GEOLOGICAL SURVEY OF CANADA

OPEN FILE 1767

COMPUTER PROGRAM FOR CONVERTING ARC-NODE VECTOR DATA TO RASTER FORMAT

M. Steneker and G.F. Bonham-Carter

Mineral Resources Division

1988

GEOLOGICAL SURVEY OF CANADA

OPEN FILE 1767

COMPUTER PROGRAM FOR CONVERTING ARC-NODE VECTOR DATA TO RASTER FORMAT[1]

M. Steneker and G.F. Bonham-Carter

Mineral Resources Division

1 Project partly funded by the Canada-Nova Scotia Mineral Development Agreement 1984-1989

1988

ABSTRACT

This report describes a computer program written in FORTRAN-77 for the VAX 11-780, which is used for producing raster images of thematic maps. Input to the program consists of data describing the boundaries of map polygons in arc-node format, produced during the digitizing process. The principal source of this data is the Canada Lands Data System (CLDS), where maps are digitized by raster scanning, and vector files produced after various editing and transformation steps. Output from the program consists of raster files in a variety of formats. In this form, thematic map data can be readily integrated with remote sensing images, and with regional geophysical and geochemical data. A second program is provided which puts raster data into a run-length encoded archive format (ACSII), used by CLDS and other organizations for map exchange. Images and associated attribute files conforming to this format can be readily moved from one computer to another, and can be directly input to the TYDAC SPANS system. This is a microcomputer-based spatial analysis system used for display and manipulation of digitized maps recently purchased by GSC.

The source listings of these programs are of general use for those working in digital cartography and data integration of maps. The programs are available on diskette in ASCII or the executable program may be used directly by users of the EMR VAX system.

INTRODUCTION

From a cartographic viewpoint, geological maps in part comprise a set of interlocking polygons, each polygon belonging to a geological unit or theme. Digital capture of polygons on geological maps can be carried out in a variety of ways, normally by raster scanning or manual line following methods, and in general, raster scanning is the most efficient for complex maps, Bonham-Carter

et al. (1985). Representation of polygon in digital form can either be in raster or vector mode. In raster mode, each pixel in the raster is labelled by a polygon number which acts as a pointer to a table of attributes – rock type, age, formation name and others. In vector mode, each polygon is defined by one or more bounding arcs, terminating at nodes, and each arc, or line segment, is described by two or more (X-Y) coordinate pairs, Figure 1. Associated with each arc are labels which refer to the polygon on the right and on the left of the arc. Arc-node vectors may be generated from manual digitizers, or they may be derived by digital processing of raster-scanned data, Crain (1984).

Any particular polygon on a map is thus identified by finding those arcs with labels pointing to its polygon number. Except for polygons intersecting the map boundary, every arc has separate polygons to right and left, so each arc does double duty describing the common boundary between adjacent polygons. This is efficient for data storage, but requires a list of arc pointers associated with each polygon for rapid access of the arcs bounding a polygon. This would be important, for example, for drawing a polygon on a video display device using a polygon-fill algorithm.

Alternatively, it is often desirable to generate a raster file from arc-node data. The raster data can be in expanded form with every pixel described by polygon number, or in run-length encoded form wherein adjacent pixels on a line belonging to the same polygon are described by run-length and polygon label.

For several years, the GSC has contracted the raster scanning of a limited number of maps to Canada Lands Data Systems (CLDS), a unit of Environment Canada in Hull. These maps have been used in data integration research projects, for example Aronof *et al.* (1986). Two digital formats are obtained from CLDS, one using the vector arc-node format, which in CLDS terminology is 'Pairwise Contact' or PC data. The other format is run-length encoded raster, with a predefined pixel size and raster origin. Ellwood (personal communication) has developed software

written in FORTRAN on the departmental CYBER for reading these files, and for producing Applicon plots which have been used extensively for poster displays, *eg.*, Ellwood *et al.* (1987). For these digitally-produced displays, the raster files were used in the main, with the vector PC data only used to draw polygon boundaries, using Applicon software to convert the vector lines to raster form.

There is, however, a need to generate raster files from arc-node data locally, without having to rely on CLDS processing capability. This arises because:

1. Raster files from CLDS are confined to one pixel size and raster origin. This if windowing is needed, zooming to a small area at a larger scale, the pixel size is too coarse, yet the full precision of the vector data would allow for much greater resolution.
2. Change of projection is not readily feasible with raster format, yet in vector form this is straightforward, although computationally demanding.
3. the handling of coastline information from two co-registered maps poses a problem using the CLDS system as will be described further below.

As no commercial package was readily available, a program (VEC2RAS) was written in FORTRAN-77 to convert arc-node data to raster format for the VAX 11-780. VEC2RAS will accept data in PC format from CLDS, and with minor modifications other arc-node formats, such as INTERGRAPH files or arc-node data generated from digitizing tables. The program allows the user to choose a pixel size, origin, and whether the raster file records are to correspond to lines parallel to N-S or E-W. Output can be in run-length encoded, or expanded formats, using binary or ASCII representation. The source listing of VEC2RAS is in Appendix 1.

A further option is available for producing a simplified vector file, useful for drawing polygon boundaries on an image. The co-ordinates used for the vector file are the pixel coordinates of the selected grid (as opposed to UTM coordinates in metres normally used on the CLDS vector files). With two exceptions, each arc that

occurs on the original file appears in the new file, but with many fewer coordinate pairs, consistent with the resolution of the raster file. The first exception is for the case of a window smaller than the original map, when clipping retains only those lines, or part-lines, occurring in the window. The second exception is for two-classification PC data (see below), when boundaries are kept for one selected classification only.

Compact vector files, still retaining the polygon-left, polygon-right labels, are invaluable in data integration studies. For example, the boundaries of selected geological units or a coastline can be superimposed on a coloured image of geochemical or geophysical data.

A computer program to select and plot polygon boundaries from the simplied vector file is straightforward to write, and efficient to execute because the simplified vector file is much smaller than the full CLDS PC data file. Further, because the coordinates are pixel coordinates, correctly registering vector polygon outlines with a raster is easy, and vector polygon boundaries will always correctly fit the rasterized polygon fill. An example of using the boundary option is shown in the the example dialogue, Appendix 2.

A second program, CLDSARC, is shown in Appendix 3. This takes formatted run-length encoded raster files and writes them out in an archive format, used by CLDS and by GSC for exchange of thematic map data. Because the archive format uses ASCII representation, and 80-column records, it is bulky but ideal for exchange between machines. The specifications for the archive format are shown in Appendix 4. Archive format is not included directly as an option in VEC2RAS because it would often require two tape drives simultaneously, one for the CLDS input tape, one for the archived output.

A separate program is also available, not published here, for generating a list of arc-pointers for each polygon in the arc-node table, and ordering arcs to form a

continuous boundary, which is useful for displaying polygons with polygon fill algorithms. This program is also in FORTRAN-77, for the VAX 11-780 using VMS.

METHOD

The algorithm on which VEC2RAS is based is described by Foley and Van Damm (1982). Each arc or line segment is defined by n (x, y) coordinate pairs, and labelled according to the polygon on each side. The program reads the vector data for each arc, and finds the intersection of the arc with every row of a grid (with predefined origin and pixel size), keeping track of the polygon to right and left of the intersection. This operation is repeated for every arc, whereupon the intersections for each grid row are sorted by grid column. This leads easily to a run-length encoded description of the grid row, one (run length, run polygon) per intersection.

The principal steps in the calculation are:

1. Define the pixel size, grid origin, and grid dimensions.
2. For every arc in the vector file, find the coordinates at which the arc intersects each grid line parallel to the x-axis, and store these x-coordinates rounded to the nearest grid line parallel to the y-axis. The X-axis is normally orientated E-W, unless the 90° rotation option is used (Figure 2).
3. If two intersections fall on the same grid-line intersection, retain only the intersection with largest x-coordinate before rounding. This ensures that along every line parallel to x, only one polygon intersection per pixel is possible.
4. If two or more intersections on any line parallel to the x-axis intersect at exactly the same x-coordinate, the grid line is moved a small distance and the two arc intersections recalculated. Then the new intersection with the largest x-coordinate is chosen as in 3.

5. After calculating all intersections for all arcs, each line parallel to the x-axis is subdivided into intervals (or runs) between intersections. These intervals, to the nearest pixel, and their polygon labels, comprise the output in run-length encoded form, line-by-line.

INPUT FORMAT

CLDS tapes with PC data are in the format shown in Tables 1 and 2. Both files are in ASCII. Field widths may vary depending on the job. The program actually uses the second file first, because it contains the lookup table relating 'face' number to polygon number. Thus, the first file is skipped, the second file read, and then the tape rewound for processing the pairwise contact data.

In some situations, the 'face' number is related to more than one polygon number. This occurs when two maps have been merged to produce a single vector file. For this case each arc has a 'face' to right and to left; each face is related to two polygons, one from the first map, and one from the second map. CLDS (and other GIS's) can merge many maps into a single vector file in this manner. The VEC2RAS program is at present limited to two merged maps (also referred to a two classifications).

For situations where two maps, such as surficial geology and bedrock geology have both been raster scanned, and both contain a coastline which should be exactly co-registered, CLDS uses the following method to ensure that no under- or overlap exists between the coastlines. This task is virtually impossible when scribing the two coastlines on separate documents and then scanning them independently:

1. The accurate coastline is put on to the first map.
2. A line parallel to the coast, but everywhere clearly offshore, is put on to the second map, and polygon boundaries which intersect the coastline extended to meet the 'false' coastline.

3. The two maps are raster scanned, edited, raster-to-vector converted, and the polygons classified as normal.
4. The two vector files are merged, so the combined file has polygons, 'faces', that are classified according to both maps.

When extracting raster files for each geology map, the polygons corresponding to SEA on the first map are used to clip the polygons on the second map back to the correct coastline. In this way, the accurate coastline is used for both maps. VEC2RAS will enquire from the user whether more than one polygon classification is being used; if it is, the polygon numbers for SEA must be entered for both map classifications.

The conventions for defining the coordinate origin and grid arrangement are shown in Figure 2. Note that the lines of the raster file, each line starting on a new record, can be made a) to run W to E, starting in the NW corner; or b) to run S to N starting in the SW corner by using the 90° rotation option.

The limitations of the program are: no more than 1024 raster lines, and no more than 1024 theme changes per line. The grid may be much smaller than the maximum and minimum coordinates found in the input file, *i.e.*, windows with only part of the input are possible – 'zoom in'. Alternatively the grid may be larger than the input data, with padding round the outside – 'zoom out'. If an image with more than 1024 lines are required, this is possible using multiple passes, changing the origin successively but keeping the same pixel size. Concentration of the resulting raster files must be carried out separately, but with judicious sequencing of the origin is not difficult. Normally the limit of 1024 changes in theme per line is not a practical restriction.

OUTPUT

File formats in the present version include:

1. Expanded raster. Formatted file.

   Record 1. NROWS, NCOLS (freefield Integer format).

   Record 2. Blank.

   Record 3. Image data from first row (N*I4) format.

   Record 4 to (3 + NROWS). Repeat of Record 3.

2. Morpholog (expanded raster). Unformatted file.

   The name Morpholog refers to an image analysis program, commercially available and in use at GSC.

   Record 1. NCOLS, NROWS (Integer *4).

   Record 2 to (NROWS + 1). Image data, one record per row (Integer *2). If the image is larger than 512 x 512, a warning is issued and decimation is offered as an option.

3. Rasterized boundary file. Unformatted (Morpholog) file. Same as output 2 except boundary pixels are 1's, with 0's elsewhere.

4. Run-length encoded file. Formatted.

   Record 1. "__H", Descriptive comment line. The "__H" must be in cols. 1 – 2 (A2, A60).

   Record 2. "__H", NROWS, NCOLS (A2, I8, I7).

   Record 3. "__H", EASTING, NORTHING of origin (A2, I13, I12).

   Record 4. "__H", EASTING, NORTHING of top-right corner (A2, IB, I12).

   Record 5. "__H", EASTING SCALE, NORTHING SCALE (A2, F12.5, F15.5).

   Expressed in <reciprocal form, *e.g.*, 1:1,000 is 1,000.

   Record 6. "__H", UTM ZONE (A2, F5.0)

   Record 7. "__H", Integer number usually set to 1 unless the second classification of a double classification image is used (= 2). (A2, I4).

