# DOCUMENTATION AND USERS' GUIDE TO THE SPATIAL ENVIRONMENTAL DATA DIGITIZING SYSTEM, SEDDS

By Kenneth J. Lanfear and Anastase Nakassis

---

U. S. GEOLOGICAL SURVEY

OPEN-FILE REPORT 80-871

1980

## Contents

## Contents (continued)

## Illustrations

# DOCUMENTATION AND USERS' GUIDE TO THE SPATIAL ENVIRONMENTAL DATA DIGITIZING SYSTEM, SEDDS

Kenneth J. Lanfear and Anastase Nakassis

---

U. S. GEOLOGICAL SURVEY

---

## Abstract

The Spatial Environmental Data Digitizing System, SEDDS, was developed for analyzing large amounts of environmental data from a variety of sources. This system can combine data from maps of different scales and projections, place spatial data within a matrix of grid cells, and store the data in either of two different storage notations using integers or individual bits, each designed for high speed retrieval by mathematical models. The key concept in the design of SEDDS is the automatic conversion of all spatial data to a standard map projection coordinate system. SEDDS is fully operational, and has been applied to a number of oilspill trajectory analysis models and other applications.

## Introduction

Environmental models often must incorporate data from many sources. This imposes severe demands upon any system which must place spatial data in a digital form, because there is little standardization among the many disciplines involved in preparing environmental data. A further complication is that some environmental data bases tend to be extremely large, which necessitates that computer files be structured for efficient high-speed data retrieval.

The Spatial Environmental Data Digitizing System, SEDDS, is designed to digitize environmental data in a simple and general-purpose manner by accepting a wide variety of source materials, including maps at different scales and projections, as well as electronic data in various formats. It was developed in 1979, initially applied to the problem of oilspill trajectory modeling (see Lanfear and others, 1979), and has since been applied to other areas such as synthetic fuels studies (see Rickert and others, 1979).

A basic design feature of SEDDS is that all map projection conversions are performed by subroutines which act as interchangeable modules for the various procedures. This feature allows the use of different map projections with only minimal procedure changes. Another feature of SEDDS is a simple digitizing routine that automatically stores the reference points that are necessary for converting data into geographical coordinates, and allows all types of maps to be digitized in exactly the same manner. The system includes a number of highly efficient computer algorithms for performing the map conversions and other operations that are necessary for digitizing.

This paper is intended both as a general introduction to SEDDS and as a users' manual. Its organization follows a logical sequence of digitizing steps. After a discussion of digitizing theory, the procedures for digitizing a map are described. This is followed by an explanation of the grid system, instructions for converting maps to the grid system, and a discussion of grid storage. Finally, examples of digitized maps, taken from actual applications of SEDDS, are presented. The appendices contain procedure, program, and subroutine listings, and descriptions of the most important algorithms.

## Theory

For digitizing purposes, a map projection is considered any one-to-one transformation of geographical coordinates into a plane rectangular Cartesian coordinate system. That is,

$$[xm,ym] = T[phi,lamda] \qquad \text{Equation 1}$$

coordinates [xm(i),ym(i)], by transforming known latitudes and longitudes:

$$[xm(i),ym(i)] = T[phi(i),lamda(i)] \qquad \text{Equation 6}$$

It is nearly always possible to find latitudes and longitudes for at least three reference points.

Once the coefficients A1, B1, C1, A2, B2, and C2 have been determined, the geographical coordinates of any digitized point, [xd,yd], can be found by a combination of equations 2 and 5:

$$[phi,lamda] = Tinv[xm,ym] = Tinv[Tdm[xd,yd]] \qquad \text{Equation 7}$$

$$[phi,lamda] = Tinv[A1 + B1\ xd + C1\ yd,\ A2 + B2\ xd + C2\ yd] \qquad \text{Equation 8}$$

Since equation 8 converts all digitized points into geographical coordinates, it is possible to combine sets of data derived from maps of different scales and projections.

For analysis or plotting, it is usually necessary to convert the geographical coordinates back into some cartographic projection of a specific area. SEDDS can establish a system of local plane rectangular Cartesian coordinates, called grid coordinates [xg,yg]. The transformation of map coordinates into grid coordinates is the same as equation 5:

$$Tmg[xm,ym] = [Ag1 + Bg1\ xm + Cg1\ ym,\ Ag2 + Bg2\ xm + Cg2\ ym]$$

$$= [xg,yg] \qquad \text{Equation 9}$$

In a manner similar to that of finding the digitizer reference points, the coefficients Ag1, Bg1, Cg1, Ag2, Bg2 and Cg2 can be precisely determined by specifying both the map and the grid coordinates of three linearly independent reference points. For example, the reference points may correspond to the origin and to the maximum extents of the x and y axes of the grid system. This procedure is equivalent to specifing the translation, rotation, and scale factors for the projection, but it allows these parameters to be specified in a more convenient, implicit manner.

## Digitizing a Map

SEDDS is designed to use the Interactive Graphics Design System (IGDS) by M&S Computing, Inc. At the installation located in the USGS National Center, Reston, Virginia, there are five digitizing stations connected to a PDP-11/70 computer. Each station consists of a digitizing table, terminal controls, and two CRT displays. Data files

where [xm,ym] are the map projection coordinates of a point, [phi,lamda] are the latitude and longitude (i. e., geographical coordinates) of the same point, and T is the map projection transformation.

The form of T is determined by the choice of map projection, and the values of its parameters depend upon the scale of the projection and its orientation to the Earth. Although this choice may be very important in analyzing spatial data, it is not critical in digitizing maps with SEDDS: virtually any projection may be used. However, at least an implicit solution must exist for finding Tinv, the inverse of T:

$$[phi,lamda] = Tinv[xm,ym] \qquad \text{Equation 2}$$

The vector digitizer table may be considered as a plane rectangular Cartesian coordinate system where it is possible to electronically record the location of any point [xd,yd]; this is usually performed by placing a crosshair over the point to be digitized and instructing the digitizing equipment to record the location of the crosshair, although coordinates can also be entered (less efficiently) from the digitizer keyboard. When a map is placed upon the digitizer, each point [xm(i),ym(i)] of the map coordinates corresponds to exactly one point [xd(i),yd(i)] of the digitizer coordinates. Since both the map and the digitizer employ plane rectangular Cartesian coordinate systems, the relationship between the two can be expressed by the following equations:

$$xm = A1 + B1\ xd + C1\ yd \qquad \text{Equation 3}$$

$$ym = A2 + B2\ xd + C2\ yd \qquad \text{Equation 4}$$

If both the digitizer and map coordinates are known for any set of at least three linearly independent reference points, the coefficients A1, B1, C1, A2, B2, and C2 can be found by a least squares fitting procedure. An example of a least squares procedure can be found in Bouchard and Moffitt (1965). Equations 3 and 4 can be combined to define a transformation, Tdm, of digitizer coordinates into map coordinates:

$$Tdm[xd,yd] = [A1 + B1\ xd + C1\ yd,\ A2 + B2\ xd + C2\ yd]$$

$$= [xm,ym] \qquad \text{Equation 5}$$

One very important advantage of defining the transformation, Tmg, of digitizer to map coordinates by equation 5 is that it automatically adjusts for any linear stretching of the map being digitized, as part of an implicit scaling, translation, and rotation process. Hence, the need for high quality, stable base materials is reduced.

Since most maps do not actually show the values of map coordinates, it is usually necessary to first calculate the reference point map

are stored on disks, and can be transferred to magnetic tapes. The IGDS is designed to serve many purposes, so SEDDS uses only a part of the IGDS capabilities; only those actually used are discussed in this paper.

A set of digitized data is called a design file. The IGDS internal storage format for a design file includes the following information which is relevant to SEDDS:

> Type of record: Data records can be identified as either straight line segments or as strings of connected points. Header records control the processing.
>
> Level: Files can be constructed in up to 63 levels, each amounting to a map overlay. SEDDS uses this feature to distinguish between different types of points (reference points, boundaries, etc.).
>
> Points [xd(i),yd(i)]: Digitized points used by SEDDS are output in blocks of 31.

A map is digitized by placing it securely on the digitizing table, issuing commands to create a new design file, and defining the digitizer axes, scale factors, and rotation angle. Although the logic of SEDDS does not require any specific orientation of the map with respect to the digitizer axes, it is convenient to define the axes to coincide with the boundaries of the map; this convention helps in placing the map back onto the digitizer table to make corrections. (Corrections can also be made using the SEDDS software.)

Reference points are digitized as straight line segments on level 1. This allows the operator to easily see the reference points on the CRT screen of the digitizer. Each reference point, [xd(i),yd(i)], is recorded using the cursor and is identified as the beginning of a straight line. The end point of the line is given the value of [xd(i)+1000*phi(i),yd(i)+1000*lamda(i)] using the keyboard for input. Storing the reference points in this manner assures that the reference points will remain correctly paired, and eliminates the need to store the points on a separate digitizer file or on a sheet of paper.

Outlines of map areas are digitized on levels 11 through 63. Usually, these outlines are constructed as strings of connected points as the cursor is moved around the border of the area, although straight line segments may also be used. Separate map features are digitized on different levels to assist in processing the data. For example, details of a map can be stored on different levels, with only certain levels selected for plotting or further processing. These details could include landmarks which are helpful in illustrating a map but are not necessary for analysis of the data.

Levels 6 through 10 are used to mark interior points of the map areas; this is necessary for distinguishing between the inside and the outside of the areas. The combination of a boundary and at least one interior point provides all the necessary information for processing algorithms to fill in grid areas. The use of interior points is discussed further on page 15.

Since a map may include a number of separate areas, each to be distinguished from the others, SEDDS includes a feature for assigning an identification number to each area. From each separate area, one interior point, [xint(i),yint(i)], is selected by the operator and marked with the cursor as the beginning of a straight line segment; at the other end of the line, the x coordinate is not changed (making the line vertical), but the y coordinate is used to record the identification number, Ni, as [xint(i),yint(i)+1000*Ni]. The two pairs of coordinates thus identify an interior point and the identification number, and are stored on levels 2 through 5. Use of identification points is discussed on page 14.

The IGDS permits the operator to display a digitized map on two CRT displays located at the digitizer table. Options allow the operator to selectively view each level, window areas, and to enlarge portions of a display. Because the data are stored on a disk, the operator has flexibility in correcting mistakes.

Once a design file is completed to the operator's satisfaction, it is copied from the disk to a magnetic tape. The tape is used for transferring the data to an IBM System/370 Model 155 computer for further processing. The design file is also temporarily retained on the PDP 11/70 disk so that if mistakes are found they can be corrected and reprocessed. To summarize, the information that is transferred to the IBM system is:

Level 1: Reference points.

Levels 2-5: Identification numbers for areas.

Levels 6-10: Interior points.

Levels 11-63: Boundary points.

Once a data tape has been produced by the digitizer, all processing (except for reprocessing the digitizer files to make corrections) is performed on an IBM System/370 Model 155 computer located at the USGS National Center, Reston, Virginia. All programs are written in Fortran IV, Level H, and are controlled by a catalogued procedures library. All data files referred to in the sections that follow are stored on an IBM 3330 disk, unless otherwise indicated. Processing of data through the

initial phases of digitizing is illustrated by the flow chart shown in Figure 1. Computer codes for the various procedures can be found in the appendices.

The raw digitized data are first processed by procedure, DIGIMAPS. (See page 70 for the program and procedure listing of DIGIMAPS.) DIGIMAPS reads the data, converts it from the PDP format to the IBM format, and places it in one of three files according to its level. Reference points (level 1) are placed on a file called SEDDS.DIG.CTL.&FILE, where &FILE is the map name. Identification numbers (level 2) are stored in a file called SEDDS.DIG.PNAMES.&FILE. Interior points (levels 6-10) and boundary points (levels 11-63) are both stored in a file called SEDDS.DIG.MAP.&FILE; this file is structured to retain information about the level of each point and about how strings of points are connected to each other.

The next processing step is to convert the digitized points [xd,yd] into geographical coordinates [phi,lamda], and then into coordinates for a standard map projection, [xm,ym]. The type of map projection depends upon how the data is planned to be used. For models which involve navigation, such as oilspill trajectory models, a Mercator projection is often employed because of its superior characteristics for solving navigational problems. Procedure CFDIME (convert file from digitizer to Mercator coordinates - see page 33 for a listing of CFDIME), using techniques described in chapter 2, converts data which have been digitized from a Mercator projection map into a standard Mercator coordinate system. This standard system has its origin at latitude 0 and longitude 0, and has a scale factor of unity; subroutines are available (see Appendix B) that can quickly convert these standard Mercator coordinates into geographical coordinates. It is not uncommon for data to be provided on Universal Transverse Mercator (UTM) projection maps. These maps can be converted to the standard mercator projection by procedure CFDIUTME (convert file from digitizer to UTM to Mercator - see page 37 for a listing of CFDIUTME), used in place of CFDIME. The difference between the two programs is that CFDIUTME allows for transforming the map projection; CFDIME is used when it is not necessary to transform the projection. The digitized points are first converted to UTM coordinates in a similar manner as in CFDIME; then a subroutine CUTLL (convert UTM to latitude/longitude) converts the points to geographical coordinates; finally subroutine CLLME (convert latitude/longitude to Mercator) converts the geographical coordinates to the standard Mercator system.

