data.

Data are written as binary 8-bit bytes with values from 0 to 255.