Record 8. NROWS, NCOLS (I6, I7).

Record 9. POLYGON, PIXEL RUN (I6, I7).

Record 10 to end. SAME as 10.

5. This file contains arc-node data, but restricted to a window (if selected) and simplified into integer pixel coordinates, thus greatly reducing file volume. The pixel origin (0, 0) is at the same location as the UTM origin in the header (Record 3).

   Record 1 to 7. HEADER INFO (same as 1 to 7 in file format 4).

   Record 8. LEFT POLY, RIGHT POLY, A* (I4, I5, I4) where A* = # of pts used to define this segment.

   Record 9 to (A* + 8) X-COORD, Y-COORD (2I5).

   Record (A* + 9) LEFT POLY, RIGHT POLY, B* (I4, I5, I4) where B* = # of pts used to define this segment

   Record (A* + 10) to (A* + B* + 9) X-COORD, Y-COORD (2I5).

6. Run-length encoded. Unformatted.

   Same as 4 except unformatted file.

SAMPLE RUN

The dialogue in Appendix 2 was dumped to a line printer to show a typical session. The responses of the user have been added in a different typeface. A CLDS tape was first mounted using the command: MOUNT/FOREIGN/BLOCKSIZE = 2818 MT: CG1156 TAPE. The blocksize and tape label was supplied on the CLDS tape description, see Table 1. In this case the local file name for use with the program is TAPE. The program is executed by typing RON DUB1: [BCVAX.EXEC] VEC2RAS.

PROGRAM PERFORMANCE

A number of factors affect performance, but typically the program will produce a raster image from PC data at the rate of about 1000 segments (arcs) per minute. Typical maps will have 1000 to 10000 arcs, so interactive use of the program is quite tolerable.

ACKNOWLEDGEMENTS

We thank Casey van der Grient for some early work on this problem, and Danny Wright for his advice as a user.

REFERENCES

Aronof, A., Goodfellow, W.D., Bonham-Carter, G.F., and Ellwood, D.J., 1986, Integration of surficial geochemistry and Landsat imagery to discover skarn tungsten deposits using image analysis techniques, Proc. IGARSS' 86 Symposium, Zurich, 8 – 11 Sept., 1986, p. 513-520.

Bonham-Carter, G.F., Ellwood, D.J., Crain, I.K., and Scantland, J.L., 1985, Raster scanning techniques for the capture, display and analysis of geological maps, Canada Lands Data Systems, Lands Directorate, Environment Canada, Report R003210, 12 p.

Crain, I.K., 1984, A comparison of raster scanning and manual digitizing, Canada Lands Systems, Lands Directorate, Environment Canada, Report R001300.

Ellwood, D.J., Bonham-Carter, G.F., and Goodfellow, W.D., 1986, An automated procedure for catchment basin analysis of stream geochemical data: Nahanni River Map Area, Yukon and Northwest Territories, Geological Survey of Canada, Paper 85-26.

Ellwood, D.J., Bonham-Carter, G.F., and Rogers, P.J., 1987, Integration and display of surficial geochemical data using catchment basins, Cobequid Highlands, Nova Scotia, GSC Forum, Poster.

Foley, J.D. and Van Dam, A., Fundamentals of computer graphics, Addison-Wesley, 664 p.

FIGURE CAPTIONS

1. Diagram of arc-node vector representation of polygon data.

2. Conventions for definig grid origin and placement.

3. Black and white reproduction of a coloured image produced on a Tektronics 4696 ink-jet plotter, approximately 7 inches square. The image was initially created on a Chromatics 7900 at a 1024 x 1024 resolution. The raster image in B shows the surficial geology of Eastern Nova Scotia. The inset portion in A shows the same image but rotated clockwise 90° and using a smaller scale. This example is described in the sample dialogue (Appendix 2) and the CLDS tape description is shown in Tables 1 and 2.

4. Black and white reproduction of a colour image showing polygon boundaries of the same map in Figure 3. The simplified vector file used to create this image was created as an option in VEC2RAS. Selected boundaries, such as 'coastline' or 'all granite contacts', can be plotted using this file which uses integer pixel coordinates and is therefore very compact. Images with solid themes and selected boundaries provide an effective means to compare two maps, for example bedrock geology with a catchment basin geochemical map.

TABLE CAPTIONS

1. File organization for CLDS PC-data.
2. File organization for CLDS Descriptive Dataset.

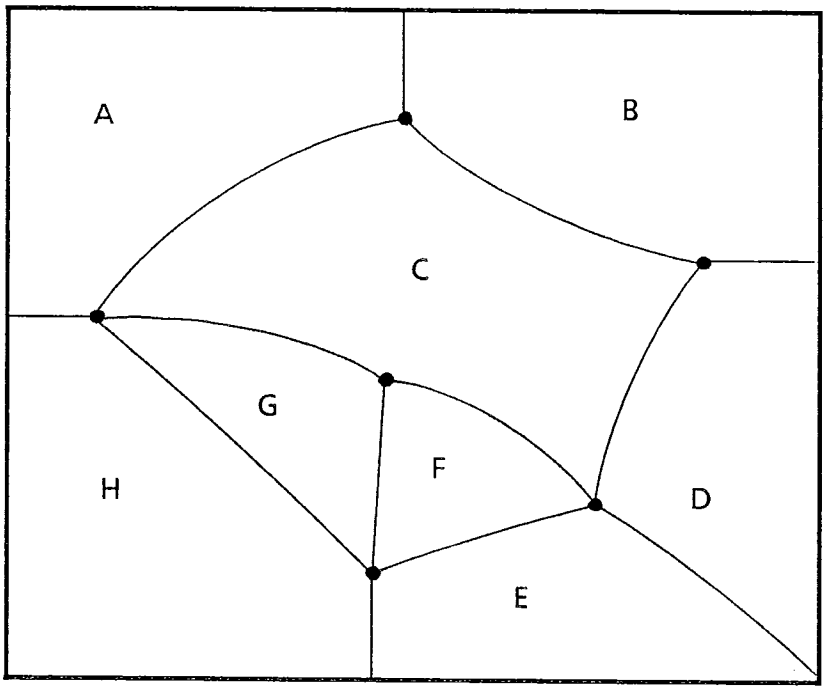
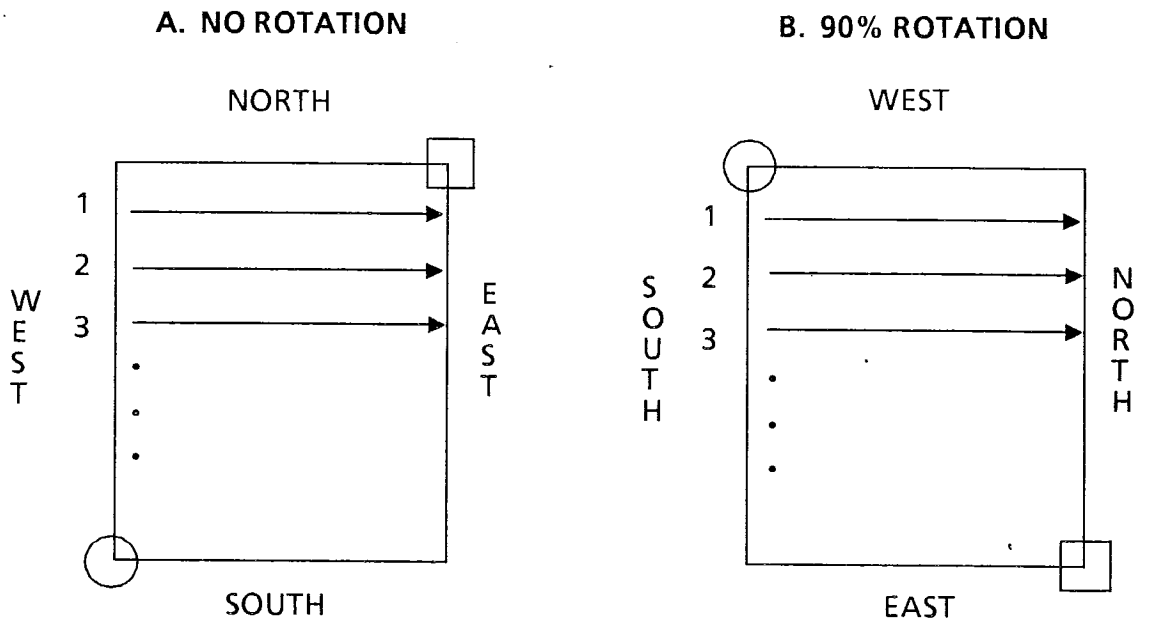


Figure 1. Eight polygons labelled A to H. Each polygon is bounded by one or more line segments, or 'arcs', which intersect at 'nodes'.



= co-ordinates of the SW corner of the grid file, in the same units as the input vectors, usually UTM metres.

= co-ordinates of the NE corner of the grid file, also in the same units as the input vectors.

Pixel size: defined in units of input vectors, normally metres, for E-W and N-S.

Grid size: defined by number of pixels E-W and N-S.

The numbered arrows indicate the sequence and direction of the 'scan' lines, each line held as a separate record in the output file.

Figure 2. Conventions for defining grid origin and placement. Three alternative options are possible for specifying the size and resolution of the raster grid, see the example dialogue