Programs CFDIME and CFDIUTME are designed in a modular fashion so that, by switching the map projection subroutines, they can be modified to accept data from essentially any projection and produce output in any other. Map projection subroutines are available for Mercator, Universal Transverse Mercator (UTM) and Albers equal area (continental United States) projections. These subroutines, written in FORTRAN IV language, use single-precision arithmetic to achieve high computational speeds;

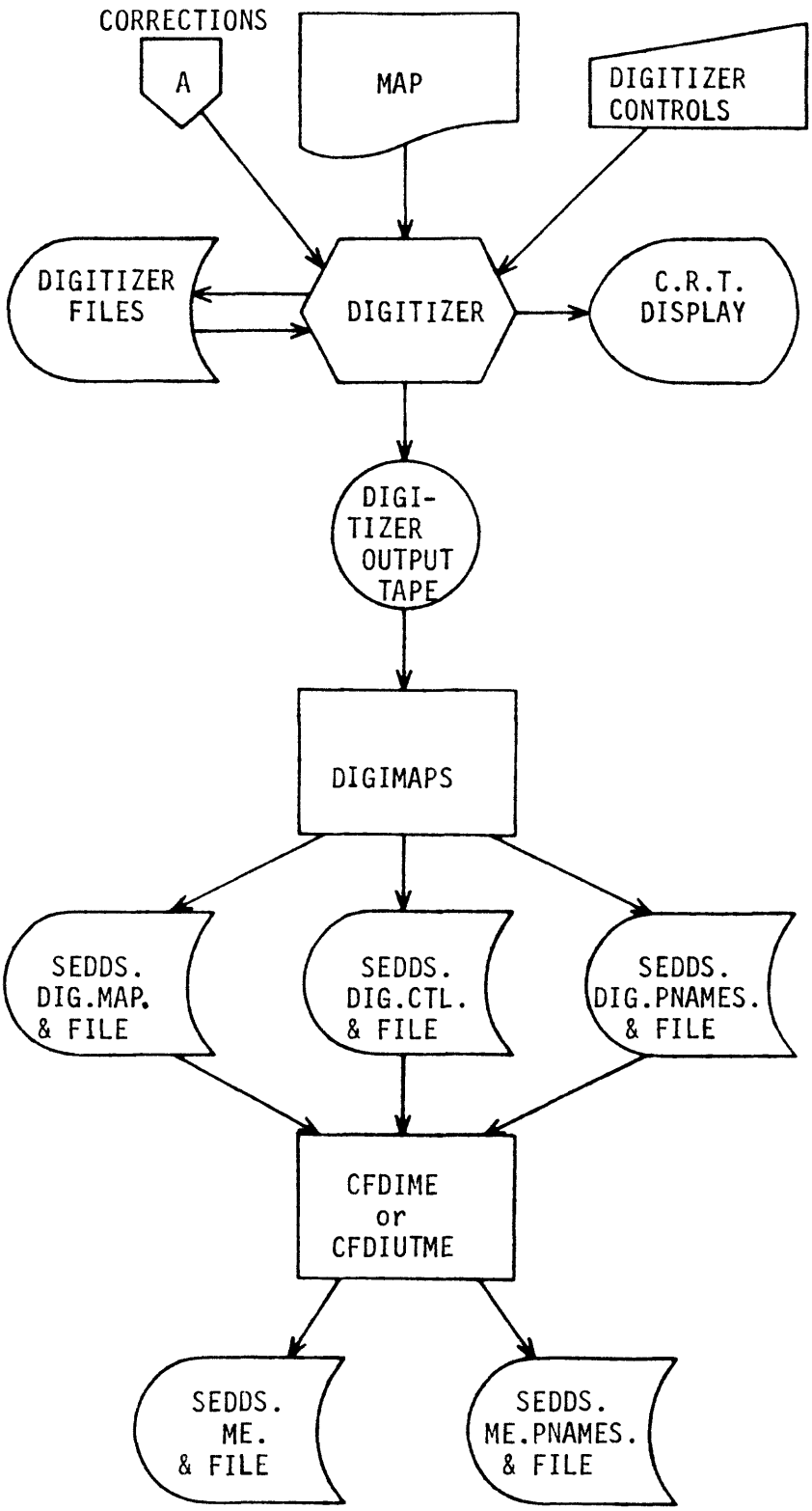


Figure 1.--Flow chart showing the sequence of operations for digitizing a map.

on an IBM System/370, Model 155 computer, each requires less than 2 milliseconds of cpu time to convert one point to or from geographical coordinates. At this rate, approximately 1 million points can be processed in 1/2 hour of cpu time. They are accurate to within 20 m on the surface of the Earth.

Procedures CFDIME or CFDIUTME store the interior and boundary points as strings of points in Mercator coordinates. They also retain the information from DIGIMAPS regarding the level of each point and how the strings are connected to one another. The converted file, called SEDDS.ME.&FILE, contains a header record for identification, the strings of points, and a trailer record of zeros; with this structure, any number of these files can be concatenated in later procedures. The identification points (if any) are also processed by CFDIME or CFDIUTME, and are stored on a file called SEDDS.ME.PNAMES.&FILE.

A critical feature of SEDDS is that the files produced by CFDIME or CFDIUTME are totally independent of the original map projection of the data. Hence, data from source maps at different types may be combined by simply processing each map through CFDIME or CFDIUTME and concatenating the output files. Furthermore, if spatial data exists in another electronic form (such as a magnetic tape containing the boundary points of counties, expressed in latitude and longitude), it is possible to write procedures to imitate the output of CFDIME, bypassing entirely the digitizing step.

It is sometimes desirable to perform analysis of data using an Albers equal area projection. Maps of the United States commonly employ this projection, using standard parallels of 29.5 degrees north and 45.5 degrees north latitude. Procedure CFDIAL (convert file from digitizer to Albers equal area - see page 29 for a listing of CFDIAL) performs exactly like CFDIME, except that it digitizes an Albers equal area map and produces a digitized file of Albers equal area coordinates. This file can be used with grids on the Albers equal area projection.

## Defining a Grid System

Analysis of spatial data is performed using a grid system superimposed over a portion of a map projection. The transformation between the map coordinates and grid coordinates is given by equation 9 (page 4). The grid used by SEDDS has a maximum dimension of 480 units on a side, with equal scale factors for each axis (i.e., square cells). The grid is completely defined by selecting appropriate coefficients for equation 9, along with maximum and minimum values for the axes.

As previously stated, the properties of certain map projections can be very useful in interpreting and analyzing spatial data. A Mercator projection, for example, is very useful for quickly solving navigational

problems, while the cells of a grid superimposed on an Albers equal-area projection will all represent approximately equal-sized areas on the surface of the Earth. The user must balance the costs of converting digitized data to a given projection with the costs of analysis using other projections.

Locating the grid with respect to the selected map projection is also very important: if the best resolution is to be obtained, the grid must be oriented to minimize wasted spaces.

The three reference points necessary for finding the coefficients of equation 9 are determined as:

$$Tmg[xm(1),ym(1)] = [0,0] \qquad \text{Equation 10}$$

$$Tmg[xm(2),ym(2)] = [xmax,0] \qquad \text{Equation 11}$$

$$Tmg[xm(3),ym(3)] = [0,ymax] \qquad \text{Equation 12}$$

To define the values xmax and ymax, let

$$xl = sqrt((xm(2)-xm(1))**2 + (ym(2)-ym(1))**2) \qquad \text{Equation 13}$$

and

$$yl = sqrt((xm(3)-xm(1))**2 + (ym(3)-ym(1))**2) \qquad \text{Equation 14}$$

Then, if $xl \geq yl$,

$$xmax = 480 \qquad \text{Equation 15}$$

$$ymax = 480 * (yl/xl) \qquad \text{Equation 16}$$

or, if $xl < yl$,

$$ymax = 480 \qquad \text{Equation 17}$$

$$xmax = 480 * (xl/yl) \qquad \text{Equation 18}$$

There remains the problem of describing the points [xm(1),ym(1)], [xm(2),ym(2)], and [xm(3),ym(3)] such that they define a rectangular area on the original map projection. If the vectors [xm(2)-xm(1),ym(2)-ym(1)] and [xm(3)-xm(1),ym(3)-ym(1)] are not perpendicular, the grid transformation will be a distortion of the original map projection. Although this is permissible, desirable qualities of the map projection could be lost.

It is easy to fulfill these conditions on a Mercator projection, since the lines of latitude and longitude are perpendicular: for example, an area defined by maximum and minimum latitudes and longitudes will be rectangular. A similar situation exists whenever the map

coordinates [xm,ym] can be readily identified and used to describe an area. For map projections where [xm,ym] are not evident - Albers equal area coordinates are almost never printed on maps - more sophisticated geometry must be used to correctly establish a rectangular grid area without distorting the original map projection.

Procedure DEFGRME (define a grid in Mercator coordinates - see page 51 for a listing of DEFGRME) performs all the calculations to define a grid over a portion of a Mercator projection map; this is illustrated in Figure 2. The output of procedure DEFGRME is a file SEDDS.&STUDY.GENERAL, which contains the limits of the grid, xmax and ymax, as well as the coefficients Ag1, Bg1, Cg1, Ag2, Bg2, and Cg2 necessary for converting from standard Mercator coordinates into grid coordinates. A similar procedure, DEFGRAL (define a grid in Albers equal area coordinates - see page 48 for a listing of DEFGRAL) exists for establishing a grid system on an Albers equal area projection of the United States.

## Converting a Digitized Map from Geographical Coordinates to Grid Coordinates

Grid coordinates provide a convenient means for plotting, checking and analyzing digitized maps. The file SEDDS.&STUDY.GENERAL, established by procedure DEFGRME contains all the information necessary for converting any map stored as Mercator coordinates into a specific Mercator grid. Procedure DEFGRAL establishes a similar file for Albers equal area projections. If a map happens to be stored in geographical coordinates or in a coordinate system other than that of the GENERAL file, it must be converted to the appropriate projection using the map conversion subroutines.

Procedure CFMAGR (Convert File from Map to Grid - see page 41 for a listing of CFMAGR) performs the actual conversion of the map file into grid coordinates, so that it can be plotted and checked; this procedure converts all points and scissors the boundaries of the map file to fit the grid boundaries. CFMAGR offers the option of converting only points on specified levels. The output of CFMAGR is a file of points called SEDDS.&STUDY.&FILE, which is actually a series of generalized plotting instructions in the form, "move to [x,y] with the pen up/down."

Procedure PLOT converts the output of CFMAGR into a complete set of plotting instructions for a Gerber 4477 plotter, using a modification of a subroutine package supplied by Gerber Scientific Instrument Corp. Since CFMAGR produces generalized plotting instructions, its output can be readily adapted to any other type of plotter by programs similar to PLOT. The use of these two procedures allows a considerable amount of flexibility on the part of the user, as illustrated by Figure 3.

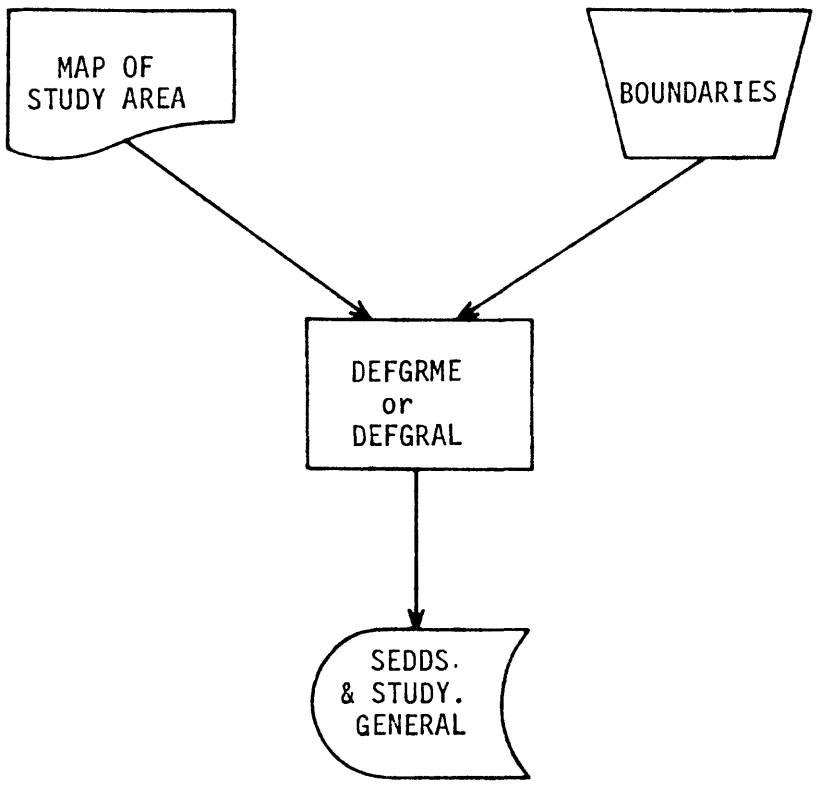


Figure 2.--Flow chart showing the sequence of operations for defining a base map.

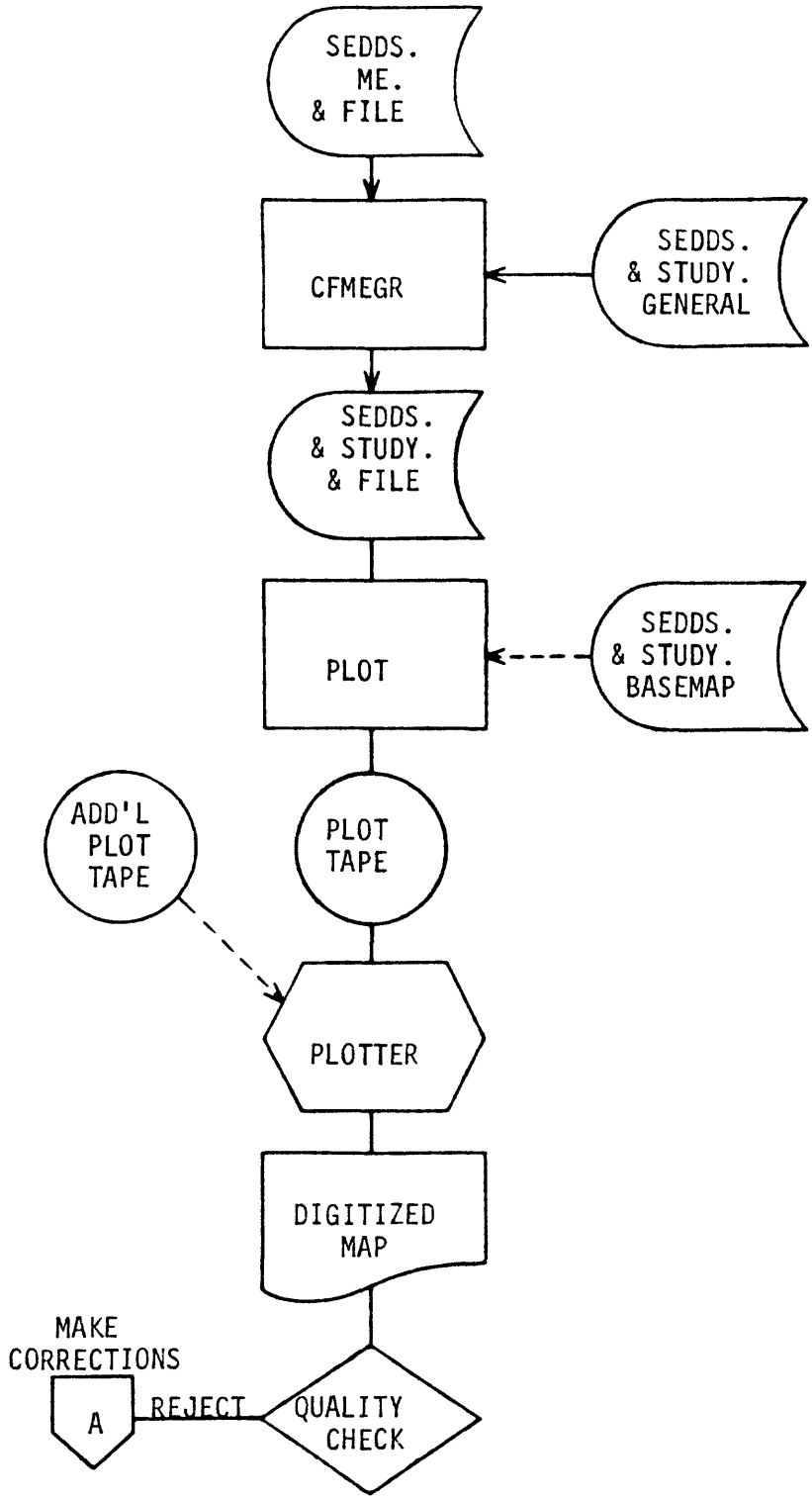


Figure 3.--Flow chart showing the sequence of operations for plotting a digitized map.

Both CFMAGR and PLOT are designed to accept concatenated files as input. Hence, any number of digitized maps can be combined into a single plot. PLOT also includes the options of plotting the output on a base map drawn from the file SEDDS.&STUDY.BASEMAP (which is produced using CFMAGR), plotting with different colors, and of displaying the grid pattern or lines of latitude and longitude (available for Mercator projection grids only). The Gerber 4477 plotter permits maps to be mechanically overlaid by plotting two or more files without changing the paper.

CFMAGR and PLOT provide SEDDS with the ability to overlay data that is derived from maps of different scales and projections. Although this is useful for spatial analysis, algorithms described in the next section provide an even more powerful analytical tool. In practice, then, the primary function of CFMAGR and PLOT is quality control: they present the digitized data in an easily understood visual form. Experience gained in operating SEDDS has demonstrated the importance of a quality control step in preventing erroneous data from causing costly failures in later analysis procedures.

## Forming a Matrix of Grid Cells

SEDDS is designed to represent spatial data in a matrix format, with each element of a matrix corresponding to one grid cell. Given the boundary of an area, and at least one interior point, all grid cells that are contained within the area can be found, and a matrix whose elements correspond to each of these cells can be marked with an identifying code. There are two methods for marking these elements: compact storage and numbered storage.

Compact storage notation utilizes all 32 bits of a standard IBM 4-byte integer; each bit indicates whether the cell is or is not contained in each of 32 different areas. Bits are numbered, high order to low order, from 0 through 31, with the 0th bit indicating the sign of the integer. For example, an element with the integer value of 9 (binary: 00000000_00000000_00000000_00000101) could mean that the corresponding grid cell is contained in areas 29 and 31. Compact storage notation allows a cell to be contained in several intersecting areas and, in effect, allows storing as many as 32 maps in a single matrix. Besides its obvious advantages in saving storage space, this notation allows rapid identification of intersecting map areas and other logical comparisons using the special functions LAND, LOR and LXOR, which are described in Appendix D.

As its name implies, numbered storage notation is used to assign specific numbers to the elements of the matrix. The advantage of this notation is that the locations of more than 32 areas can be recorded in in the same array, by assigning a unique number to each area. The disadvantage is that the areas can not intersect, since only one integer

(requiring at least 16 bits) can be assigned to each cell. Numbered storage notation is useful for recording areas such as States and counties, which normally do not overlap. Since it is rarely necessary to assign more than a few thousand identification numbers, the array used for numbered storage notation is composed of 2-byte integers, rather than the 4-byte integers of compact storage notation. Identification numbers can range from -32767 to 32767.

Compact storages matrices are formed, as illustrated in Figure 4, by the procedure TGTMATRX (target matrix - see page 82 for a listing of TGTMATRX), which reads the locations of the boundaries of each area from the files named SEDDS.ME.&FILEa, SEDDS.ME.&FILEb, etc., and turns "on" the i th bit of all the array elements which correspond to cells enclosed by each area i. Up to 32 areas are processed by a single run of TGTMATRX. The resulting array is stored as a direct access file called SEDDS.&STUDY.DA.TGTMATRX. Procedure DEFPOLY (define polygons - see page 54 for a listing of DEFPOLY) using essentially the same algorithms as TGTMATRX, forms a direct access matrix for numbered storage, as illustraged in Figure 5. Boundary locations are read from SEDDS.ME.&FILE, and the identification numbers and their locations are read from the file SEDDS.ME.PNAMES.&FILE. The output of DEFPOLY is named SEDDS.&STUDY.SEGMATRX. Both TGTMATRX and DEFPOLY use the same blocking algorithm for storing the 480 x 480 matrices; they are stored in 256 blocks, 30 x 30 cells in size. The equations for converting between full matrix elements and elements within blocks are given in Appendix E.

The algorithm that is used for marking the cells of an area, given the boundary and at least one interior point, is the same for both TGTMATRX and DEFPOLY. All cells on the boundary are first marked; next, an interior point (or, for DEFPOLY, an identification point) is selected and, moving outward from this point, all cells that can be reached without having to encounter a boundary cell are marked. When no more cells can be marked, successive interior points are examined in the same manner. The minimum number of interior points that are needed depends upon the shape of the area. A circle, for example, can be completely filled from one interior point, provided that point does not lie on a boundary cell; an hourglass with a neck less than two cells wide will require at least two interior points, one in the top and one in the bottom; areas with very irregular boundaries may require a great many interior points. In digitizing interior points, it should also be kept in mind that different choices of base maps may result in only part of an area falling within the grid system; in this case, several interior points are recommended even for simple areas, so that at least one of them will fall within the area of the base map.

If a boundary crosses any part of cell, that cell is marked as part of the area enclosed by the boundary. This has the effect of enlarging the radius of digitized areas by an average of about 0.56 times the width of a cell, with a maximum enlargement of one cell diagonal. There

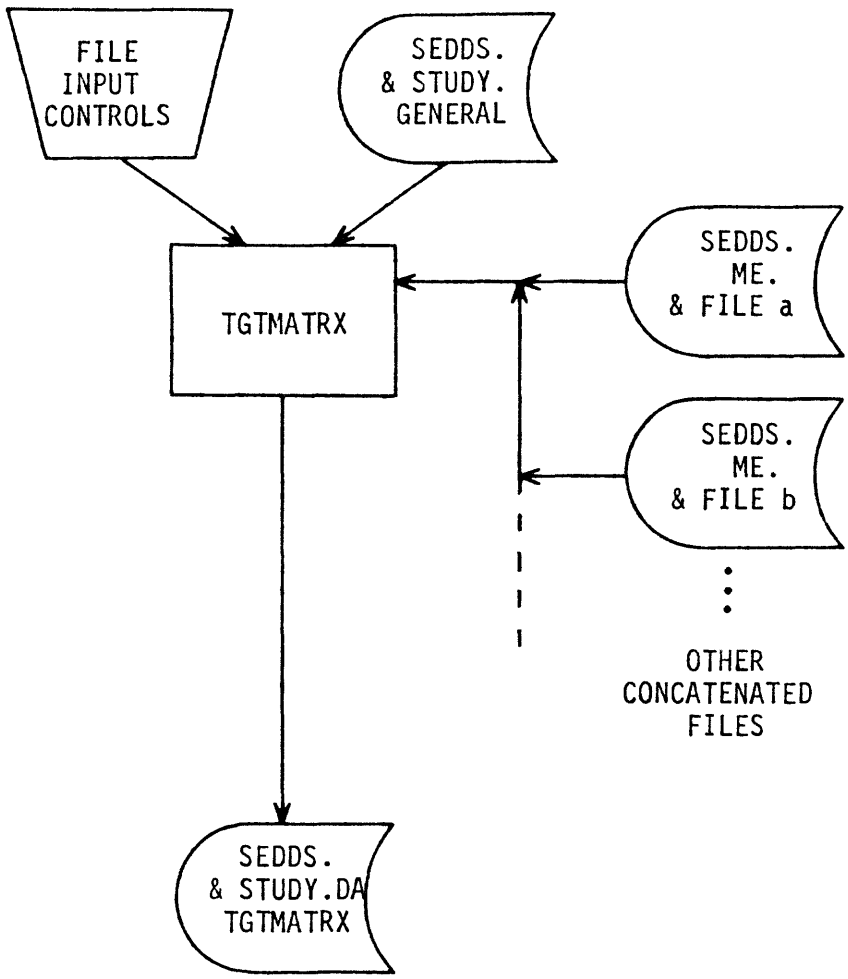


Figure 4.--Flow chart showing the sequence of operations for constructing a matrix in compact storage notation.

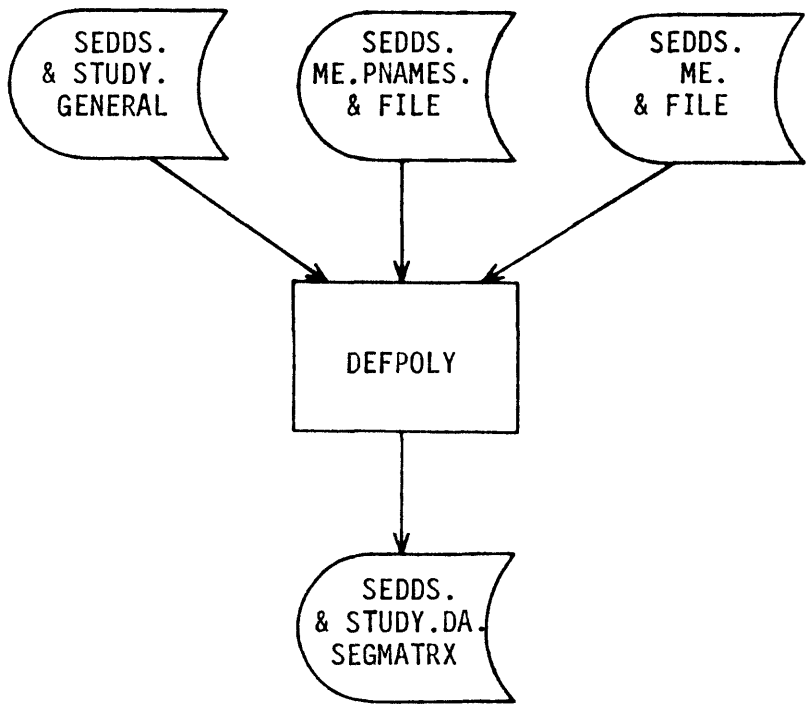


Figure 5.--Flow chart showing the sequence of operations for constructing a matrix in numbered storage notation.

is an additional boundary effect with numbered storage: cells on the boundary of two areas will be assigned the identification number of the last area processed.

It is possible to display the contents of large matrices using the subroutines PAT1, PAT2, or PAT3 to produce plots on the Gerber 4477 plotter. On the same scale as the base map, PAT1 plots a vertical line through every cell in which bit i is "on". PAT2 plots a horizontal line through each such cell. PAT3 calls both PAT1 and PAT2 to mark a cross in each cell.

## Examples of Digitizing Maps

The previous sections of this paper presented a technical description of SEDDS. This section presents examples of digitized maps that were used for an actual study, an oilspill risk analysis for proposed outer continental shelf oil and gas leases for South Atlantic OCS lease sale 56 (Samuels and Lanfear, 1980).