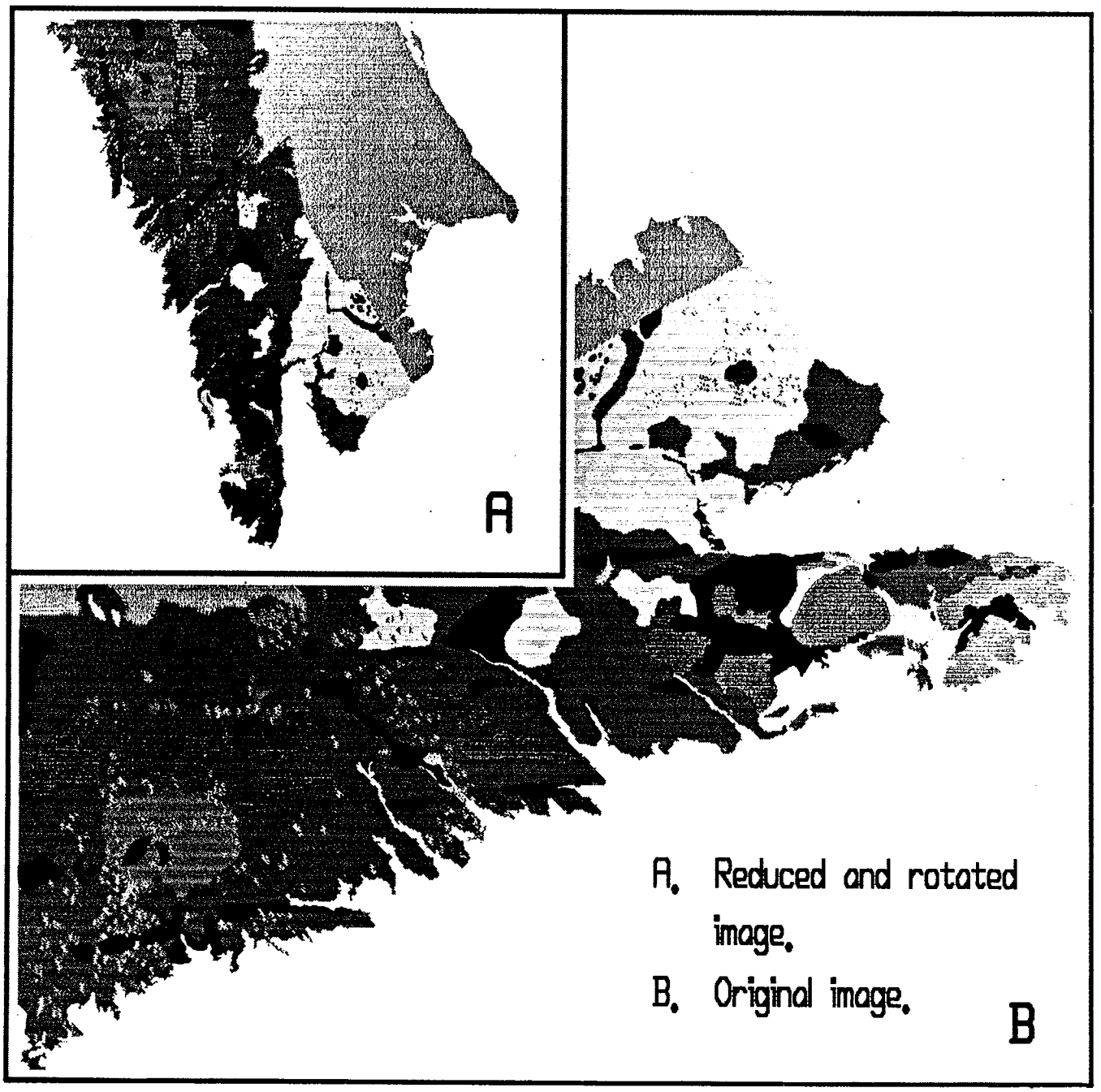


FIGURE 3

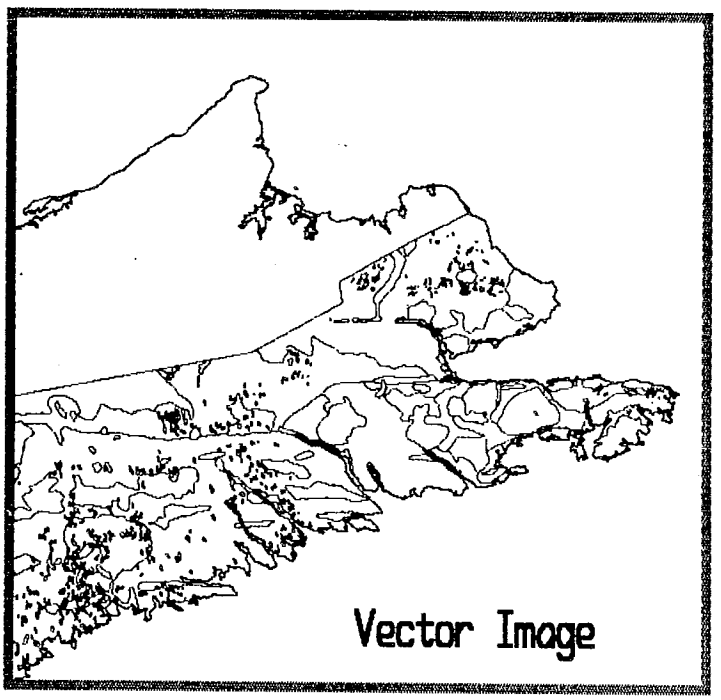


FIGURE 4

TABLE 1

PAIRWISE CONTACT (PC DATA) (UTM): IDS **PROJECT:** 71-2103

Image Data Set **DATE :** May 5th/87

DESCRIPTION:

Segment file for coverage 9373

FILE STRUCTURE:

Field No.	Field Name	Field Type	Field Length		Comments
1	FACR	picture '999999'	6 bytes		
2	FACL	picture '999V99'	6 bytes		
3	ZONE	picture '99'	2 bytes		
4	NOVERT	picture '9999'	4 bytes		
5	XUTM	picture '999999'	6 bytes	]	This pair
6	YUTM	picture '99999999'	8 bytes	]	repeated
				]	NOVERT times

WHERE:

FACR - POLYGON number on the right side of the segment. All segments required to define a POLYGON (an area) have the same POLYGON number.
FACL - Same as FACR except polygon is on the left side of the segment.
ZONE - UTM zone.
NOVERT - The number of points along the segment.
XUTM, YUTM - UTM eastings and northings in meters.

(FACR or FACL = 0 defines areas outside the map).

FILE STATISTICS:

2879 records 53995 verticies 8,113,022 bytes

Record Format	FB	Recording Mode	ASCII
Logical Record Length (LRECL)	2818	Density	1600 Bpi.NL
Block Size (BLKSIZE)	2818	Track 9	
Number of Records	2879		
Tape Volume Serial Number	CG1211	File	1
Data Set Name (DSN)	CGI01.CGIS.COV9373.PC9373		

INTERNAL USE
Coverage 9373 Date May 5/87 By André Daigneault
Backup NIL

PLEASE COPY AND RETURN TAPE

CLDS Doc/CLDS 500-530

TABLE 2

DESCRIPTIVE DATA FOR COVERAGE (UTM): DDS PROJECT: 471-2103

Descriptive Data Set DATE : May 5, 1987

FILE STRUCTURE:

Field No.	Field Name	Field Type	Field Length
1	FACE #	picture '999999'	6 bytes
2	AREA	picture '99999999V99'	10 bytes
3	XUTM	picture '9999999'	7 bytes
4	YUTM	picture '99999999'	8 bytes
5	ZONE	picture '99' V for implied decimal only	2 bytes
6	POLY_NO	picture '9999'	4 bytes

WHERE:

FACE # - CLDS polygon identifier
AREA - Polygon area (in hectares)
XUTM - Centroid X
YUTM - Centorid Y
ZONE - Zone number
POLY_NO - Polygon number

Record Format	FB	Recording Mode	ASCII
Logical Record Length (LRECL)	37	Density 1600	Bpi.NL
Block Size (BLKSIZE)	37	Track 9	
Number of Records	854		
Tape Volume Serial Number	CG1211	File 2	
Data Set Name (DSN)	CGI01.CGIS.COV9373.DESC9373		

INTERNAL USE
Coverage 9373 Date May 5th/87 By André Daigneault
Backup NIL

PLEASE COPY AND RETURN TAPE CLDS Doc/CLDS 500-530

```
      PROGRAM Vector_To_Raster
C_____________________________________________________________________
C
C     Vector_To_Raster program
C---------------------------------------------------------------------
C     Written by: M. Steneker
C
C        This program takes digitized line segments on tape from
C     Environment Canada's CLDS (as well as other places which is
C     the reasoning for the OTHER clause), and creates  a polygon table.
C     The tape consists of one line segment / record and
C     which has a left and right polygon attributed to it.
C        The program creates depending on what is required either a
C     raster table of the image, or a run-length encoded file of the
C     image.  This program also has the capability of windowing part
C     of the data vector file, and allows for magnification or shrinkage
C     of the window.
C
C---------------------------------------------------------------------
C                     Program variables
C---------------------------------------------------------------------
C
      PARAMETER  n_poly=1024,n_segments=10000,n_polyseg=256,
     +           n_vertices=300000

      BYTE  inbuff(4096)

      INTEGER*4 record_len, xy_buf(1024,2),HEX_30,
     +          x_coord(n_polyseg), y_coord(n_polyseg), ydes,
     +          num_coords, ystart, xstart, a, b, c,
     +          max_segments,max_coords, max_seg(n_poly), arrow,
     +          head, tail, max_y, m1, m2, rightside, area, xdes,
     +          start, Pnum, focus, xstop, ystop

      INTEGER*2 next(0:n_poly, 0:n_poly), null, FaceToPoly(2000,2),
     +          end(0:n_poly), out(1:n_poly),
     +                         out1(1:n_poly),
     +                         out2(1:n_poly),
     +          dump(0:n_poly)

      LOGICAL migsif, pcdata, other, renew, flip, traverse, clipping,
     +        ballpark, dump_out, morph, bound, error, Cor, Siz, Num,
     +        face, is_tape, rle, crle, boundry, first, passone

      REAL*4 exp(6),exp1(8),
     +           true_x,     slope, x1, x2, y1,     y2,
     +           true_xprim, slopeprim,     y1prim, y2prim,
     +       yint, xint, rsave(0:n_poly, 0:n_poly),
     +       ixmin, ixmax, iymin, iymax

      CHARACTER fname*24, ffname*24, boundfile*24, fn*85, reply*1,
     +          label*60

      INTEGER*2 save1(0:n_poly, 0:n_poly), pointer, xnum, ynum,
     +          second_last, distance, last, reduce, numofcoords,
     +          left, right, oldleft, oldright, map_y, n, nprim

      INTEGER*4 save2(0:n_poly, 0:n_poly)

      DATA CHAIN/1/, HEX_30/'30'X/, exp/100000.0,10000.0,1000.0,100.0,
     +     10.0,1.0/,exp1/10000000.0,1000000.0,100000.0,10000.0,1000.0,
     +     100.0,10.0,1.0/,vertice_pointer/1/,Xo/99999999/,Xn/-999999/,
     +     Yo/99999999/,Yn/-99999999/,head/1/,tail/2/,null/-1/

C
C   /*   Request the data file and the coordinates     */
C
      ixmin=9999999.
      iymin=9999999.
      ixmax=0.
      iymax=0.

      flip = .false.

              PRINT*
      PRINT*,'            Vector to Raster File Program '
              Print*
      Print 1100
1100  Format (' Do you require a brief introduction ? y/N ',$)
      Read (*,'(a1)') fn
      If (fn.eq.'y'.or.fn.eq.'Y') then
```

```
          Call Help (1)
        End If
        fn = ' '
                PRINT *

        PRINT 1000
1000    FORMAT (' The name of the input vector file : ',$)
        READ(*,'(A32)') ffname
1022            PRINT *

        PRINT 1342
1342    FORMAT (' Is the input vector file on tape ? (Y/n) : ',$)
        READ (*,'(a1)') fn
                PRINT *

        If (fn.eq.'n' .or. fn.eq.'N') then
            Is_tape = .FALSE.
        else
            Is_tape = .TRUE.
        endif

        PRINT 1002
1002    FORMAT(' The file type or structure : ',
     +          '(m)igsif, (P)cdata, (o)ther (?)Help ',$)
        READ(*,'(A1)') fn
                PRINT *

        If (fn.eq.'?') THEN
            Call Help (2)
            GoTo 1022
        End If

        passone = .FALSE.
        migsif  = .FALSE.
        pcdata  = .FALSE.
        other   = .FALSE.

        IF (fn.EQ.'M'.OR.fn.EQ.'m') THEN
            migsif = .TRUE.
        ELSE IF (fn.EQ.'O'.OR.fn.EQ.'o') THEN
            other = .TRUE.
        ELSE
            pcdata = .TRUE.
        ENDIF

        If (migsif.or.pcdata.or.other) Then
             flip = .false.
        Else
             GoTo 1022
        End if

        IF (migsif.or.pcdata) then
             PRINT 1003
1003         FORMAT(' The logical record length for image data set : ',$)
             READ(*,*) l_recl
        ENDIF
                PRINT *

        WRITE(*,56)
56      FORMAT (' Many CLDS files require a second file (a descriptive',
     +' data set) which links',/,' the face value to the true polygon',
     +' number. ',/,' Do you wish such linking ? (y/N) : ',$)
        Pnum=0
        GOTO 58
555        IF (Is_tape) THEN
             PRINT*
             PRINT*,'              One moment ..... Tape Rewinding'
             REWIND (10)
             REWIND (10)
             PRINT*, char(7)
           ENDIF
556        CLOSE (10)

            WRITE(*,59)
59          FORMAT (' Do you wish to read or re-read the descriptive',/,
     +      ' data set to allow for new face value linking ? (y/N) : ',$)
            second = .true.
58      READ (*,'(a1)') fn
        focus = 0
        Face = ((fn .eq. 'Y') .or. (fn .eq. 'y'))
                PRINT*
        IF (Face) THEN
```

```
      CALL FaceReader (FaceToPoly, migsif, pcdata, other, Pnum,
     +                 is_tape, ffname)
      ENDIF
            PRINT*

      IF (Pnum .eq. 1) THEN
           focus = 1
      ELSEIF (Pnum .eq. 2) THEN

          PRINT 577
577    FORMAT(' For the two classification sets which do you wish to ',
     +          'focus on ?',/,' (n)one   (F)IRST   (s)econd : ',
     +          $)
          READ(*,'(a1)') fn
          IF ((fn .eq. 'n') .or. (fn .eq. 'N')) THEN
              focus = 0
          ELSE IF ((fn .eq. 's') .or. (fn .eq. 'S')) THEN
              focus = 2
          ELSE
              focus = 1
          ENDIF
      ENDIF

      Face = focus.ne.0
            PRINT*

      IF (migsif.or.pcdata) then
           OPEN(10,ERR=333,FILE=ffname,FORM='unformatted',
     +          RECORDTYPE='VARIABLE', STATUS='old')
           GO TO 444
333        OPEN(10, FILE=ffname, FORM='unformatted',
     +          STATUS='old')
444        CONTINUE
      ELSE
           OPEN(10,FILE=ffname,FORM='formatted',STATUS='old')
      END IF

      WRITE (*,*)' What are the UTM coords of S-W corner for the OUTPUT'
      WRITE (*,*)' image ? '
      IF (passone) write (*,*) '                          (default is ',
     +          ystart,')'
      PRINT 1009
 1009 FORMAT ('            Northing : ',$)

         ystart = NumberReader (ystart)

      IF (passone) write (*,*) '                          (default is ',
     +          xstart,')'
      PRINT 1099
1099  FORMAT ('            Easting  : ',$)

         xstart = NumberReader (xstart)

      Cor = .true.
      Siz = Cor
      Num = Siz

1888  print*
      print*, ' There are three possible ways to define the output',
     +        ' image. Default is 1.'
      print*
      print*, '         1. The size for each pixel N-S, and E-W.'
      print*, '            and the number of pixels in N-S, and E-W'
      print*
      print*, '                ie (50m X 60m, 256 X 512)'
      print*
      print*, '         2. The number of pixels in N-S, and E-W'
      print*, '            and the N-E corner coordinates.'
      print*
      print*, '                ie (512 X 1024, ystop, xstop)'
      print*
      print*, '         3. The size of each pixel N-S, and E-W '
      print*, '            and the N-E corner coordinates.'
      print*
      print*, '                ie (100m X 30m, ystop, xstop)'
      print*
      print 1800
1800  format (' ',$,'    Choice : ',$)
      read (*,'(a1)') fn

      IF (fn.eq.'1'.or.fn.eq.' ') THEN
           Cor = .false.
```

```
      ELSE IF (fn .eq. '2') THEN
           Siz = .false.
      ELSE IF (fn .eq. '3') THEN
           Num = .false.
      ELSE
           Goto 1888
      END IF

      IF (Siz) Then
        print *
        print 1007
1007    FORMAT (' The size of each output pixel',/,
     +      '     e.g.,  1 output pixel = 5.2 input units',/,
     +      '     Along Northing : '$)

          READ(*,*), yint

        PRINT 1777
1777  FORMAT ('     Along Easting  : '$)

          READ(*,*), xint

      END IF

      IF (Num) THEN

        WRITE (*,*)
1066    PRINT 1006
1006    FORMAT (' Number of pixels in the output image',
     +           '      i.e.,     Output Image Size',/,
     +           '     Along Northing : '$)

          READ(*,*), ynum

        PRINT 1778
1778    FORMAT ('     Along Easting  : '$)

          READ(*,*), xnum

        IF (ynum .gt. 1024) THEN
            print*
            print*,' Sorry maximum allowable N-S value is 1024,',
     +             ' please re-enter.'
            GoTo 1066
        End If
      END IF

      IF (Cor) THEN

            print *
            print*,'What are the UTM coords of N-E corner for the ',
     +             'output image ? '
            IF (passone) print*,'                       (default is ',
     +                          ystop,')'
            print 1070
1070  FORMAT ('            Northing : ',$)

         ystop = NumberReader (ystop)

            IF (passone) print*,'                       (default is ',
     +                          xstop,')'
      PRINT 1071
1071  FORMAT ('             Easting : ',$)

         xstop = NumberReader (xstop)

      END IF
      passone = .TRUE.

      IF (num .and. cor) THEN
          xint = (xstop - xstart)/xnum
          yint = (ystop - ystart)/ynum
      ELSE IF (num .and. siz) THEN
          xstop = xstart + xnum * xint
          ystop = ystart + ynum * yint
      ELSE
          xnum = (xstop - xstart)/xint
          ynum = (ystop - ystart)/yint
      END IF

      PRINT *
1012  PRINT 1001
```

```
1001  FORMAT (' Do you wish to rotate the image 90 degrees clockwise ?',
     +        ' y/N/? ',$)
      READ(*,'(a1)') reply
        If (reply.eq.'?') Then
           Call Help (3)
           GoTo 1012
        End If
      If (reply.eq.'y'.or.reply.eq.'Y') flip = .true.
      write (*,*)

      IF (flip) THEN
         x1 = ystart
         ystart = xstart
         xstart = x1
         x1 = yint
         yint = xint
         xint = x1
         x1 = ystop
         ystop = xstop
         xstop = x1
         a = ynum
         ynum = xnum
         xnum = a
       END IF

        IF (ynum .gt. 1024) THEN
            print*
            print*,' Sorry maximum allowable N-S value is 1024,',
     +              ' please re-enter.'
            GoTo 1066
        End If

      sliver = (yint*1.0) / 100

                clipping = .TRUE.
C
C     Just a quick interactive check to see if all is correct before running
C
      Print*
      Print*,'CONFIRMATION of OUTPUT DIMENSIONS & FACE MAPPINGS.'
      Print*
      Print*,'1. S-W corner has coordinates (', ystart,',', xstart,')'
      Print*,'2. N-E corner has coordinates (', ystop ,',', xstop, ')'
      Print*,'3. Number of pixels in E-W directions =', xnum
      Print*,'4. Number of pixels in N-S directions =', ynum
      Print*,'5. Size (in input scale) for 1 E-W output pixel =',xint
      Print*,'6. Size (in input scale) for 1 N-S output pixel =',yint
      IF (face) THEN
        write(*,126) focus
126     format(' 7. Face Value   is sent to   Polygon Set ',i2)
      ELSE
        Print*,'7. Face Value   is not sent to any   Polygon Set.'
      ENDIF
      write (*,*)
      WRITE(*,121)
 121  Format ('           Is this all correct ? Y/n ',$)
      Read (*,'(a1)') fn
      WRITE (*,*)
      IF (fn .eq. 'n' .OR. fn .eq. 'N') GOTO 556
         PRINT*

         WRITE (*,222)
222      FORMAT (' Output formats ',//,
     +           '          -  Vector Boundaries    (h)    ',
     + ' (formatted)',/,
     +           '          -  Expanded Raster,            ',
     + ' (formatted)',/,
     +           '          -  Morpholog,                  ',
     + ' (unformatted)',/,
     +           '          -  Morpholog Boundaries Only,  ',
     + ' (unformatted)',/,
     +           '          -  Run Length Encoded,  (h)    ',
     + ' (formatted)',/,
     +           '          -  Run Length Encoded,         ',
     + ' (unformatted)',/,
     +           ' any or all of them. ',/,
     +           '                                ',
     +  'NOTE : (h) - Header Records Included.',//)
         write (*,223)
223      FORMAT (' If you wish the following format, type a filename, ',
     +           'otherwise type a return key.',/)
```

```
C
C
C   /*   Initialize the pointers       */
C
      write (*,69)
      fname = '                              '
69         format (' Please enter the vector boundary filename :',$)
166        format(' H  ',i2,a2,' polygon set')
      read (*,'(a24)')  fname
      IF (fname .ne. '                              ') THEN
         OPEN (UNIT=11, FILE=fname, STATUS='NEW', FORM='FORMATTED')
         write(*,*)

         write(*,*) ' A 60 character label may be placed on',
     +                    ' this file.'
         write(*,91)
 91      format(' Label :',$)

         read(*,'(a60)') label
         write(11,'(a3,a60)') ' H ',label
         write(11,*) 'H ', xnum, ynum, '                    is the ',
     +                              'size of the image.'
         write(11,*) 'H ', xstart, ystart,'        is the origin ',
     +                              'of the image.'
         write(11,*) 'H ', xstop, ystop, '        is opposite the ',
     +                              'origin.'
         write(11,*) 'H ', xint, yint, ' is the scale of the image.'
         write(11,*) 'H ', zone ,' is the UTM zone.'
         IF (face) THEN
           IF (focus.eq.1) THEN
           write(11,166) focus,'st'
           ELSE
             write(11,166) focus,'nd'
           ENDIF
         ELSE
           write (11,*) 'H  0 polgon sets used with the face values.'
         ENDIF
         boundry = .TRUE.
      ELSE
         boundry = .FALSE.
      ENDIF
            print*
      DO 11, i = 1, n_poly