One of the first steps in most analyses is to define a study area. The study area for this example ranged from latitude 28 degrees north to 37 degrees north, and from longitudes 81 degrees 40 minutes west to 73 degrees west. Using these boundaries, procedure DEFGRME was run to define a grid; with square cells, this grid extends 391 units in the east-west direction and 480 units in the north-south direction. Each grid cell is approximately 2 km on a side, although this varies because of the Mercator projection.

A base map was prepared from two National Oceanic and Atmospheric Administration navigation charts, chart number 11009 (Cape Hatteras to Straights of Florida) and chart number 13003 (Cape Sable to Cape Hatteras). Each of these charts is a 1:1,200,000-scale Mercator projection; each partially overlaps the other. These maps were digitized in two separate operations with a dividing line at Ocracoke Inlet (about latitude 35 degrees north). The two files produced by two runs of CFDIME were called SEDDS.SATL56.ME.BASEMAP.NORTH and SEDDS.SATL56.ME.BASEMAP.SOUTH (SATL56 is the identification code for the analysis, and replaces the term &STUDY in previous file names). These files were concatenated and processed by CFMAGR, which produced a single file of the base map in grid coordinates, called SEDDS.SATL56.BASEMAP. Procedure PLOT read this file and prepared a plot tape for the Gerber 4477 plotter, which produced the map shown in Figure 6. Figure 7 shows the base map drawn with the grid system, rather than the lines of latitude and longitude shown in Figure 6; the two can be drawn on the same map, but the result is hard to read when reduced to page size.

Data input to the example oilspill analysis included locations of brown pelican rookeries, marked on a UTM map; this map was used for other parts of the environmental analysis, as well as the oilspill

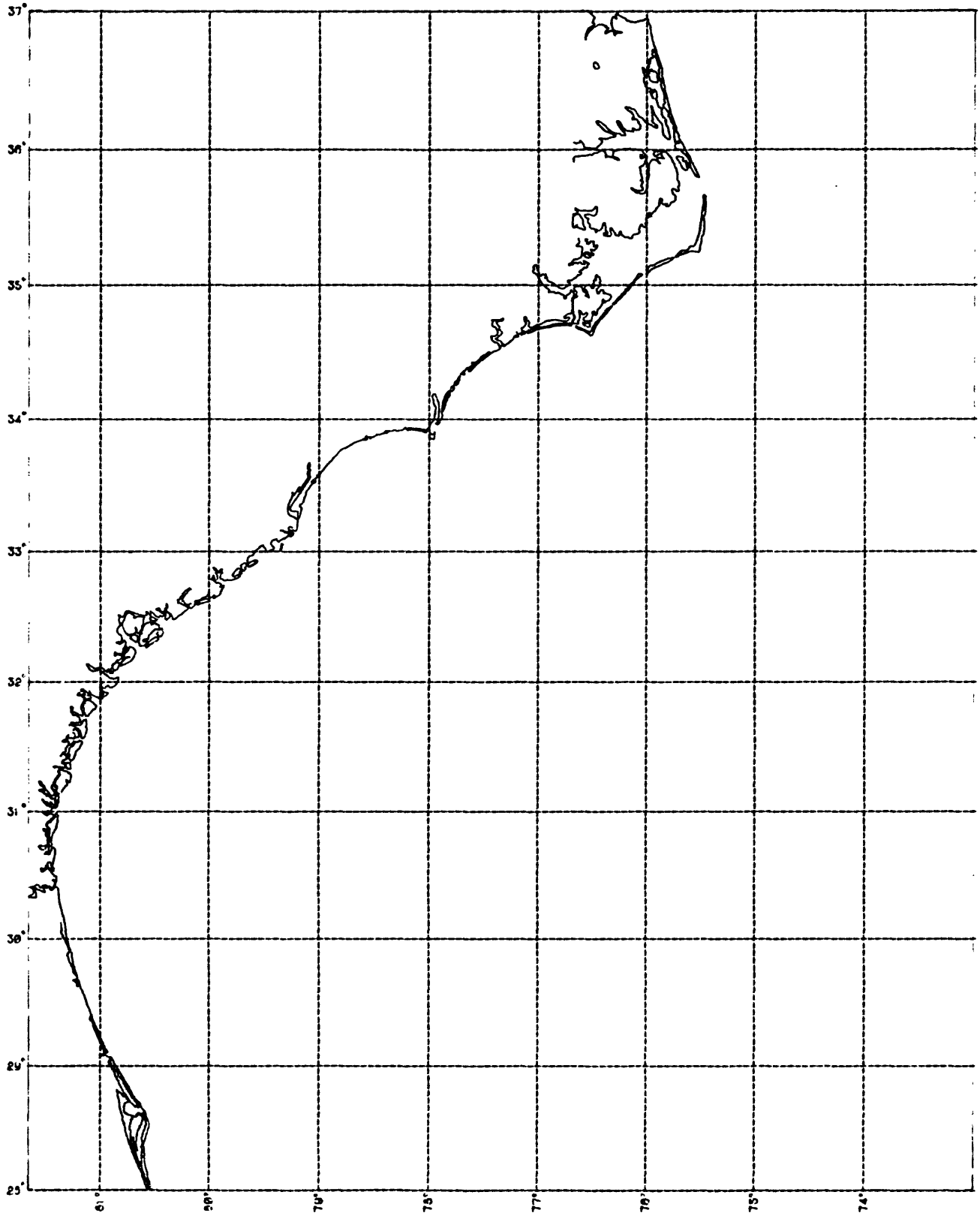


Figure 6.--Example of a digitized base map, showing the South Atlantic coastline of the United States, with lines of latitude and longitude.

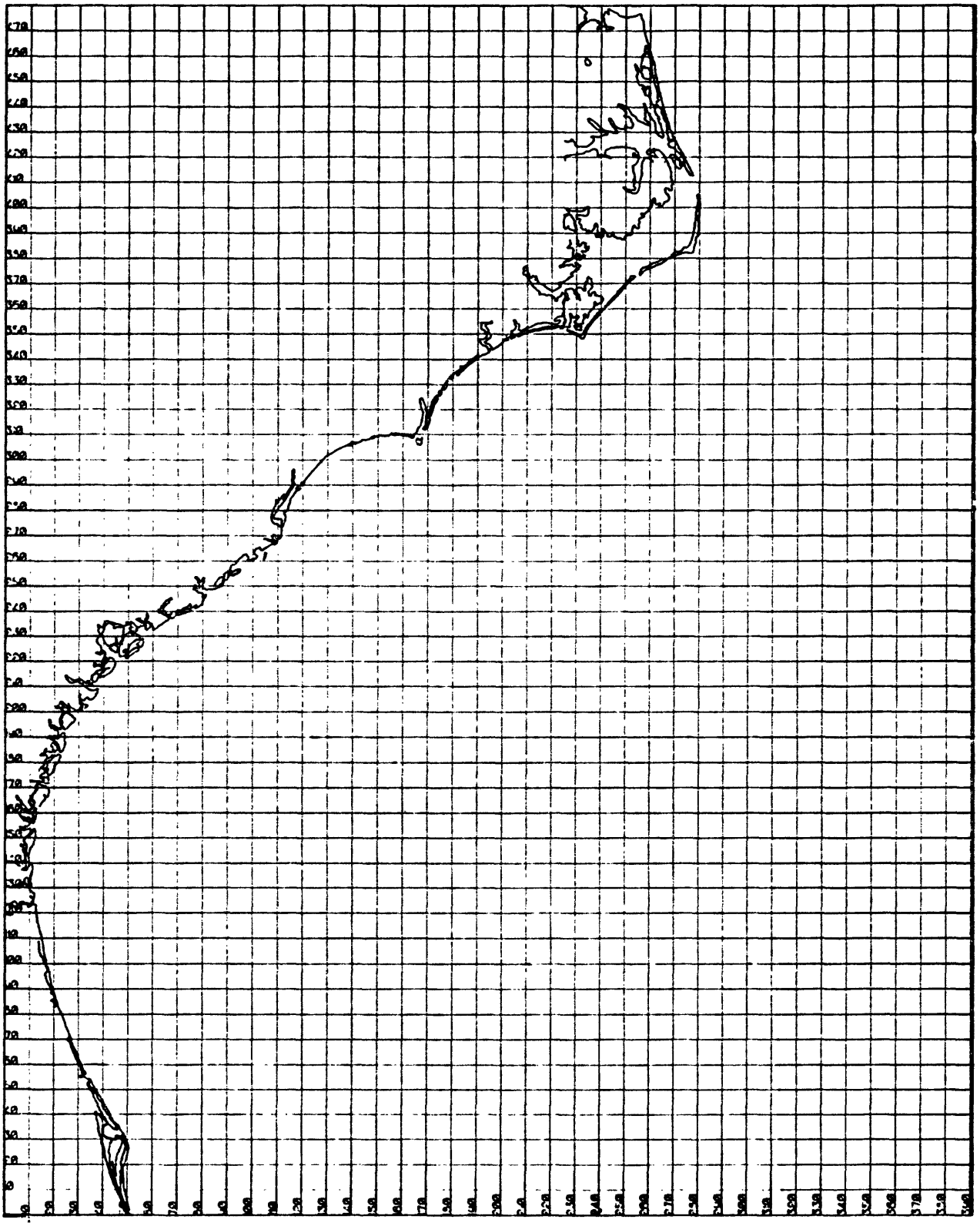

Figure 7.--Example of a digitized base map, showing the South Atlantic coastline of the United States, with grid markings at 10-cell intervals.

model. The map was digitized and processed through procedures DIGIMAPS, CFDIUTME, CFMAGR, and PLOT, and the resulting plot is shown in Figure 8, along with the base map produced earlier. Figure 9 shows the same plot, but with the addition of the interior points (see page 15); the zig-zag lines are interior points. The output of CFDIUTME, called SEDDS.SATL56.ME.PELICAN, was processed through TGTMATRX (along with other targets) to form a 480 x 480 matrix which, among other things, identified all those cells which contained a brown pelican rookery. The locations of these cells are shown in Figure 10, which was produced using subroutine PAT3.

For another part of the example analysis, it was desired to divide the coastline into different segments. Each of these segments was enclosed within a polygon, an identification number was assigned, and the digitized map was processed through DIGIMAPS, CFDIUTME, and DEFPOLY to produce a 480 x 480 matrix, in numbered storage notation, which identifies the segment to which each cell belongs. The digitized representation of these polygons is shown in Figure 11.

## Summary and Conclusions

The spatial environmental data digitizing system, SEDDS, was developed for analyzing large amounts of environmental data from a wide variety of sources. Data input can be from hand-drawn maps or from sophisticated electronic transmissions. Maps of any projection can be digitized using standard automatic techniques and stored in both line segment and raster notations. Because all data is stored independently of the original map projection, SEDDS can be used to combine data from maps of different scales and projections.

The data storage formats used by SEDDS were intended for use with mathematical models, where high-speed access to a large data base is required. It has been successfully used in the U. S. Geological Survey Oilspill Trajectory Analysis (OSTA) Model, and in a study of synthetic fuel resources.

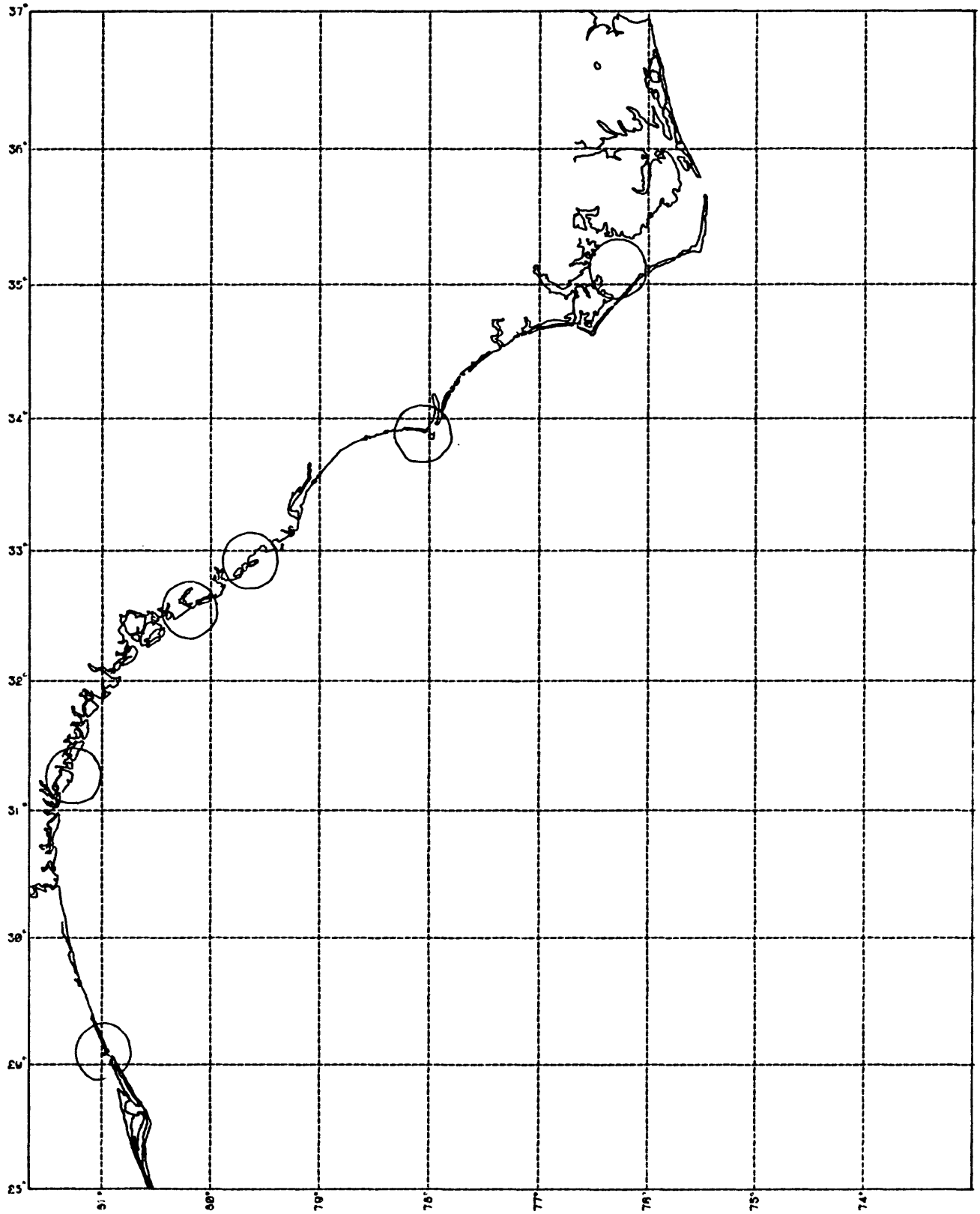


Figure 8.--Example of digitized data, showing the outlines of brown pelican rookeries on a base map of the South Atlantic coastline of the United States.