        save1 (i,0) = null * 2
        save2 (i,0) = 0
        next (i,0) = null
        end (i) = 1
 11    CONTINUE
C
C
      i = 0
      max_coords = 0
      max_segments = 0
      num_coords = 0
C
C     Repeat as long as there are more records to read .....
C
10    Continue
        i = i + 1
        max_segments = max_segments + 1
        coord_start = num_coords
        IF (migsif) THEN
C
C       /*   I haven't  checked this migsif stuff out yet so it
C            might not work !     Casey, July30, 1986  */
C
          READ(10,err=999) record_len,sif_id,num_coords,data_type,
     +    cont_code,left,right,((xy_buf(j,k),j=1,(record_len-8)/4),
     +    k=1,2)
          zone = 0
          IF (data_type.EQ.CHAIN) GO TO 999
C
C     Other data structures that the user may build his own
C      format for the data files.  refer to last subroutine in
C      this file for information
C
        ELSE IF (other) THEN
          CALL  OtherRead (left, right, num_coords, x_coord, y_coord,
     +                        flip, error)
C
C         if the reading is all done then go to printing it out.
C
```

```
          zone = 0
          IF (error) GO TO 999
C
C   Pcdata data files that are recieved form CLDS
C
        ELSE IF (pcdata) THEN
          READ(10,ERR=999,END=999) (inbuff(j),j=1,l_recl)
          DO 2  k = 1,l_recl
             inbuff(k) = inbuff(k) - HEX_30
2         CONTINUE

          right = JINT (inbuff(1)*100000.0+inbuff(2)*10000.0+
     +             inbuff(3)*1000.0 + inbuff(4)*100.0 +inbuff(5)*10.0
     +             + inbuff(6))
          left = JINT(inbuff(7)*100000.0+inbuff(8)*10000.0 +
     +                inbuff(9)*1000.0 + inbuff(10)*100.0 +
     +                inbuff(11)*10.0 + inbuff(12))
          zone = JINT(inbuff(13)*10.0 + inbuff(14))
          num_coords = JINT(inbuff(15)*1000.0 + inbuff(16)*100.0+
     +                 inbuff(17)*10.0+inbuff(18))

             IF (face) THEN
               right = FaceToPoly(right, focus)
               left =  FaceToPoly(left, focus)
             ENDIF

          IF (JMOD(i,50).EQ.0) WRITE(*,1011) i
1011      FORMAT('          Processing segment # ',I5)

          istart = 19
          ic = 0
C
          IF (flip) THEN
          DO 3 j = 1,num_coords
            y_coord(j) = 0
            x_coord(j) = 0

            max_coords = max_coords + 1

            DO 4 k = 1,6
               y_coord(j) = JINT(inbuff(istart+ic)*exp(k)) + y_coord(j)
               ic = ic + 1
4           CONTINUE
            DO 5 k = 1,8
               x_coord(j) = JINT(inbuff(istart+ic)*exp1(k)) + x_coord(j)
               ic = ic + 1
5           CONTINUE

3         CONTINUE
          Temp = Left
          Left = Right
          Right = Temp
          ELSE

            DO 6 j = 1,num_coords
              x_coord(j) = 0
              y_coord(j) = 0
              max_coords = max_coords + 1
              DO 7 k = 1,6
                 x_coord(j) = JINT(inbuff(istart+ic)*exp(k))
     +                                + x_coord(j)
                 ic = ic + 1
7             CONTINUE
              DO 8 k = 1,8
                 y_coord(j) = JINT(inbuff(istart+ic)*exp1(k))
     +                                + y_coord(j)
                 ic = ic + 1
8             CONTINUE
6           CONTINUE

          END IF
        END IF
C
C   If a vector file is desired then output the left and right
C   poly numbers.
C
       IF ((boundry) .and. (numofcoords.gt.1) .and.
     +     (oldleft.ne.oldright)) THEN
          call compress (oldleft, oldright, numofcoords, out, out1)
       ENDIF

       oldleft = left
```

```
      oldright = right
      numofcoords = 0
C
C    /*  For each pair of points in xy_coord determine the intervals.. */
C
        first = .TRUE.

        DO 9 j = 1, num_coords-1

          IF (y_coord(j) .le. y_coord(j+1)) THEN
             x1 = x_coord(j) * 1.0
             y1 = y_coord(j) * 1.0
             x2 = x_coord(j+1) * 1.0
             y2 = y_coord(j+1) * 1.0
             IF (y1.lt.iymin) iymin = y1

               rightside = right

           ELSE
             x1 = x_coord(j+1) * 1.0
             y1 = y_coord(j+1) * 1.0
             x2 = x_coord(j) * 1.0
             y2 = y_coord(j) * 1.0
             IF (y2.gt.iymax) iymax = y2

               rightside = left

           END IF

          IF (x_coord(j)*1.0 .lt. ixmin) ixmin = x_coord(j)*1.0
          IF (x_coord(j)*1.0 .gt. ixmax) ixmax = x_coord(j)*1.0
C
C  /*   Determine if the xy pairs are in the ball park */
C  /*   of the section to be examined.                 */
C
          IF (clipping) THEN
            ballpark =   ((y1 .le. ystop)
     +           .and.    (y2 .ge. ystart))
          ELSE
            ballpark = ((x1.ge.xstart) .and. (x1.le.xstop)
     +      .and. ( x2 .ge. xstart) .and. (x2 .le. xstop)
     +      .and. ( y1 .ge. ystart) .and. (y1 .le. ystop)
     +      .and. ( y2 .ge. ystart) .and. (y2 .le. ystop))
          ENDIF

          IF (ballpark) THEN
C
C  /*   determine the y intervals for y1 and y2.         */
C  /*   if exactly on an interval line then increase the */
C  /*   value by yint/1000 .                             */
C

            m1 = int ((y1 - ystart) / yint + 1)
            y1prim = y1
71          if ((m1 - 1) * yint .eq. y1 - ystart) then
               y1 = y1 + sliver
               go to 71
            endif

            map_y = m1

            m2 = int ((y2 - ystart) / yint)
            y2prim = y2
 72         if ( m2 * yint .eq. y2 - ystart) then
               y2 = y2 + sliver
               go to 72
            endif
C
C  /*     line does cross an interval.  */
C
            IF (m2 .ge. map_y) THEN
                  IF (x2-x1 .ne. 0) THEN
                       slope = ((y2 - y1) * 1.0)/(x2 - x1)
                  ELSE
                       slope = 99999.0
                  END IF
C
C               WHILE (m2 .ge. map_y) DO
C
 100             IF (slope.ne.99999.) THEN
                     true_x = ((map_y * yint + ystart) - y1)
     +                              * 1.0 / slope + x1
```

```
                ELSE
                   true_x = x1
                END IF
C
                n = ((true_x - xstart) * 1.0) / xint

C
C       determine if clipping required and handle some unique cases that
C       may arise with respect to the boundaries of the output image..
C

                IF  ((n .gt. xnum) .or.
     +                (map_y .lt. 1)  .or.
     +                (map_y .gt. ynum) .or.
     +               ((true_x .gt. x1) .and.
     +                (true_x .gt. x2)))     THEN
                    go to 888

                ELSE IF (n .le. 0) THEN

                    IF ((true_x .gt. rsave(map_y,0)) .or.
     +                  (rsave(map_y,0).ge.1)) THEN
                        rsave(map_y,0) = true_x
                        save2(map_y,0) = rightside
                    ENDIF
                    go to 888

                ELSE IF (true_x .lt. rsave(map_y,0)) THEN

                    IF (face) THEN
                       IF (facetopoly(right,focus)
     +                          .ne. rightside) THEN
                          save2(map_y,0) = facetopoly(right,' focus)
                       ELSE
                          save2(map_y,0) = facetopoly(left, focus)
                       ENDIF
                    ELSE
                       IF (right .ne. rightside) THEN
                          save2(map_y,0) = right
                       ELSE
                          save2(map_y,0) = left
                       ENDIF
                    ENDIF

                    rsave(map_y,0) = true_x

                ELSE IF ((true_x .lt. x1) .and.
     +                   (true_x .lt. x2)) THEN

                    go to 888

                ENDIF

C
C         If vector boundary file is to be created then,
C         save the coordinates.
C

            IF ((boundry) .and. first) THEN
                numofcoords = numofcoords + 1
                IF ((y1prim .ne. y1) .or.
     +              (y2prim .ne. y2)) THEN

                  IF (slope .ne. 99999.0) THEN
                       slopeprim = ((y2prim - y1prim) * 1.0)/
     +                                 (x2 - x1)
                       true_xprim = ((map_y * yint + ystart) - y1prim)
     +                              * 1.0 / slopeprim + x1
                  ELSE
                       slopeprim = slope
                       true_xprim = x1
                  END IF
                  nprim = ((true_x - xstart) * 1.0) / xint
                  out(numofcoords)  = nprim
                ELSE
                  out(numofcoords)  = n
                END IF
                out1(numofcoords) = map_y
                IF (out(numofcoords).gt.xnum) THEN
                     numofcoords = numofcoords - 1
                ELSE
                     first = .FALSE.
```

```
                ENDIF
              ENDIF

C
C         Input new value conditions
C
C                   While (not end of y-line)
C                        then check if x already exists
C                        if so then check if x should be replaced
C                        if x doesn't exist add it onto the end.
C                              and link the pointers to it.
C

                      pointer = 0
                      traverse = .true.
 111                  Continue


C                           valid x to test then
C                           saved x > x then save
C                           saved x = x then compare
C                           saved x < x then traverse to next saved x
C
                      if (pointer.eq.n_poly) then
                               traverse = .false.
                      else if (next(map_y,pointer).eq.null) then
                               traverse = .false.
                      else if (save1(map_y, pointer).ge.n) then
                               traverse = .false.
                      else if (save1(map_y, pointer).lt.n) then
                               last = pointer
                               pointer = next(map_y, last)
                      else
                        print *,'Something funny is going on.'
                      end if
                      if (traverse) goto 111
C
C         If reached the end of the link list.
C
                      if ((pointer.eq.n_poly).or.
     +                   ( end(map_y)+1.eq.n_poly)) then
                         print *, ' error - too many points on one',
     +                           ' raster line.   line = ',map_y
                         goto 999
C
C        If that x point already exists in the link list then compare.
C
                    else if (save1(map_y, pointer).eq.n) then
                       if (rsave(map_y, pointer).lt.true_x)   then
                             renew = .true.
                       else
                            renew = .false.
                       end if
C
C       /*              If passed the x point                */
C
                    else if (save1(map_y, pointer).gt.n) then
                             spot = end(map_y)
                             next(map_y, spot) = pointer
                             next(map_y, last) = spot
                             pointer = spot
                             end (map_y) = spot + 1
                             renew = .true.
C
C         /*  If the x point has reached the end of the list.   */
C
                    else if (next(map_y,pointer).eq.null) then
                             spot = end(map_y)
                             next(map_y, pointer) = spot
                             next(map_y, spot) = null
                             pointer = spot
                             end (map_y) = end (map_y) + 1
                             renew = .true.
C
C     /*    If nothing satisfied by now, major problems exist.   */
C
                      else
                        print *,'++++++++++ MAJOR ERROR ++++++++++'
                        renew = .false.
                      end if
C
                      IF (renew) THEN
```

```
                        save1 (map_y, pointer) = n
                        save2 (map_y, pointer) = rightside
                       rsave (map_y, pointer) = true_x
                      END IF

 888                  map_y = map_y + 1
                      if (m2 .ge. map_y) go to 100

                      IF (boundry) THEN
                         IF ((out1(numofcoords) .ne. map_y -1)
     +                  .or. (out(numofcoords) .ne. n)) then
                            numofcoords = numofcoords + 1
                            out1(numofcoords)  = map_y - 1
                            out(numofcoords) = n
                            IF (out(numofcoords).gt.xnum) THEN
                                  numofcoords = numofcoords - 1
                            ENDIF
                         ENDIF
                         first = .TRUE.
                      END IF
C
C                   END WHILE
C
              ELSE IF (boundry) THEN
                IF (first) THEN
                  numofcoords = numofcoords + 1
                  out1(numofcoords) = m1
                   out(numofcoords) = int((x1-xstart)/xint)
                  if ((out(numofcoords).lt.0).or.
     +                (out(numofcoords).gt.xnum)) then
                      numofcoords = numofcoords - 1
                  else
                      if (out1(numofcoords).lt.0) then
                          out1(numofcoords) = 1
                      else if (out1(numofcoords).gt.ynum) then
                          out1(numofcoords) = ynum
                      endif
                      first = .FALSE.
                  endif
                ENDIF
                IF (.not.first) then
                      numofcoords = numofcoords + 1
                      out1(numofcoords) = out1(numofcoords-1)
                      out(numofcoords) = int((x2-xstart)/xint)
                      if ((out(numofcoords).lt.0).or.
     +                    (out(numofcoords).gt.xnum)) then
                          numofcoords = numofcoords - 1
                      else
                        if (out1(numofcoords).lt.0) then
                          out1(numofcoords) = 1
                        else if (out1(numofcoords).gt.ynum) then
                          out1(numofcoords) = ynum
                        endif
                        IF ((out1(numofcoords-1) .eq. out1(numofcoords))
     +                                          .and.
     +                   (out(numofcoords-1) .eq. out(numofcoords))) then
                              numofcoords = numofcoords - 1
                        ENDIF
                      endif
                  ENDIF

              END IF

            END IF
 9        CONTINUE
            IF (x_coord(j)*1.0 .lt. ixmin) ixmin = x_coord(j)*1.0
            IF (y_coord(j)*1.0 .lt. iymin) iymin = y_coord(j)*1.0
            IF (x_coord(j)*1.0 .gt. ixmax) ixmax = x_coord(j)*1.0
            IF (y_coord(j)*1.0 .gt. iymax) iymax = y_coord(j)*1.0

C
C  This means that there did exist an xy pair and that no error
C  occured when trying to read the value.  Therefore return to
C  the top and try to read the next xy pair.
C
       GO TO 10
C
C  To get to this point it shows that all the xy pairs have been
C  read in and now output of the data may begin
C
 999    CONTINUE
```

```
      IF ((boundry) .and. (numofcoords.gt.0) .and.
     +    (oldright.ne.oldleft)) THEN
          call compress (oldleft, oldright, numofcoords, out, out1)
          close(11)
      ENDIF
C
C         Determine the order to output the information...
C
      IF (flip) THEN
                a = 1
                b = ynum
                c = 1
      ELSE
                a = ynum
                b = 1
                c = -1
      END IF
C
C         Determine the type of output required...
C
        print*, char(7), char(7)
        WRITE (*,200)
200     FORMAT ('        All input segments read .....')
C
C   If a vector file is desired then output the left and right
C   poly numbers.
C

C        READ  (*,'(a1)') reply
        reply = 'A'
        WRITE (*,*)
        write(*,223)
C
C            Set-up the Output Raster File
C
        IF (reply .eq. 'r' .or. reply .eq. 'R'.or.
     +      reply .eq. 'a' .or. reply .eq. 'A') then
C
C          For all the regions of interest....
C
            WRITE (*,301)
            fname = '                                        '
301         FORMAT (' Please input the expanded raster',
     +           ' filename :',$)
            READ (*,'(a32)') fname
            IF (fname .ne. '                                 ') Then
            OPEN (UNIT=6, FILE=fname, FORM='formatted',
     +           STATUS='NEW')
            WRITE (6,*) ynum, xnum
            DO 201 l= a, b, c
             j = 0
             pointer = 0
             last = save2(l,j)
             write (6,*)
C            write (*,*)
C
C          While next(l,j) continue writing out info for the one line
C
 203          IF (next(l,j).ne.null) THEN
                j = next(l,j)
                IF (save2(l,j) .ne. last) THEN
                   do 204, k=1, save1(l,j)-pointer
C                     write (*,205) last
 204                  write (6,205) last
 205                  format ('+',i4,$)
                   IF (save1(l,j).gt.0) THEN
                        pointer = save1(l,j)
                   ELSE
                        pointer = 0
                   END IF
                   last = save2(l,j)
                END IF
                GOTO 203
C
C          END WHILE
C
           END IF
C
C      /*   If the end of the region has not been reached   */
C
            IF (pointer .lt. xnum) THEN
              write (6,205) ((last),k=1,(xnum-pointer))
```

```
C               write (*,205) ((last),k=1,(xnum-pointer))
              END IF
201           CONTINUE
              CLOSE (6)
              ENDIF
              WRITE (*,*)
           END IF
C
C       If a morpholog data file of the raster image is wanted,
C          simply enter the filename when prompted for one ....
C       For all the regions of interest dump to a morpholog file ....
C
              xdes = 1
              ydes = 1
              WRITE (*,300)
              fname =          '                                    '
 300          Format (' Please input the morpholog',
     +             ' filename : ',$)
              READ (*,'(a32)') fname
              morph = (fname .ne.  '                                    ')
              WRITE (*,*)
              WRITE (*,380)
              boundfile =          '                                    '
 380          Format (' Please input the morpholog BOUNDARY',
     +                ' filename : ',$)
              READ (*,'(a32)') boundfile
              bound = (boundfile .ne. '                                    ')
              IF (bound.or.morph) THEN
               IF (xnum.gt.512 .or. ynum.gt.512) THEN
                 IF (xnum.gt.512) THEN
                    WRITE (*,*)
777                 WRITE (*,*) ' WARNING - number of x pixels ',
     +                        'must be less than or equal to 512,'
                    WRITE (*,*)'              you have ',xnum,
     +                         ' pixels'
                    xdes = xnum / 512
                    IF (xdes*512 .lt. xnum) xdes = xdes + 1
                    WRITE (*,*) '          recommended integer ',
     +                        'decimation factor of ',xdes
                 END IF
                 IF (ynum.gt.512) THEN
                    WRITE (*,*)
                    WRITE (*,*) 'WARNING - number of y pixels ',
     +              'must be less than or equal to 512,'
                    WRITE (*,*) '              you have ',ynum,
     +                         ' pixels'
                    ydes = ynum / 512
                    IF (ydes*512 .lt. ynum) ydes = ydes + 1
                    WRITE (*,*) '          recommended integer ',
     +                        'decimation factor of ',ydes
                 END IF
                 WRITE (*,*)
                 WRITE (*,700)
700              FORMAT (' Please enter x decimation factor...',$)
                 READ (*,*) xdes
                 WRITE (*,701)
701              FORMAT (' Please enter y decimation factor...',$)
                 READ (*,*) ydes
                 c = c * ydes
               END IF
               IF (morph) THEN
                 OPEN (UNIT=6, FILE=fname, FORM='unformatted',
     +              STATUS='NEW')
                 Write (6) xnum/xdes, ynum/ydes
               END IF
               IF (bound) THEN
                 OPEN (UNIT=7, FILE=boundfile, FORM='unformatted',
     +              STATUS='NEW')
                 Write (7) xnum/xdes, ynum/ydes
                 Do 718 l=1,n_poly
                       out2(l) = -10
718              Continue
               END IF
               DO 712, l= a, b, c
                  out1 (1) = 1
                  start = 1
                  j = 0
                  kk = 0
                  last = save2(l,0)
                  pointer = 0
C
C          While next(l,j) continue writing out info for the one line
```

```
C
 703              IF (next(1,j).ne.null) THEN
                    j = next(1,j)
                    do 704, k=1, (save1(1,j)-pointer)/xdes
                       kk = kk + 1
 704                   out(kk) = last
                    IF (((save1(1,j)-pointer)/xdes .gt. 0).and.
     +                  (last.ne.save2(1,j))) THEN
                       start = start + 1
                       out1 (start) = 1
                       do 707, k = start+1, kk
 707                     out1 (k) = 0
                       start = kk
                    END IF
                    IF (save1(1,j).gt.0) THEN
                         pointer = save1(1,j)
                    ELSE
                         pointer = 0
                    END IF
                    last = save2(1,j)
                  GOTO 703
C
C                 END WHILE
C
                  END IF
C
C       /*   If the end of the region has not been reached   */
C
                  IF (kk .lt. xnum/xdes) THEN
                     pointer = kk + 1
                     DO 713, kk = pointer, xnum/xdes
                       out(kk) = last
                       out1(kk) = 0
 713                 CONTINUE.
                     kk = xnum/xdes
                  END IF

c                  IF (kk.gt.512) THEN
c                      WRITE (*,*) 'ERROR -- in x decimation factor, ',
c     +                            'please increase it.'
c                      GO TO 777
c                  END IF

                  IF (kk .ne. xnum/xdes) THEN
                     WRITE (*,*) 'ERROR -- kk <> xnum/xdes', kk, xnum/xdes
                  END IF

                  out1(kk) = 1

                  Do 717 j = 1, kk
                     IF (out2(j).ne.out(j)) THEN
                          out1(j) = 1
                     ENDIF
                     out2(j)=out(j)
 717                 Continue

                  IF (morph) WRITE (6) (out(j),j=1,kk)
                  IF (bound) WRITE (7) (out1(j),j=1,kk)

 712               CONTINUE
                   IF (morph) CLOSE (6)
                   IF (bound) CLOSE (7)
                 END IF
             write (*,*)
C
C         For the run length encoded output
C
           IF (reply .eq.'L'.or.reply.eq.'l'.or.
     +         reply .eq.'A'.or.reply.eq.'a') THEN

              boundfile = '                                '
              crle = .FALSE.
              fname = '                                    '
              rle = .FALSE.


              WRITE (*,806)
 806          FORMAT (' Please input the formatted run length encoded',
     +             ' filename :',$)
              READ (*,'(a32)') fname
              If (fname .ne. '                                    ') Then
                    rle = .TRUE.
```

```
            endif
                  print*
            IF (rle) THEN
               OPEN (UNIT=6, FILE=fname, FORM='formatted',
     +            STATUS='NEW')
               write(*,*) ' A 60 character label may be placed on',
     +                    ' this file.'
               write(*,91)
               read(*,'(a60)') label
               write(6,'(a3,a60)') ' H ',label
               write(6,*) 'H ', xnum, ynum, '                    ',
     +                     'is the size of the image.'
               write(6,*) 'H ', xstart, ystart,'       is the ',
     +                    'origin of the image.'
               write(6,*) 'H ', xstop, ystop, '       is opposite',
     +                    ' the origin.'
               write(6,*) 'H ', xint, yint, ' is the scale of ',
     +                    'the image.'
               write(6,*) 'H ', zone ,' is the UTM zone.'
               IF (face) THEN
                 IF (focus.eq.1) THEN
                   write(6,166) focus,'st'
                 ELSE
                   write(6,166) focus,'nd'
                 ENDIF
               ELSE
                 write (6,*) 'H  0 polgon sets used with the ',
     +                       'face values.'
               ENDIF
               boundry = .TRUE.
               WRITE (6,*) ynum, xnum
             ELSE
               boundry = .FALSE.
             ENDIF
                  print *

            WRITE (*,606)
606         FORMAT (' Please input the unformatted run length encoded',
     +          ' filename :',$)
            READ (*,'(a32)') boundfile

            IF (boundfile .ne. '                                ') THEN
                  crle = .TRUE.
            endif


            IF (crle) THEN
               OPEN (UNIT=7, FILE=boundfile, FORM='unformatted',
     +            STATUS='NEW')
               WRITE (7) ynum, xnum
            ENDIF

            IF (crle.or.rle) THEN
              DO 601 l= a, b, c
                j = 0
                distance = 0
                last = save2(1,0)
                second_last = last
                pointer = 0
C
C          While next(l,j) continue writing out info for the one line
C
 603            IF (next(l,j).ne.null) THEN
                  j = next(l,j)
                  IF ((save2(l,j).ne.last).and.
     +                (save1(l,j)-pointer.gt.0)) THEN
                    IF (distance .ne. 0) THEN
                        IF (rle) write (6,*) second_last, distance
                        IF (crle) write (7) second_last, distance
                    ENDIF
                   second_last = last
                   distance = save1(l,j)-pointer
                   IF (save1(l,j).gt.0) THEN
                      pointer = save1(l,j)
                   ELSE
                      pointer = 0
                   END IF
                   last = save2(l,j)
                  END IF
               GOTO 603
C
C             END WHILE
```

```
C
                END IF
C
C       /*   If the end of the region has not been reached   */
C
                IF ((pointer .lt. xnum)      .and.
     +              (second_last .eq. last)) THEN
                   reduce = xnum - pointer + distance
                   IF (reduce.gt.0) THEN
                      IF (rle) write (6,*) last, reduce
                      IF (crle) write (7) last, reduce
                   ENDIF
                ELSE IF (distance .gt. 0) THEN
                   reduce = xnum - pointer
                   IF (rle) THEN
                         write (6,*) second_last, distance
                      IF (reduce.gt.0)
     +                    write (6,*) last, reduce
                   ENDIF
                   IF (crle) THEN
                         write (7) second_last, distance
                      IF (reduce.gt.0)
     +                    write (7) last, reduce
                   ENDIF
                ELSE
                   reduce = xnum - pointer
                   IF (reduce.gt.0) THEN
                      IF (rle) write (6,*) last, reduce
                      IF (crle) write (7) last, reduce
                   ENDIF
                ENDIF

601             CONTINUE
                IF (rle) CLOSE (6)
                IF (crle) CLOSE (7)
             ENDIF
             WRITE (*,*)
          END IF

          IF (flip) THEN
            WRITE (*,*) ' Northing Range   min = ',ixmin,' max = ', ixmax
            WRITE (*,*) ' Easting  Range   min = ',iymin,' max = ', iymax
          ELSE
            WRITE (*,*) ' Northing Range   min = ',iymin,' max = ', iymax
            WRITE (*,*) ' Easting  Range   min = ',ixmin,' max = ', ixmax
          END IF
          print*

       write (*,1111)
 1111  format (' ','Test somemore ? Y/n ',$)
       READ(*,'(a1)') reply
       WRITE (*,*)
         IF (reply.ne.'n'.and.reply.ne.'N') go to 555