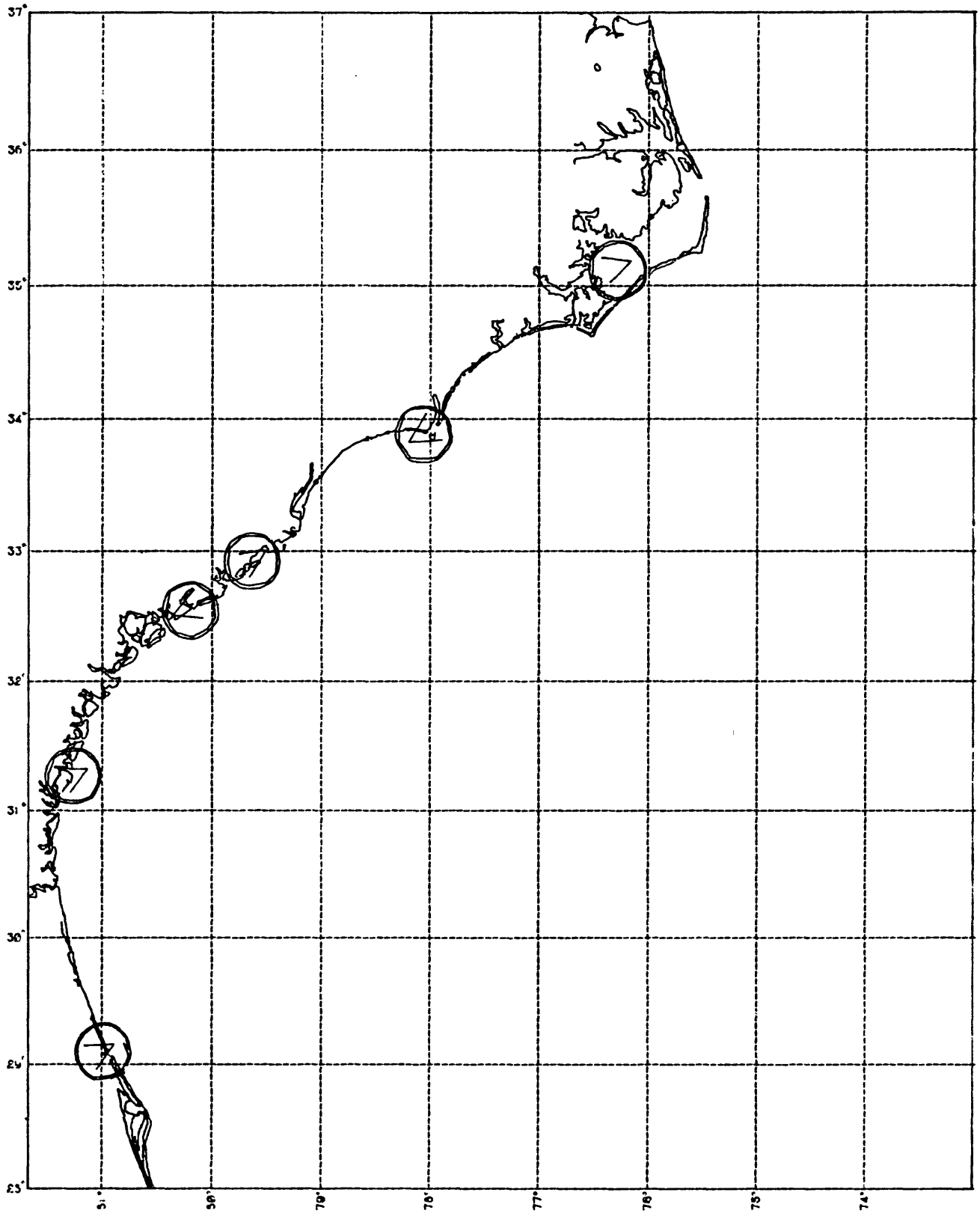


Figure 9.--Example of digitized data, showing the outlines of brown pelican rookeries, along with interior points, on a base map of the South Atlantic coastline of the United States.

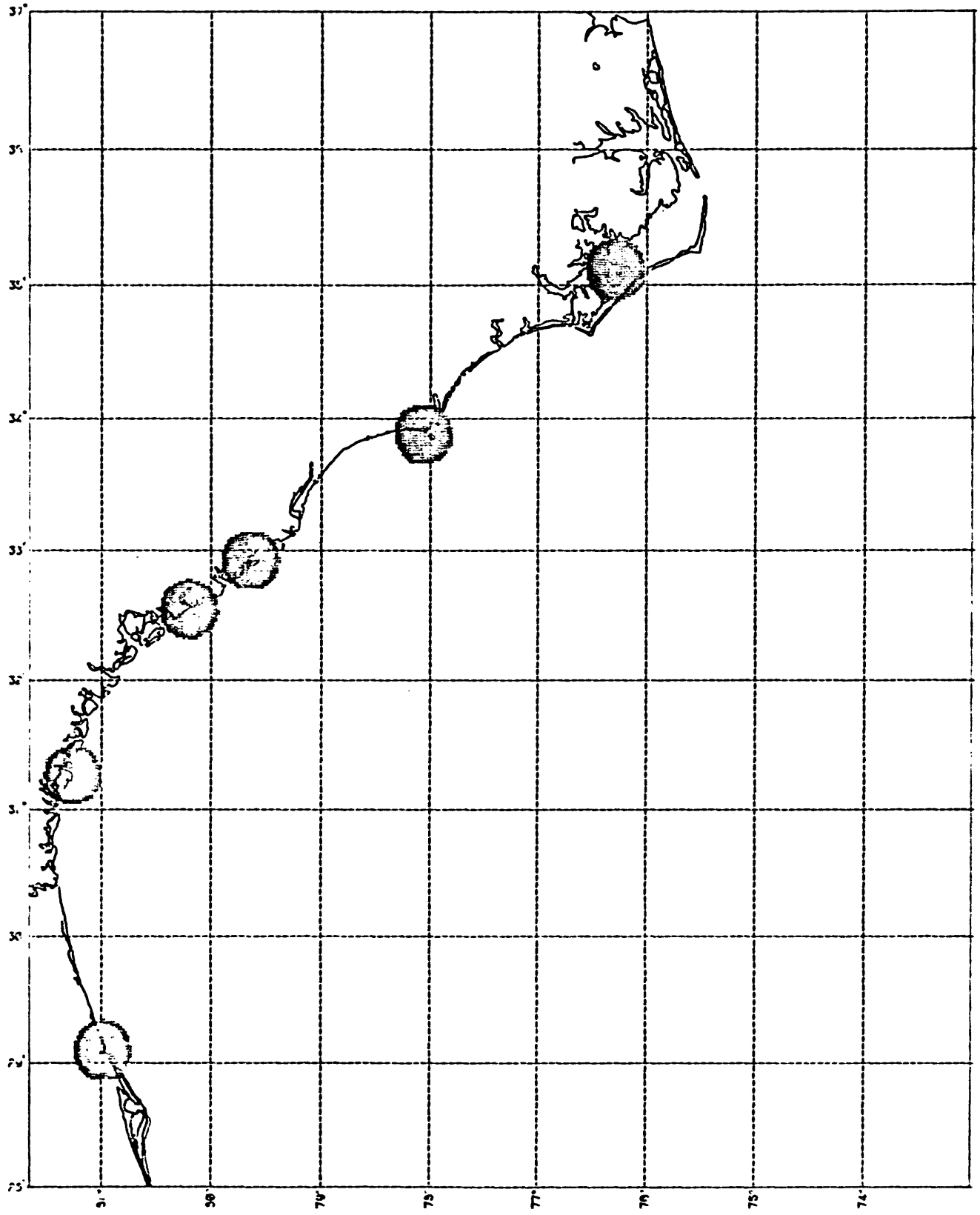

Figure 10.--Example of a plot of digitized data in compact storage notation, showing brown pelican rookeries on a base map of the South Atlantic coastline of the United States.

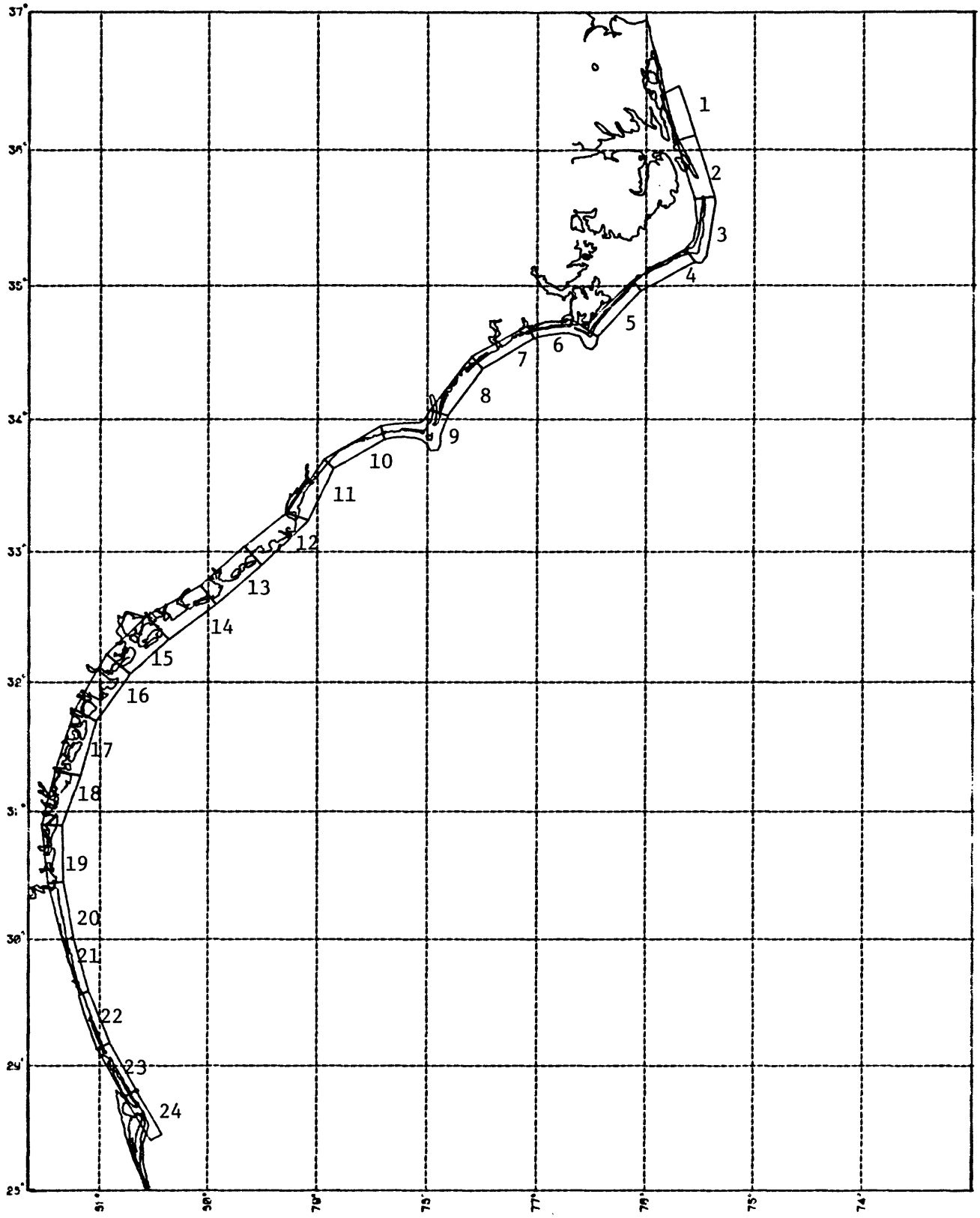


Figure 11.--Example of a plot of digitized data in numbered storage notation, showing the South Atlantic coastline of the United States divided into segments.

## References Cited

Bouchard, H., and Moffitt, F. H., 1965, Surveying: Scranton, Pennsylvania, International Textbook Company, 754 p.

Lanfear, K. J., Smith, R. A., and Slack, J. R., 1979, An introducton to the oilspill risk analysis model: Offshore Technology Conference Proceedings OTC 3607, p. 2173-2175.

Rickert, D. A., Ulman, W. J., and Hampton, E. R., 1979, Synthetic fuels development earth-science considerations: U. S. Geological Survey 45p.

Samuels, W. B., and Lanfear, K. J., 1980, An oilspill risk analysis for the South Atlantic (Proposed Sale 56) outer continental shelf lease area: U. S. Geological Survey Open-File Report 80-650, 76 p.

# Appendix A

Procedures and Programs

## Disclaimer for Distribution of Programs

Although these programs have been tested by the Geological Survey, United States Department of the Interior, no warranty, expressed or implied, is made by the Geological Survey as to the accuracy and functioning of the programs and related program material, nor shall the fact of distribution constitute any such warranty, and no responsibility is assumed by the Geological Survey in connection therewith.

Procedure: CFDIAL
Executes program: CFDIAL
Purpose: Convert the file of a digitized Albers equal area map
to Albers equal area coordinates for the United States.
Reads files: SEDDS.DIG.MAP.&FILE
SEDDS.DIG.CTL.&FILE (optional)
SEDDS.DIG.PNAMES.&FILE
Writes files: SEDDS.ME.&FILE
SEDDS.ME.PNAMES.&FILE
Text page reference: 9

Procedure listing:

```
//CFDIAL PROC UVOL=CCDXXX,U2=2
//*
//* REQUIRED: &FILE
//*
//*CLEAR THE OLD FILES, IF ANY.
// C EXEC PGM=IEFBR14
//CLEAR1  DD  UNIT=3330,VOL=SER=&UVOL,DISP=(OLD,DELETE),
// DSN=SEDDS.&STUDY..ME.&FILE
//CLEAR2  DD  UNIT=3330,VOL=SER=&UVOL,DISP=(OLD,DELETE),
// DSN=SEDDS.&STUDY..ME.PNAMES.&FILE
//G EXEC PGM=CFDIAL,TIME=&U2,REGION=120K
//STEPLIB  DD  UNIT=3330,VOL=SER=&UVOL,DISP=SHR,DSN=SEDDS.PGMLIB
//FT05F001 DD DDNAME=SYSIN
//FT06F001 DD SYSOUT=A
//FT25F001 DD UNIT=3330,VOL=SER=&UVOL,
// DISP=SHR,
// DSN=SEDDS.&STUDY..DIG.CTL.&FILE
//FT30F001 DD UNIT=3330,VOL=SER=&UVOL,
// DISP=SHR,
// DSN=SEDDS.&STUDY..DIG.MAP.&FILE
//FT40F001 DD UNIT=3330,VOL=SER=&UVOL,
// DISP=(NEW,KEEP),DCB=(RECFM=VBS,LRECL=X,BLKSIZE=6444),
// SPACE=(TRK,(10,5),RLSE),DSN=SEDDS.&STUDY..ME.&FILE
//FT45F001    DD  UNIT=3330,VOL=SER=&UVOL,
// DISP=SHR,
// DSN=SEDDS.&STUDY..DIG.PNAMES.&FILE
//FT50F001    DD  UNIT=3330,VOL=SER=&UVOL,
// DISP=(NEW,KEEP),DCB=(RECFM=VBS,LRECL=X,BLKSIZE=6444),
// SPACE=(TRK,(10,5),RLSE),DSN=SEDDS.&STUDY..ME.PNAMES.&FILE
```

Program listing:

```
C
C     PROGRAM CFDIAL
C
```

```
C     CONVERTS A FILE OF DIGITIZED COORDINATES TO COORDINATES
C        OF AN ALBERS EQUAL AREA PROJECTION.
C     XDR,YDR: DIGITIZED REFERENCE POINTS
C     XPBAR,YPBAR: MAP REFERENCE POINTS IN LONGITUDE AND LATITUDE.
C     XD,YD DIGITIZED COORDINATES
C     XP,YP ALBERS EQUAL AREA COORDINATES, USING
C        STANDARD PARALLELS FOR THE CONTINENTAL UNITED STATES.
C
C     INPUT:
C        CARD NUMBER 1:
C           COL. 01-10 BLANK
C           COL. 11-70 FILE NAME (15A4)
C
C     OPTIONAL CARD INPUT:
C        IF THE DISK FILE CONTAINS ERRORS, REFERENCE POINTS MAY BE
C        INPUT FROM CARDS.
C        CARD NUMBER 2:
C           COL. 01-06 NUMBER OF REFERENCE POINTS.
C        CARD NUMBER 3 TO N:
C           COL. 01-01 BLANK
C           COL. 02-16 DIGITIZED X REFERENCE POINT (E15.7)
C           COL. 17-17 BLANK
C           COL. 18-32 DIGITIZED Y REFERENCE POINT (E15.7)
C           COL. 33-33 BLANK.
C           COL. 34-48 LONGITUDE (E15.7)
C           COL. 49-49 BLANK.
C           COL. 50-64 LATITUDE (E15.7)
C
C
      INTEGER WHEN(7)
      REAL FILE (15)
      REAL XDR(100),YDR(100),XPBAR(100),YPBAR(100)
      REAL XD(31),YD(31),XP(31),YP(31),O(5)
      REAL XN(100),YN(100),NUMBER(100),XO(100),YO(100)
C
C     READ THE FILE NAME.
      READ (5,5001) FILE
 5001 FORMAT (10X,15A4)
C
C     FIND THE TIME.
      CALL NOW(WHEN)
C
C     WRITE THE HEADER RECORD.
      WRITE (40) FILE, WHEN
C
C     PRINT THE HEADER INFORMATION.
      WRITE (6,6001) FILE, WHEN
 6001 FORMAT (' PROGRAM CFDIAL CONVERTED THE FOLLOWING FILE FROM',
     1   ' DIGITIZED TO ALBERS EQUAL AREA COORDINATES.'//
```

```
     2   ' FILE: ',15A4//
     3   ' TIME: ',I2,A3,I2,5X,I2,':',I2,':',I2,1X,A2)
C
C     READ AND CHECK THE CONTROL POINTS.
C     XPBAR IS LATITUDE AND YPBAR IS LONGITUDE.
      READ (25,END=100) N,O,XDR,YDR,XPBAR,YPBAR
      IF (N.GT.O) GO TO 100
      IF (N.LT.O) GO TO 999
C
C     IF THE FILE CONTAINS ERRORS, READ THE CONTROL POINTS FROM CARDS.
      READ (5,1000) N
 1000 FORMAT (I6)
      READ (5,1010) (XDR(I),YDR(I),XPBAR(I),YPBAR(I),I=1,N)
 1010 FORMAT (4(1X,E15.7))
      WRITE (6,6000) N
 6000 FORMAT (' THE ',I4,'INPUT CONTROL POINTS WERE:'//
     1   '     XDR              YDR              LAT                ',
     2   'LONG ')
      WRITE (6,1020) (XDR(I),YDR(I),XPBAR(I),YPBAR(I),I=1,N)
 1020 FORMAT (' ',4(2X,E15.7))
C
  100 CONTINUE
C
C     CONVERT MAP REFERENCE POINTS TO ALBERS.
      MAP=1
      DO 20 I=1,N
      SLAT=XPBAR(I)*3600.0
      SLON=YPBAR(I)*3600.0
      IF (MAP.EQ.1) CALL CLLAL1 (SLAT,SLON,XPBAR(I),YPBAR(I),IERR)
   20 CONTINUE
      IF (MAP.EQ.1) WRITE (6,6011)
 6011 FORMAT ('OUSED ALBERS PROJECTION FOR THE CONTINENTAL U.S.')
C
C     DETERMINE COEFFICIENTS FOR A LINEAR TRANSFORMATION OF DIGITIZER
C        COORDINATES TO ALBERS EQUAL AREA COORDINATES.
      IPRT=1
      CALL CXYXYD (XDR,YDR,XPBAR,YPBAR,N,A1,B1,C1,A2,B2,C2,IPRT,IERR)
C
C     COEFFICIENTS CONVERT DIGITIZED COORDINATES TO ALBERS.
  300 CONTINUE
      READ (30,END=500) N,J,XD,YD
C     N: NUMBER OF POINTS IN BLOCK.
C     J: LEVEL.
      M=IABS(N)
      CALL CXYXY3 (XD,YD,XP,YP,A1,B1,C1,A2,B2,C2,M)
      WRITE (40) N,J,XP,YP
      GO TO 300
C
C     WRITE A BLANK RECORD TO SIGNIFY END OF FILE.
```

```
  500 CONTINUE
      N=0
      J=0
      CALL ZERO4(XP,31)
      CALL ZERO4(YP,31)
      WRITE (40) N,J,XP,YP
C
C     POLYGON CONVERSIONS.
  600 CONTINUE
      READ (45,END=99,ERR=99) JALL,XN,YN,NUMBER
      IF (JALL.EQ.0) GO TO 99
      CALL CXYXY3 (XN,YN,XO,YO,A1,B1,C1,A2,B2,C2,JALL)
      WRITE (50) JALL,XO,YO,NUMBER
      GO TO 600
C
C     ERROR MESSAGE
  999 CONTINUE
      WRITE (6,6999) N
 6999 FORMAT (' ERROR: N=',I4,' THE NUMBER OF CONTROL POINTS ',
     1   'MUST BE POSITIVE.')
   99 CONTINUE
      STOP
      END
/*
```

Procedure: CFDIME
Executes program: CFDIME
Purpose: Convert the file of a digitized Mercator map
to Mercator coordinates.
Reads files: SEDDS.DIG.MAP.&FILE
SEDDS.DIG.CTL.&FILE (optional)
SEDDS.DIG.PNAMES.&FILE (optional)
Writes file: SEDDS.ME.&FILE
Text page reference: 7

Procedure listing:

```
//CFDIME PROC UVOL=CCDXXX,U2=2
//*
//* REQUIRED: &FILE
//*
//* CLEAR THE OLD FILES, IF ANY.
// EXEC PGM=IEFBR14
//C1  DD  UNIT=3330,VOL=SER=&UVOL,DISP=(OLD,DELETE),
// DSN=SEDDS.&STUDY..ME.&FILE
// EXEC  PGM=IEFBR14
//C2  DD  UNIT=3330,VOL=SER=&UVOL,DISP=(OLD,DELETE),
// DSN=SEDDS.&STUDY..ME.PNAMES.&FILE
//G EXEC PGM=CFDIME,TIME=&U2,REGION=120K
//STEPLIB  DD  UNIT=3330,VOL=SER=&UVOL,DISP=SHR,DSN=SEDDS.PGMLIB
//FT05F001 DD DDNAME=SYSIN
//FT06F001 DD SYSOUT=A
//FT25F001 DD UNIT=3330,VOL=SER=&UVOL,
// DISP=SHR,
// DSN=SEDDS.&STUDY..DIG.CTL.&FILE
//FT30F001 DD UNIT=3330,VOL=SER=&UVOL,
// DISP=SHR,
// DSN=SEDDS.&STUDY..DIG.MAP.&FILE
//FT40F001 DD UNIT=3330,VOL=SER=&UVOL,
// DISP=(NEW,KEEP),DCB=(RECFM=VBS,LRECL=X,BLKSIZE=6444),
// SPACE=(TRK,(10,5),RLSE),DSN=SEDDS.&STUDY..ME.&FILE
//FT45F001    DD  UNIT=3330,VOL=SER=&UVOL,
// DISP=SHR,
// DSN=SEDDS.&STUDY..DIG.PNAMES.&FILE
//FT50F001    DD  UNIT=3330,VOL=SER=&UVOL,
// DISP=(NEW,KEEP),DCB=(RECFM=VBS,LRECL=X,BLKSIZE=6444),
// SPACE=(TRK,(10,5),RLSE),DSN=SEDDS.&STUDY..ME.PNAMES.&FILE
```