       IF (Is_tape) THEN
            PRINT*
            PRINT*,'             One moment ..... Tape Rewinding'
            REWIND (10)
            REWIND (10)
            PRINT*, char(7)
       ENDIF
       CLOSE (10)

       STOP
       END

       SUBROUTINE HELP (what)
       INTEGER what
       IF (what .eq. 1) THEN
          WRITE (*,1)

  1      FORMAT (' ',//,
     +   '    This program is designed to compute a raster format',/,
     +   ' from an already existing vector file.  The input file is',/,
     +   ' a vector file and must be in the format of MIGSIF, PCDATA',/,
     +   ' or of a users written format. The vector data file may be',/,
     +   ' windowed, blown up, shrunk down or even rotated clockwise',/,
     +   ' 90 degrees.  The only restriction is that the output y',/,
     +   ' direction can never exceed 1024 pixels in size.',/,
     +   '    An added advancement is the ability to input face value,',/,
     +   ' polygon value pairs.  This way the true image may be seen,',/,
     +   ' given a vector file of multiple images all in one.')
```

```
      ELSE IF (What .eq. 2) THEN
         WRITE (*,2)

2       FORMAT (' ',/,
     +  '     The file type for the input vector file must be one of',/,
     +  ' the three types shown, if it is not then to run the file',/,
     +  ' code changes must be made to the subroutine OTHER found ',/,
     +  ' at the end of vec2ras.for, to allow for the input of your',/,
     +  ' file type. (subroutine OTHER has a format at present for ',/,
     +  ' possible reference purposes.)')

      ELSE IF (What .eq. 3) THEN
         WRITE (*,3)

3       FORMAT (' ',/,
     +  '     This requests if the image given is to be rotated 90',/,
     +  ' degrees clockwise.  This option is mainly to allow for ',/,
     +  ' the full image to fit.  Since 1024 is the max allowance',/,
     +  ' in the y-direction, if the image is 1222 in the y, but',/,
     +  ' only 999 in the x-direction a simple rotation is all that',/,
     +  ' would be required.')
      END IF
      RETURN
      END



      SUBROUTINE compress (left, right, num, out, out1)
C
C     This subroutine trys to quickly compress the data that will be saved
C    in the new vector boundary file.
C     Compression is always being traded for speed, so the only test to
C    compress will be if the line shown has three or more horizontal or
C    vertical points, were the middle points may be eliminated.
C
      INTEGER*2 left, right, num, out(1:1024), out1(1:1024),
     +                  donenum, done(1024,2)

      donenum = 0
      if (num.ge.2) then
        if (num.gt.2) then
          donenum = 1
          done(1,1) = out(1)
          done(1,2) = out1(1)
          do 1 i = 2, num - 1
             if ((done(donenum,2) .eq.out1(i))    .and.
     +            (out1(i+1) .eq.out1(i)))  then
                if ((done(donenum,1).lt.out(i))     .and.
     +              (out(i)  .lt.out(i+1))) then
                   go to 1
                else if ((done(donenum,1).gt.out(i))     .and.
     +                    (out(i)  .gt.out(i+1))) then
                   go to 1
                else
                   donenum = donenum + 1
                   done(donenum, 1) = out(i)
                   done(donenum, 2) = out1(i)
                endif
             else if ((done(donenum, 1).eq.out(i))    .and.
     +            (out(i+1) .eq.out(i)))  then
                if ((done(donenum,2).lt.out1(i))     .and.
     +              (out1(i)  .lt.out1(i+1))) then
                   go to 1
                else if ((done(donenum,2).gt.out1(i))     .and.
     +                    (out1(i)  .gt.out1(i+1))) then
                   go to 1
                else
                   donenum = donenum + 1
                   done(donenum, 1) = out(i)
                   done(donenum, 2) = out1(i)
                endif
             else
                donenum = donenum + 1
                done(donenum, 1) = out(i)
                done(donenum, 2) = out1(i)
             endif
1         continue

          write (11,4) left, right, donenum+1
4         format(' ',i4,i5,i4)
```

```
            do 2  kij = 1, donenum
              write (11,3) done(kij,1), done(kij,2)
 2          continue
            write (11,3) out(num), out1(num)
 3          format(' ',i4,i5)

          else
            write (11,4) left, right, 2
            write(11,3) out(1), out1(1)
            write(11,3) out(2), out1(2)
          endif
        endif

        return
        end



        SUBROUTINE clip (x1, y1, x2, y2, xint, yint, xnum, ynum,
       +                 xcoord, ycoord)

        INTEGER*2 xcoord, ycoord, xnum, ynum
        REAL      xint, yint, x1, y1, x2, y2
C
C     This subroutine is responsible for determining is any two
C   xy pairs that define an area, do in any way cross the grid
C   region that has been declared.  And if so save the two sets
C   of coordinates.
C
        end



        SUBROUTINE FaceReader  (FaceToPoly, migsif, pcdata, other, Pnum,
       +                        tape, image)

C
C           This routine is designed to read and load up an array of
C       face value, polygon value pairs, to be used when reading
C       a polygon file.
C           With the tapes sent over from CLDS all the vector files
C       are followed by a second file.  This second file has the following
C       data.   CLDS identifier (face_value)
C               Polygon Area, Centroid X and Y, Zone Number and
C               Polygon Number[s].
C           This knowlegde is required to coorespond the CLDS identifier
C       with the valid polygon number, allowing for numbers below the
C       value of 512.
C

        BYTE  inbuff(4096)

        INTEGER*2 FaceToPoly(2000, 2)

        INTEGER*4 Face_Num, Area, X_Utm, Y_Utm, HEX_30,
       +          Zone, Poly_Num, Num_Records, L_Recl,
       +          Pnum, Face, Flen, Plen1, Plen2, Poly1, Poly2,
       +          sea(20), sea_max, sea_to_be, accurate,
       +          filelabel

        REAL hold
        LOGICAL migsig, pcdata, other, mask, Is_tape, clipping,
       +        dump, tape
        CHARACTER filename*24, fn*1, fname*24, blanks*24, image*24

        DATA HEX_30/'30'X/
C
C   /*   Request the data file and the output filenames     */
C
        blanks = '                                '
        fname = '                               '
        filename = '                                  '

        PRINT*,' There are two ways to input the face-polygon',
       +       ' relationship...'
34        PRINT*
        PRINT*,' EITHER enter the CLDS descriptive data set'
        PRINT*,' OR     enter the saved face-polygon results,'
        PRINT*,'              generated from an earlier run.'
        PRINT*
        WRITE (*,141)
141     FORMAT (' (D)escriptive data or (e)arlier run saved data',//,
```

```
     +        '  Enter choice :',$)
      READ (*,'(a1)') reply
      PRINT*

      IF ((reply .eq. 'E') .or. (reply .eq. 'e')) THEN
          WRITE (*,40)
40        FORMAT (' Name of the already built face-poly filename :',$)
             READ (*,'(a24)') fname
          filename = blanks
      ELSE
        IF (tape) THEN
           PRINT 26
26         FORMAT (' Is the descriptive data set the second file,',/,
     +        '  on the same tape drive as the image set? (Y/n) ',$)
           READ (*,'(a1)') fn
           Is_tape = .FALSE.
           IF ((fn .ne. 'N') .and. (fn .ne. 'n')) THEN
              Is_tape = .TRUE.

              IF (migsif.or.pcdata) THEN
                OPEN(12,ERR=2,FILE=image,FORM='unformatted',
     +             RECORDTYPE='VARIABLE', STATUS='old')
                GO TO 3
2               OPEN(12, FILE=image, FORM='unformatted',
     +             STATUS='old')
3               CONTINUE
              ELSE
                OPEN(12,FILE=image,FORM='formatted',STATUS='old')
              END IF
              filename = image
              PRINT*
              ii = 0
           PRINT*, '         Reading past the first file, one moment ...'
              PRINT*
28            READ (12,END=27,ERR=999)
              ii = ii + 1
              IF (jmod(ii, 1000).eq.0) then
                 print*,ii,' records passed.'
              ENDIF
              GO TO 28
999           PRINT*,'MASSIVE ERROR HAS OCCURED.'
27            PRINT*, char(7), char(7)
              ii = 0
              PRINT*, '      First file has now been passed by ......'
           ELSE
              PRINT 1
1             FORMAT (' Name of the input CLDS',/,
     +                '  description data set :',$)
              READ(*,'(A24)') filename
              fname = blanks

              IF (migsif.or.pcdata) THEN
                OPEN(12,ERR=72,FILE=filename,FORM='unformatted',
     +             RECORDTYPE='VARIABLE', STATUS='old')
                GO TO 73
72              OPEN(12, FILE=filename, FORM='unformatted',
     +             STATUS='old')
73               CONTINUE
              ELSE
                OPEN(12,FILE=filename,FORM='formatted',STATUS='old')
              END IF

           ENDIF
        ELSE
           PRINT 1
           READ (*,'(A24)') filename
           fname = blanks

           IF (migsif.or.pcdata) THEN
                OPEN(12,ERR=82,FILE=filename,FORM='unformatted',
     +             RECORDTYPE='VARIABLE', STATUS='old')
                GO TO 83
82              OPEN(12, FILE=filename, FORM='unformatted',
     +             STATUS='old')
83              CONTINUE
           ELSE
                OPEN(12,FILE=filename,FORM='formatted',STATUS='old')
           END IF

        ENDIF
      ENDIF
```

```
      IF ((fname .eq. blanks) .and. (filename .ne. blanks)) THEN
        PRINT 4
4       FORMAT(' The logical record length of the descriptive data',/,
     +         'set? (normally 37 or 41) : ',$)
        READ(*,*) l_recl
              PRINT*

        PRINT*, ' How many images are associated with ',
     +    'the input vector file ?'

        IF (l_recl .eq. 37) THEN
           write (*,5), 1, 2
           sea_max = 1
        ELSE
           write (*,5), 2, 2
           sea_max = 2
        ENDIF
5       FORMAT (' ',' (default is ',i1,', max is ',i1,') : ',$)

        pnum = NumberReader (sea_max)
              PRINT*

        PRINT 6
6       FORMAT(' What is that starting byte for the face # ?',/,
     +         '       (default is 1) : ',$)
        face = NumberReader (1)
              PRINT*

        PRINT 7
7       FORMAT(' What is the length of the face value (in bytes) ?',/,
     +         '       (default is 6) : ',$)
        flen = NumberReader (6)
              PRINT*

        PRINT 8
8       FORMAT(' What is that starting byte for the first image ',
     + 'poly_no ?',/,'       (default is 34) : ',$)
        poly1 = NumberReader (34)
              PRINT*

        PRINT 9
9       FORMAT (' What is the length of this polygon number',
     +  ' (in bytes) ?',/,'       (default is 4) : ',$)
        plen1 = NumberReader (4)
              PRINT*

        accurate = 0
        sea_to_be = 0

        IF (Pnum .eq. 2) THEN

           PRINT 10
10         FORMAT(' What is that starting bit for the second images ',
     +    'polygons numbers ?',/,'       (default is 38) : ',$)
           poly2 = NumberReader (38)
                 PRINT*

         PRINT 11
11       FORMAT (' What is the length of the polygon numbers',
     +   ' (in bits) ?',/,  '       (default is 4) : ',$)
         plen2 = NumberReader (4)
                 PRINT*

         PRINT 24
24       FORMAT (' Does one image require cookie-cutting',
     +           ' with the other for coastline accuracy ? ',//,
     +           ' NOTE: This requires knowledge of polygon',
     +           ' numbers for SEA in each image.',//,
     +           ' Do you wish to cookie-cut one  image',
     +           ' with the other (y/N) ',$)
         READ (*,'(a1)') fn
                 PRINT*

         IF ((fn .eq.'y') .or. (fn.eq.'Y')) THEN

                   PRINT*
           clipping = .true.
           PRINT 21
21         FORMAT (' The accurate coastline is found in',
     +             '  image # ? (1/2) ',/,
     +             ' (default is 1) ',$)
           accurate = Numberreader (1)
```

```
                PRINT*

            PRINT 29, accurate
29          FORMAT (' How many unique polygon numbers define',
     +      /,       ' the sea on the accurate image set ',i1,'.',/,
     +               ' (default is 1) ',$)
            sea_max = numberreader (1)
                PRINT*

            PRINT 22, accurate
22          FORMAT (' The sea on this set ',i1,
     +               ' is assigned polygon numbers ',/,$)
            READ (*,*) (sea(ii),ii=1,sea_max)
                PRINT*

            PRINT 23, JMOD(accurate,2) + 1
23          FORMAT (' The sea on the other image ',i1,
     +               ' will be assigned to the one polygon number :',$)
            READ (*,*) sea_to_be
                PRINT*

          ELSE
             clipping = .false.
          ENDIF

        ENDIF
        PRINT*
        PRINT*,' The face value to polygon number'
        PRINT*,' look up table may be saved,'
        PRINT*,' OR, just hit return key to omit the option.'
        PRINT*
        WRITE (*,30)
        fname = '                                        '
30     FORMAT (' The filename to save, face to polygon pairs :',$)
            READ (*,'(a32)') fname
        IF (fname .ne. '                                 ') Then
            OPEN (UNIT=13, FILE=fname, FORM='unformatted',
     +          STATUS='NEW')
            write (13) pnum
            dump = .TRUE.
        ENDIF
        PRINT*
C
        MAXP = 0
        i = 0
              print*
13        i = i + 1
          IF (i .gt. 2000) GOTO 18
          READ(12,END=20, err = 19) (inbuff(j),j=1,l_recl)
          DO 14  k = 1,l_recl
             inbuff(k) = inbuff(k) - HEX_30
14        CONTINUE

          hold = 0

          DO 15, j=face, (face + flen - 1)
            hold = hold * 10 + inbuff(j)
15        CONTINUE

          face_num = JINT (hold)
          hold = 0

          DO 16, j=poly1, (poly1 + plen1 - 1)
            hold = hold * 10 + inbuff(j)
16        CONTINUE

          FaceToPoly(Face_Num,1) = JINT (hold)

          mask = .FALSE.

          IF ((clipping) .and. (accurate.eq.1)) THEN
             Ihold = JINT (hold)
             DO 31, ii=1, sea_max