Program listing:

```
C     PROGRAM CFDIME
C
C     CONVERTS A FILE OF DIGITIZED COORDINATES TO MERCATOR COORDINATES
C     XDR,YDR,-  DIGITIZED REFERENCE POINTS
```

```
C     XPBAR,YPBAR - MAP REFERENCE POINTS IN LONGITUDE AND LATITUDE
C     XD,YD - DIGITIZED COORDINATES
C     XP,YP - MERCATOR COORDINATES
C
C     INPUT:
C         CARD NUMBER 1:
C         COL. 01 - 10 BLANK
C         COL. 11 - 70 FILE NAME (15A4)
C
C     OPTIONAL CARD INPUT:
C       IF DISK FILE CONTAINS ERRORS, REFERENCE POINTS ARE
C       INPUT ON CARDS
C     CARD NUMBER 2:
C       COL.01 - 06 NUMBER OF REFERENCE POINTS
C     CARD NUMBERS 3 - N:
C       COL. 01 BLANK
C       COL. 02 - 16 DIGITIZED X REFERENCE POINT (E15.7)
C       COL. 17 BLANK
C       COL. 18 - 32 DIGITIZED Y REFERENCE POINT (E15.7)
C       COL. 33 BLANK
C       COL. 34 - 48 LONGITUDE (E15.7)
C       COL. 49 BLANK
C       COL. 50 - 64 LATITUDE (E15.7)
C
      INTEGER WHEN(7)
      REAL FILE(15)
      DIMENSION XDR(100),YDR(100),XPBAR(100),YPBAR(100)
      DIMENSION XD(31),YD(31),XP(31),YP(31),O(5)
     *  ,XN(100),YN(100),NUMBER(100)
     *  ,XO(100),YO(100)
C
C     READ THE FILE NAME
      READ(5,5001)FILE
 5001 FORMAT(10X,15A4)
C
C     FIND THE TIME
      CALL NOW(WHEN)
C
C     WRITE THE HEADER RECORD
      WRITE(40)FILE,WHEN
C
C     PRINT THE HEADER INFORMATION
      WRITE(6,6001)FILE,WHEN
 6001 FORMAT(' PROGRAM CFDIME CONVERTED THE FOLLOWING FILE FROM',
     1    ' DIGITIZED TO MERCATOR COORDINATES.'//
     2    ' FILE: ',15A4//
     3    ' TIME: ',I2,A3,I2,5X,I2,':',I2,':',I2,1X,A2)
C     READ CONTROL POINTS AND CHECK THEM
      READ(25,END=100) N,O,XDR,YDR,XPBAR,YPBAR
```

```
      IF(N.GT.0) GO TO 100
      IF(N.LT.0) GO TO 999
C
C     IF FILE CONTAINS ERRORS, READ CONTROL POINTS FROM CARDS
      READ(5,1000)N
 1000 FORMAT(I6)
      READ(5,1010) (XDR(I),YDR(I),XPBAR(I),YPBAR(I), I=1,N)
 1010 FORMAT(4(1X,E15.7))
      WRITE(6,6000)N
 6000 FORMAT(' THE ',I4,'INPUT CONTROL POINTS WERE:'//
     1'     XDR              YDR              XPBAR              YPBAR')
      WRITE(6,1020) (XDR(I),YDR(I),XPBAR(I),YPBAR(I), I=1,N)
 1020 FORMAT(' ',4(2X,E15.7))
C
  100 CONTINUE
C
C     CONVERT MAP REFERENCE POINTS FROM DEGREES TO SECONDS LAT.-LONG.
      DO 20 I=1,N
      SLAT=XPBAR(I)*3600.0
      SLON=YPBAR(I)*3600.0
C
C   CONVERT SLAT AND SLON TO MERCATOR COORDINATES
      CALL CLLME (SLAT,SLON,X,Y,ERROR)
      XPBAR(I)=X
      YPBAR(I)=Y
   20 CONTINUE
C
C   DETERMINE COEFFICIENTS FOR LINEAR TRANSFORMATION OF DIGITIZER
C     COORDINATES TO MERCATOR COORDINATES
      IPRT=1
      CALL CXYXYD (XDR,YDR,XPBAR,YPBAR,N,A1,B1,C1,A2,B2,C2,IPRT,ERROR)
C
C   COEFFICIENTS A1,B1,C1,A2,B2,C2, ARE NOW DETERMINED
C     CONVERT DIGITIZED COORDINATES TO MERCATOR COORDINATES
  300 READ(30,END=500)N,J,XD,YD
      M=IABS(N)
      CALL CXYXY3 (XD,YD,XP,YP,A1,B1,C1,A2,B2,C2,M)
      WRITE(40)N,J,XP,YP
      GO TO 300
  500 CONTINUE
C     WRITE A BLANK RECORD TO INDICATE END OF FILE
      N=0
      J=0
      CALL ZERO4(XP,31)
      CALL ZERO4(YP,31)
      WRITE(40)N,J,XP,YP
  600 READ(45,END=99,ERR=99) JALL,XN,YN,NUMBER
      IF (JALL.EQ.0) GO TO 99
```

```
      CALL CXYXY3(XN,YN,XO,YO,A1,B1,C1,A2,B2,C2,JALL)
      WRITE(50) JALL,XO,YO,NUMBER
      GO TO 600
C
C     ERROR MESSAGE
  999 WRITE(6,6999)N
 6999 FORMAT(' ERROR: N=',I4,' THE NUMBER CONTROL POINTS (N)'
     1'MUST BE A POSITIVE INTEGER')
   99 STOP
      END
/*
```

Procedure: CFDIUTME
Executes program: CFDIUTME
Purpose: Convert the file of a digitized UTM map to
Mercator coordinates.
Reads files: SEDDS.DIG.MAP.&FILE
SEDDS.DIG.CTL.&FILE (optional)
SEDDS.DIG.PNAMES.&FILE (optional)
Text page reference: 7

Procedure listing:

```
//CFDIUTME PROC UVOL=CCDXXX,U2=2
//*
//* REQUIRED: &STUDY,&FILE
//*
//*CLEAR THE OLD FILES, IF ANY.
//C EXEC PGM=IEFBR14
//CLEAR1  DD  UNIT=3330,VOL=SER=&UVOL,DISP=(OLD,DELETE),
// DSN=SEDDS.&STUDY..ME.&FILE
//CLEAR2  DD  UNIT=3330,VOL=SER=&UVOL,DISP=(OLD,DELETE),
// DSN=SEDDS.&STUDY..ME.PNAMES.&FILE
//G EXEC PGM=CFDIUTME,TIME=&U2,REGION=120K
//STEPLIB  DD  UNIT=3330,VOL=SER=&UVOL,DISP=SHR,DSN=SEDDS.PGMLIB
//FT05F001 DD DDNAME=SYSIN
//FT06F001 DD SYSOUT=A
//FT25F001 DD UNIT=3330,VOL=SER=&UVOL,
// DISP=SHR,
// DSN=SEDDS.&STUDY..DIG.CTL.&FILE
//FT30F001 DD UNIT=3330,VOL=SER=&UVOL,
// DISP=SHR,
// DSN=SEDDS.&STUDY..DIG.MAP.&FILE
//FT40F001 DD UNIT=3330,VOL=SER=&UVOL,
// DISP=(NEW,KEEP),DCB=(RECFM=VBS,LRECL=X,BLKSIZE=6444),
// SPACE=(TRK,(10,5),RLSE),DSN=SEDDS.&STUDY..ME.&FILE
//FT45F001    DD  UNIT=3330,VOL=SER=&UVOL,
// DISP=SHR,
// DSN=SEDDS.&STUDY..DIG.PNAMES.&FILE
//FT50F001    DD  UNIT=3330,VOL=SER=&UVOL,
// DISP=(NEW,KEEP),DCB=(RECFM=VBS,LRECL=X,BLKSIZE=6444),
// SPACE=(TRK,(10,5),RLSE),DSN=SEDDS.&STUDY..ME.PNAMES.&FILE
```

Program listing:

```
C
C     PROGRAM CFDIUTME
C
C     CONVERTS A FILE OF DIGITIZED COORDINATES TO MERCATOR COORDINATES
C     FOR UNIVERSAL TRANSVERSE MERCATOR MAP PROJECTION
C     XDR,YDR - DIGITIZED REFERENCE POINTS
```

```
C     XPBAR,YPBAR - MAP REFERENCE POINTS IN LONGITUDE AND LATITUDE
C     XD,YD - DIGITIZED COORDINATES
C     XP,YP - UTM COORDINATES
C
C     INPUT:
C         CARD NUMBER 1:
C         COL. 01-10 BLANK
C         COL. 11-70 FILE NAME (15A4)
C
C     OPTIONAL CARD INPUT:
C     IF DISK FILE CONTAINS ERRORS, REFERENCE POINTS ARE
C     INPUT ON CARDS
C     CARD NUMBER 2:
C       COL.01-06 NUMBER OF REFERENCE POINTS
C       CARD NUMBERS 3-N:
C       COL. 01 BLANK
C       COL.02-16 DIGITIZED X REFERENCE POINT (E15.7)
C       COL. 17 BLANK
C       COL. 18-32 DIGITIZED Y REFERENCE POINT (E15.7)
C       COL. 33 BLANK
C       COL. 34-48 LONGITUDE (E15.7)
C       COL.49 BLANK
C       COL.50-64 LATITUDE (E15.7)
C
C
      INTEGER WHEN(7)
      INTEGER ZONE(100)
      REAL FILE(15)
      DIMENSION XDR(100),YDR(100),XPBAR(100),YPBAR(100)
      DIMENSION XD(31),YD(31),XP(31),YP(31),O(5)
      DIMENSION XX(31),YY(31),XSLAT(31),XSLON(31)
      DIMENSION XN(100),YN(100),NUMBER(100),XO(100),YO(100)
      DIMENSION ZSLAT(100),ZSLON(100),XZ(100),YZ(100)
C
C     READ THE FILE NAME
      READ(5,5001)FILE
 5001 FORMAT(10X,15A4)
C
C     FIND THE TIME
      CALL NOW(WHEN)
C
C     WRITE THE HEADER RECORD
      WRITE(40)FILE,WHEN
C
C     PRINT THE HEADER INFORMATION
      WRITE(6,6001)FILE,WHEN
 6001 FORMAT(' PROGRAM CFDIUTME CONVERTED THE FOLLOWING FROM',
     1   ' DIGITIZED TO MERCATOR COORDINATES.'//
     2   ' FILE: ',15A4//
```

```
     3   ' TIME: ',I2,A3,I2,5X,I2,':',I2,':',I2,1X,A2)
C     READ CONTROL POINTS AND CHECK
      READ(25,END=100) N,O,XDR,YDR,XPBAR,YPBAR
      IF(N.GT.0) GO TO 100
      IF(N.LT.0) GO TO 999
C
C     IF FILE CONTAINS ERRORS, READ CONTROL POINTS FROM CARDS
      READ(5,1000)N
 1000 FORMAT(I6)
      READ(5,1010) (XDR(I),YDR(I),XPBAR(I),YPBAR(I), I=1,N)
 1010 FORMAT(4(1X,E15.7))
      WRITE(6,6000)N
 6000 FORMAT(' THE ',I4,'INPUT CONTROL POINTS WERE:'//
     1'     XDR              YDR              XPBAR              YPBAR')
      WRITE(6,1020) (XDR(I),YDR(I),XPBAR(I),YPBAR(I), I=1,N)
 1020 FORMAT(' ',4(2X,E15.7))
C
  100 CONTINUE
C
C     CONVERT MAP REFERENCE POINTS FROM DEGREES TO SECONDS LAT.-LONG.
      DO 20 I=1,N
      SLAT=XPBAR(I)*3600.0
      SLON=YPBAR(I)*3600.0
C
C     CONVERT SLAT AND SLON TO UTM COORDINATES
      CALL CLLUT(Y,X,IZONE,SLAT,SLON,LERR)
      XPBAR(I)=X
      YPBAR(I)=Y
      ZONE(I)=IZONE
   20 CONTINUE
C     CHECK THE ZONES
      LN=N-1
      DO 25 I=1,N
      IF(I.GT.LN) GO TO 200
      IF(ZONE(I).NE.ZONE(I+1)) GO TO 9999
   25 CONTINUE
  200 CONTINUE
C
C     DETERMINE COEFFICIENTS FOR LINEAR TRANSFORMATION OF DIGITIZER
C     TO UTM COORDINATES
      IPRT=1
      CALL CXYXYD (XDR,YDR,XPBAR,YPBAR,N,A1,B1,C1,A2,B2,C2,IPRT,ERROR)
C
C   COEFFICIENTS A1,B1,C1,A2,B2,C2, ARE NOW DETERMINED
C     CONVERT DIGITIZED COORDINATES TO UTM COORDINATES
  300 READ(30,END=500)N,J,XD,YD
      M=IABS(N)
      CALL CXYXY3 (XD,YD,XP,YP,A1,B1,C1,A2,B2,C2,M)
C     CONVERT UTM COORDINATES TO LAT.-LONG.
```

```
      DO 30 I=1,M
      CALL CUTLL(YP(I),XP(I),IZONE,SLAT,SLON,LERR)
      XSLAT(I)=SLAT
      XSLON(I)=SLON
C     CONVERT LAT.-LONG. TO MERCATOR
      CALL CLLME(XSLAT(I),XSLON(I),X,Y,ERROR)
      XX(I)=X
      YY(I)=Y
   30 CONTINUE
      WRITE(40)N,J,XX,YY
      GO TO 300
  500 CONTINUE
C     WRITE A BLANK RECORD TO INDICATE END OF FILE
      N=0
      J=0
      CALL ZERO4(XX,31)
      CALL ZERO4(YY,31)
      WRITE(40)N,J,XX,YY
  600 READ(45,END=99,ERR=99) JALL,XN,YN,NUMBER
      IF (JALL.EQ.0) GO TO 99
      CALL CXYXY3(XN,YN,XO,YO,A1,B1,C1,A2,B2,C2,JALL)
      DO 40 I=1,JALL
      CALL CUTLL (YO(I),XO(I),IZONE,SLAT,SLON,LERR)
      ZSLAT(I)=SLAT
      ZSLON(I)=SLON
      CALL CLLME(ZSLAT(I),ZSLON(I),X,Y,ERROR)
      XZ(I)=X
      YZ(I)=Y
   40 CONTINUE
      WRITE(50) JALL,XZ,YZ,NUMBER
      GO TO 600
C
C     ERROR MESSAGE
  999 WRITE(6,6999)N
 6999 FORMAT(' ERROR: N=',I4,' THE NUMBER CONTROL POINTS (N)'
     1'MUST BE A POSITIVE INTEGER')
      GO TO 99
 9999 WRITE(6,6998)
 6998 FORMAT (' ',' ZONE MISMATCH')
   99 STOP
      END
/*
```

Procedure: CFMAGR
Executes program: CFMAGR
Purpose: Converts a digitized map file from Mercator or Albers equal
area coordinates into grid coordinates.
Reads file: SEDDS.ME.&FILE (May be concatenated.)
Writes file: SEDDS.&STUDY.&FILE
Text page reference: 11