               IF (sea(ii) .eq. Ihold) THEN
                 mask = .TRUE.
                 FaceToPoly(Face_Num, 2) = sea_to_be
               ENDIF
31           CONTINUE
          ENDIF
          IF ((Pnum .eq. 2) .and. (.NOT. mask)) THEN
             hold = 0
             DO 17, j=poly2, (poly2 + plen2 - 1)
```

```
              hold = hold * 10 + inbuff(j)
17            CONTINUE

              IF ((clipping) .and. (accurate.eq.2)) THEN
                Ihold = JINT (hold)

                DO 32, ii=1, sea_max
                  IF (sea(ii) .eq. Ihold) THEN
                    mask = .TRUE.
                    FaceToPoly(Face_Num, 1) = sea_to_be
                  ENDIF
32                CONTINUE

              ENDIF
              FaceToPoly(Face_Num, 2) = JINT (hold)
            ENDIF
          IF (dump) THEN
            IF (pnum .eq. 2) THEN
              write (13) face_num, facetopoly(face_num, 1),
     +                   facetopoly(face_num, 2)
            ELSE
              write (13) face_num, facetopoly(face_num, 1)
          ENDIF
         ENDIF

         IF (JMOD(i,100).eq.0) Print 25, i
25              FORMAT (' Read in',i5,' face values.')
        GOTO 13

18      print*, char(7), char(7), char(7)
        PRINT*,' WARNING Too many Face numbers overwriting possible'
19      PRINT*,'     ERROR occured in reading the Face Value File.'

20      CONTINUE

        IF (Is_tape) THEN
           PRINT*
           PRINT*,'              One moment ..... Tape Rewinding'
           REWIND (12)
           REWIND (12)
           PRINT*, char(7)
        ENDIF
        Close(12)

       ELSE IF ((filename .eq. blanks) .and. (fname .ne. blanks)) THEN
              OPEN (UNIT=13, FILE=fname, FORM='unformatted',
     +              STATUS='OLD')
              READ (13) Pnum
              i = 0
                 print*
38               i = i + 1
                 IF (i .gt. 2000) GOTO 37
                 IF (pnum .eq.1) THEN
                    READ(13, END=35, ERR=36) face_num,
     +                                       facetopoly(face_num,1)
                 ELSE
                    READ(13, END=35, ERR=36) face_num ,
     +                                       facetopoly(face_num,1),
     +                                       facetopoly(face_num,2)
                 ENDIF
                 GOTO 38
37               print*, char(7), char(7), char(7)
            PRINT*,' WARNING Too many Face numbers overwriting possible'
36               PRINT*,' ERROR reading the face-polygon pairs file.'
                 STOP
35               CLOSE(13)
       ENDIF
       RETURN
       END


       FUNCTION NumberReader (default)
C
C        This just reads a chracter string of lingth 3 and
C      converts it to an integer, were a space means assign
C      the default value.
C
       INTEGER*4  NumberReader, number, default, start, i
       CHARACTER*11 ch

       ch = '           '
       READ(*,'(a10)') ch
```

```
      number = 0
      start = 0
1     start = start + 1
      if ((ch(start:start) .eq. ' ') .and. (start .le. 10)) goto 1

          DO 2, i = start, 10
             if ((ch(i:i).ge.'0') .and. (ch(i:i).le.'9')) then
                 number = number*10 + ichar (ch(i:i)) - 48
             else
                 goto 3
             endif
2         CONTINUE

3     IF (i .eq. start) number = default
      NumberReader = number
      RETURN
      END



      SUBROUTINE  OtherRead (left, right, num_coords, x_coord,
     +                       y_coord, flip, error)
C
C
C   This is a user's routine intended to allow the user to input
C   his own unique file type for the program to run.
C   As can be seen the following variables must be entered at
C   each pass of this function call.
C
C   Each call to this function reads from the file the polygon
C   to the left and right of any one segment, the coordinates
C   of the segment and the number of coordinates for the left
C   right pair.
C
C   It uses the variable FLIP to determine if the x read must
C   be flipped to the y-axis and vise versa for the y coord.
C
C   Finally the ERROR logical tells when the file has been
C   totally read and the EOF marker has been encountered.
C
C   The only other issue is the OPENING of the data file.
C   This is done on line 104 of the main program and may
C   require changes as well.
C
C
      PARAMETER  n_polyseg=256

      INTEGER*4 left, right, num_coords, x_coord(n_polyseg),
     +                                   y_coord(n_polyseg)
      LOGICAL flip, error

      error = false
      IF (Flip) THEN
        READ (10,fmt=*,ERR=998) num_coords, right, left
        DO 10 i=1,num_coords
          READ (10,fmt=*,ERR=999) y_coord(i), x_coord(i)
10      CONTINUE
      ELSE
        READ (10,fmt=*,ERR=998) num_coords, left, right
        DO 20 i = 1, num_coords
          READ (10,fmt=*,ERR=999) x_coord(i), y_coord(i)
20      CONTINUE
      ENDIF
      RETURN
C
C  This says that there are no more segments in the file, so
C  quit trying to read them and continue to the output of the
C  data that has already been collected.
C
998   error = .TRUE.
      RETURN
C
C  This shows that the data file used was inconsistent with the
C  data that was given to the previous read.  Therefore report
C  the error and quit the program.
C
999   WRITE (*,*) 'ERROR - Input of the x and y coordinates is ',
     +            'INCORRECT.'
      STOP
      END
```

```
            Vector to Raster File Program

Do you require a brief introduction ? y/N

The name of the input vector file :    TAPE

Is the input vector file on tape ? (Y/n) :

The file type or structure : (m)igsif, (P)cdata, (o)ther (?)Help

The logical record length for image data set :  2818

Many CLDS files require a second file (a descriptive data set) which links
the face value to the true polygon number.
Do you wish such linking ? (y/N) :  Y

 There are two ways to input the face-polygon relationship...

 EITHER enter the CLDS descriptive data set
 OR     enter the saved face-polygon results,
              generated from an earlier run.

(D)escriptive data or (e)arlier run saved data

 Enter choice : d

Is the descriptive data set the second file,
 on the same tape drive as the image set? (Y/n)

        Reading past the first file, one moment ...

       1000 records passed.
       2000 records passed.
       3000 records passed.
       4000 records passed.

     First file has now been passed by ......
The logical record length of the descriptive data
et? (normally 37 or 41) :  41

 How many images are associated with the input vector file ?
      (default is 2, max is 2) :

What is that starting byte for the face # ?
      (default is 1) :

What is the length of the face value (in bytes) ?
      (default is 6) :

What is that starting byte for the first image poly_no ?
      (default is 34) :

What is the length of this polygon number (in bytes) ?
      (default is 4) :

What is that starting bit for the second images polygons numbers ?
      (default is 38) :

What is the length of the polygon numbers (in bits) ?
      (default is 4) :

Does one image require cookie-cutting with the other for coastline accuracy ?

NOTE: This requires knowledge of polygon numbers for SEA in each image.

Do you wish to cookie-cut one  image with the other (y/N)  Y


The accurate coastline is found in  image # ? (1/2)
(default is 1)

How many unique polygon numbers define
the sea on the accurate image set 1.
(default is 1)  3

The sea on this set 1 is assigned polygon numbers
1 410 420
The sea on the other image 2 will be assigned to the one polygon number :  2
```

```
 The face value to polygon number
 look up table may be saved,
 OR, just hit return key to omit the option.

The filename to save, face to polygon pairs :  A.dds


Read in  100 face values.
Read in  200 face values.
          :
          :
Read in 1600 face values.
Read in 1700 face values.

             One moment ..... Tape Rewinding


For the two classification sets which do you wish to focus on ?
(n)one   (F)IRST   (s)econd :  S

 What are the UTM coords of S-W corner for the OUTPUT
 image ?
          Northing : 4955084
          Easting  : 538717

 There are three possible ways to define the output image. Default is 1.

         1. The size for each pixel N-S, and E-W.
            and the number of pixels in N-S, and E-W

                ie (50m X 60m, 256 X 512)

         2. The number of pixels in N-S, and E-W
            and the N-E corner coordinates.

                ie (512 X 1024, ystop, xstop)

         3. The size of each pixel N-S, and E-W
            and the N-E corner coordinates.

                ie (100m X 30m, ystop, xstop)

    Choice : 2

Number of pixels in the output image      i.e.,     Output Image Size
    Along Northing : 1024
    Along Easting  : 1024

What are the UTM coords of N-E corner for the output image ?
           Northing : 5096025
           Easting  : 668215

Do you wish to rotate the image 90 degrees clockwise ? y/N/?


CONFIRMATION of OUTPUT DIMENSIONS & FACE MAPPINGS.

1. S-W corner has coordinates (     4955084,      538717)
2. N-E corner has coordinates (     5096025,      668215)
3. Number of pixels in E-W directions =    1024
4. Number of pixels in N-S directions =    1024
5. Size (in input scale) for 1 E-W output pixel =    126.0000
6. Size (in input scale) for 1 N-S output pixel =    137.5000
7. Face Value   is sent to   Polygon Set  2

          Is this all correct ? Y/n


Output formats

        -  Vector Boundaries     (h)     (formatted)
        -  Expanded Raster,              (formatted)
        -  Morpholog,                    (unformatted)
        -  Morpholog Boundaries Only,    (unformatted)
        -  Run Length Encoded,   (h)     (formatted)
        -  Run Length Encoded,           (unformatted)
any or all of them.
                        NOTE : (h) - Header Records Included.


If you wish the following format, type a filename, otherwise type a return key.
```

```
Please enter the vector boundary filename :      B. VEC

 A 60 character label may be placed on this file.
Label : THIS IS VECTOR FILE FOR CG1156.

        Processing segment #    50
        Processing segment #   100
            :        :        :
            :        :        :
        Processing segment #  4750
        Processing segment #  4800

       All input segments read .....

If you wish the following format, type a filename, otherwise type a return key.

Please input the expanded raster filename :

Please input the morpholog filename :

Please input the morpholog BOUNDARY filename :

Please input the formatted run length encoded filename :   B. POLY

 A 60 character label may be placed on this file.
Label :  THIS IS THE POLYGON FILE FOR CG1156.

Please input the unformatted run length encoded filename :

 Northing Range   min =    4955084.     max =    5096025.
 Easting  Range   min =    538717.0     max =    668215.0

Test somemore ? Y/n  Y


             One moment ..... Tape Rewinding

Do you wish to read or re-read the descriptive
data set to allow for new face value linking ? (y/N) :


For the two classification sets which do you wish to focus on ?
(n)one   (F)IRST   (s)econd :   S

 What are the UTM coords of S-W corner for the OUTPUT
 image ?
                     (default is      4955084)
          Northing :
                     (default is       538717)
          Easting  :

 There are three possible ways to define the output image. Default is 1.

         1. The size for each pixel N-S, and E-W.
            and the number of pixels in N-S, and E-W

                ie (50m X 60m, 256 X 512)

         2. The number of pixels in N-S, and E-W
            and the N-E corner coordinates.

                ie (512 X 1024, ystop, xstop)

         3. The size of each pixel N-S, and E-W
            and the N-E corner coordinates.

                ie (100m X 30m, ystop, xstop)

    Choice : 2

Number of pixels in the output image      i.e.,     Output Image Size
    Along Northing : 512
    Along Easting  : 512
What are the UTM coords of N-E corner for the output image ?
                     (default is      5096025)
          Northing :
                     (default is       668215)
           Easting :

Do you wish to rotate the image 90 degrees clockwise ? y/N/?  Y
```

```
CONFIRMATION of OUTPUT DIMENSIONS & FACE MAPPINGS.

1. S-W corner has coordinates (      538717,     4955084)
2. N-E corner has coordinates (      668215,     5096025)
3. Number of pixels in E-W directions =    512
4. Number of pixels in N-S directions =    512
5. Size (in input scale) for 1 E-W output pixel =    275.0000
6. Size (in input scale) for 1 N-S output pixel =    252.0000
7. Face Value   is sent to   Polygon Set  2

          Is this all correct ? Y/n


Output formats

        -  Vector Boundaries    (h)     (formatted)
        -  Expanded Raster,             (formatted)
        -  Morpholog,                   (unformatted)
        -  Morpholog Boundaries Only,   (unformatted)
        -  Run Length Encoded,  (h)     (formatted)
        -  Run Length Encoded,          (unformatted)
any or all of them.
                        NOTE : (h) - Header Records Included.


If you wish the following format, type a filename, otherwise type a return key.

Please enter the vector boundary filename :

        Processing segment #    50
        Processing segment #   100
            :          :        :
            :          :        :
        Processing segment #  4750
        Processing segment #  4800

       All input segments read .....

If you wish the following format, type a filename, otherwise type a return key.

Please input the expanded raster filename :

Please input the morpholog filename :

Please input the morpholog BOUNDARY filename :

Please input the formatted run length encoded filename :

 A 60 character label may be placed on this file.  A. POLY
Label :  THIS IS THE ROTATED POLYGON FILE CG1156.
Please input the unformatted run length encoded filename :

 Northing Range   min =    538717.0     max =    5096025.
 Easting  Range   min =    538717.0     max =    5096025.

Test somemore ? Y/n   N


            One moment ..... Tape Rewinding

FORTRAN STOP
```

NOTE: The images produced from these files are shown in Figures 3A (raster file A. POLY), 3B (raster file B. POLY) and Figure 4 (vector file B. VEC).