Procedure listing:

```
//CFMAGR PROC UVOL=CCDXXX
//*
//* REQUIRED: &STUDY, &FILE
//*
//* CLEARS THE OLD OUTPUT FILE, IF ANY.
//C EXEC PGM=IEFBR14
//CLEAR1 DD UNIT=3330,VOL=SER=&UVOL,DISP=(OLD,DELETE),
// DSN=SEDDS.&STUDY..&FILE
//G EXEC PGM=CFMAGR,REGION=85K
//STEPLIB DD UNIT=3330,VOL=SER=&UVOL,DISP=SHR,DSN=SEDDS.PGMLIB
//FT05F001 DD DDNAME=SYSIN
//FT06F001 DD SYSOUT=A
//G.FT10F001 DD UNIT=3330,VOL=SER=&UVOL,DISP=SHR,
// DSN=SEDDS.&STUDY..GENERAL
//G.FT20F001 DD UNIT=3330,VOL=SER=&UVOL,DISP=SHR,
// DSN=SEDDS.&STUDY..ME.&FILE
//G.FT30F001 DD UNIT=3330,VOL=SER=&UVOL,
// DISP=(NEW,KEEP),DCB=(RECFM=VBS,LRECL=6440,BLKSIZE=6444),
// SPACE=(TRK,(10,5),RLSE),
// DSN=SEDDS.&STUDY..&FILE
```

Program listing:

```
C     PROGRAM CFMAGR
C
C     CONVERTS A FILE OF POINTS FROM MERCATOR COORDINATES TO THE
C        GRID COORDINATES USED FOR A PARTICULAR STUDY.
C     PACKS THE CONVERTED COORDINATES IN AN ARRAY OF 100, AND
C        INDICATES WHETHER THE PLOTTER SHOULD TO MOVE TO EACH
C        POINT WITH THE PEN UP OR DOWN.
C     POINTS OUTSIDE THE BOUNDARIES OF THE BASE MAP ARE EDITED, AND
C        WHERE NECESSARY, PLOTTING INSTRUCTIONS ARE GENERATED TO
C        CONTINUE LINES UP TO THE MAP BORDER.
C     DIGITIZER LEVELS MAY BE SELECTED FOR PLOTTING. THE DEFAULT
C        VALUES PLOT LEVELS 11 THROUGH 64.
C
C     INPUT:
C        CARD NUMBER 1:
C           COL. 01-08 STUDY (2A4)
```

```
C           COL. 09-10 BLANK
C           COL. 11-70 FILE NAME. (15A4)
C        CARD NUMBER 2 (OPTIONAL):
C           COL. 01-64 DIGITIZER LEVELS (0=EXCLUDE, 1=INCLUDE).
C           IF CARD 2 IS OMITTED, LEVELS 11-64 WILL BE PLOTTED.
C
C
      REAL XM(31),YM(31),XG(31),YG(31)
      REAL XP(100),YP(100)
      INTEGER INDEX/0/,MTOT/0/,ITOT/0/
      INTEGER IPEN(31)/-1,30*1/
      INTEGER LEVIN(63),JLEV(100)
      LOGICAL LEVEL(63)/10*.FALSE.,53*.TRUE./
      LOGICAL LPOUT/.TRUE./
      INTEGER IPN(100)
      INTEGER UIN/5/,UOUT/6/,UGEN/10/,UDIG/20/,UGRID/30/
      REAL FILE(15)
      INTEGER WHEN(7)
      REAL STUDY(2),STUDYG(2)
C
C     READ THE FILE NAME
      READ (UIN,5001) STUDY,FILE
 5001 FORMAT (2A4,2X,15A4)
C
C     READ THE LEVELS TO BE INCLUDED.
      READ (UIN,5002,END=5) LEVIN
 5002 FORMAT (63I1)
      DO 6 I=1,63
      LEVEL(I)=(LEVIN(I).NE.0)
    6 CONTINUE
    5 CONTINUE
C
C     FIND THE TIME.
      CALL NOW (WHEN)
C
C     READ THE BASEMAP BOUNDARIES AND THE GRID CONVERSION FACTORS.
      READ (UGEN) STUDYG,IX,IY,A1,B1,C1,A2,B2,C2
C     PROGRAM INTERLOCK TO PREVENT USING THE WRONG CONVERSION FACTORS.
      IF (STUDY(1).NE.STUDYG(1).OR.STUDY(2).NE.STUDYG(2)) GO TO 990
      XMAX=IX
      YMAX=IY
C
C     WRITE THE HEADER RECORD.
C     THE BOUNDARIES AND CONVERSION FACTORS ARE NOT REALLY
C     NECESSARY, BUT COULD BE HELPFUL IN RESOLVING ANY DISCREPANCIES.
      WRITE (UGRID) STUDY,FILE,WHEN,IX,IY,A1,B1,C1,A2,B2,C2
C
C     PRINT THE HEADER INFORMATION.
      WRITE (UOUT,6001) STUDY,FILE,WHEN
```

```
 6001 FORMAT ('1PROGRAM CFMAGR CONVERTED THE FOLLOWING FILE ',
     1   'TO GRID COORDINATES.'//
     2   ' STUDY: ',2A4//
     3   ' FILE: ',15A4//
     4   ' TIME:  ',I2,A3,I2,5X,I2,':',I2,':',I2,1X,A2//
     5   ' INPUT FILE(S):')
C
C     WRITE THE LEVELS THAT ARE TO BE PLOTTED.
      DO 7 I=1,63
      LEVIN(I)=0
      IF (LEVEL(I)) LEVIN(I)=I
    7 CONTINUE
      WRITE (UOUT,6004) LEVIN
 6004 FORMAT ('OTHE LEVELS MARKED FOR PLOTTING WERE:'/32I3/32I3)
C
C     READ THE HEADER RECORD OF THE DIGITIZER FILE.
   20 CONTINUE
      READ (UDIG,END=40)FILE,WHEN
      WRITE (UOUT,6003)FILE,WHEN
 6003 FORMAT ('0',15A4,', CREATED ',
     1   I2,A3,I2,5X,I2,':',I2,':',I2,1X,A2)
C
C     READ THE DIGITIZER FILE IN MERCATOR COORDINATES.
   10 CONTINUE
C        N = NUMBER OF POINTS (31 MAXIMUM).
C            POSITIVE IF THE PLOTTER SHOULD MOVE TO THE FIRST POINT
C            WITH THE PEN DOWN, NEGATIVE IF PEN UP.
C        XM,YM = CONNECTED POINTS.
      READ (UDIG,END=40) N,JL,XM,YM
      IF (N.EQ.0) GO TO 20
      M=IABS(N)
      IPEN(1)=ISGN(N)
C     TOTAL POINTS READ AS INPUT.
      MTOT=MTOT+M
C
C     CONVERT THE DIGITIZER FILE TO GRID COORDINATES.
      CALL CXYXY3 (XM,YM,XG,YG,A1,B1,C1,A2,B2,C2,M)
C
C     SEE IF THIS LEVEL IS TO BE PLOTTED.
      IF (LEVEL(JL).EQ..FALSE.) GO TO 60
C
C
C     PROCESS EACH INPUT POINT.
      DO 30 I=1,M
C
C     SEE IF THE POINT IS WITHIN THE BASE MAP BOUNDARIES.
C     NOTE: CAN EQUAL ZERO, BUT MUST BE LESS THAN MAXIMUM.
      IF (XG(I).LT.0.0.OR.XG(I).GE.XMAX.OR.YG(I).LT.0.0.OR.YG(I).GE.
     1   YMAX) GO TO 31
```

```
C
C     WITHIN BOUNDARIES. WAS THE PREVIOUS POINT OUT?
      IF (LPOUT) GO TO 32
C
C     RECORD THIS POINT.
   34 CONTINUE
      INDEX=INDEX+1
      XP(INDEX)=XG(I)
      YP(INDEX)=YG(I)
      JLEV(INDEX)=JL
      IPN(INDEX)=IPEN(I)
C     SAVE THE LAST POINT WITHIN THE BOUNDARIES.
      XIN=XG(I)
      YIN=YG(I)
C     PRINT IN BLOCKS OF 100.
      IF (INDEX.LT.100) GO TO 30
      ITOT=ITOT+100
      WRITE (UGRID) INDEX,IPN,JLEV,XP,YP
      INDEX=0
      GO TO 30
C
C     THE LINE IS RETURNING FROM OUT OF BOUNDS.
   32 CONTINUE
      LPOUT=.FALSE.
C     DO NOT INTERPOLATE A BOUNDARY POINT IF RETURNING WITH THE PEN UP.
      IF (IPEN(I).LT.0) GO TO 34
C     INTERPOLATE A BOUNDARY POINT.
      CALL BOUNDS (XG(I),YG(I),XOUT,YOUT,XMAX,YMAX,XBND,YBND)
C     RECORD THE BOUNDARY POINT.
      INDEX=INDEX+1
      XP(INDEX)=XBND
      YP(INDEX)=YBND
      JLEV(INDEX)=JL
C     MOVE TO THE BOUNDARY POINT WITH THE PEN UP.
      IPN(INDEX)=-1
      IF (INDEX.LT.100) GO TO 34
      ITOT=ITOT+100
      WRITE (UGRID) INDEX,IPN,JLEV,XP,YP
      INDEX=0
      GO TO 34
C
C     THE LINE IS OUT OF BOUNDS.
   31 CONTINUE
C     SAVE THE LAST POINT OUT OF BOUNDS.
      XOUT=XG(I)
      YOUT=YG(I)
C     DO NOT INTERPOLATE A BOUNDARY POINT IF THE PREVIOUS POINT WAS
C        OUT, OR IF THE PEN IS UP.
      IF (LPOUT) GO TO 30
```

```
      LPOUT=.TRUE.
      IF (IPEN(I).LT.0) GO TO 30
C     INTERPOLATE A BOUNDARY POINT.
      CALL BOUNDS (XIN,YIN,XOUT,YOUT,XMAX,YMAX,XBND,YBND)
C     RECORD THE BOUNDARY POINT.
      INDEX=INDEX+1
      XP(INDEX)=XBND
      YP(INDEX)=YBND
      JLEV(INDEX)=JL
      IPN(INDEX)=1
      IF (INDEX.LT.100) GO TO 30
      ITOT=ITOT+100
      WRITE (UGRID) INDEX,IPN,JLEV,XP,YP
      INDEX=0
C
C     END PROCESSING FOR THIS INPUT POINT.
   30 CONTINUE
C
C     GO BACK FOR ANOTHER SET OF POINTS.
      GO TO 10
C
C     PROCESS POINTS THAT ARE NOT TO BE PLOTTED.
   60 CONTINUE
      DO 68 I=1,M
C     EXCLUDE POINTS THAT ARE OUT OF BOUNDS.
      IF (XG(I).LT.0.0.OR.XG(I).GE.XMAX.OR.YG(I).LT.0.0.OR.YG(I).GE.
     1   YMAX) GO TO 68
C     POINT IS IN BOUNDS. RECORD IT WITH IPN = 0.
      INDEX=INDEX+1
      XP(INDEX)=XG(I)
      YP(INDEX)=YG(I)
      JLEV(INDEX)=JL
      IPN(INDEX)=0
      IF (INDEX.LT.100) GO TO 68
      ITOT=ITOT+100
      WRITE (UGRID) INDEX,IPN,JLEV,XP,YP
      INDEX=0
   68 CONTINUE
C
C     GO BACK FOR ANOTHER SET OF POINTS.
      GO TO 10
C
C
C     END OF DIGITIZED DATA
   40 CONTINUE
C
C     DRAIN THE BUFFER.
      IF (INDEX.EQ.0) GO TO 51
      ISTRT=INDEX+1
```

```
      DO 50 I=ISTRT,100
      XP(I)=0.0
      YP(I)=0.0
      JLEV(I)=0
      IPN(I)=0
   50 CONTINUE
      ITOT=ITOT+INDEX
      WRITE (UGRID) INDEX,IPN,JLEV,XP,YP
C
C     WRITE A BLANK RECORD TO SIGNAL THE END OF THIS FILE..
C     SINCE THE FILES MAY BE CONCATENATED, THE ZERO VALUE FOR INDEX IS
C     NEEDED TO WARN THAT THE NEXT RECORD (IF ANY) WILL BE A HEADER
C     FOR THE NEXT FILE.
   51 CONTINUE
      INDEX=0
      CALL ZERO4 (XP,100)
      CALL ZERO4 (YP,100)
      CALL ZERO4 (JLEV,100)
      CALL ZERO4 (IPN,100)
      WRITE (UGRID) INDEX,IPN,JLEV,XP,YP
C
C     WRITE THE TOTAL RECORDS READ AND WRITTEN.
      WRITE (UOUT,6002) MTOT,ITOT
 6002 FORMAT ('0INPUT POINTS:  ',I8//
     1    ' OUTPUT POINTS: ',I8)
      GO TO 99
C
C
C     ERROR MESSAGE - FILE MISMATCH.
  990 CONTINUE
      WRITE (UOUT,6990) STUDY,STUDYG
 6990 FORMAT ('0ERROR - THE STUDY REQUESTED ON CARD, ',2A4,
     1    ' DID NOT MATCH THE STUDY ON THE GENERAL FILE, ',2A4,'.')
C
   99 CONTINUE
      STOP
      END
C
C
      SUBROUTINE BOUNDS (XIN,YIN,XOUT,YOUT,XMAX,YMAX,XBND,YBND)
C
C     FINDS WHERE A LINE BETWEEN TWO POINTS CROSSES THE
C     BOUNDARY OF A MAP.
C
C     THIS VERSION IS INTENDED SPECIFICALLY FOR PROGRAM CFMAGR.
C
C
C     CHECK FOR A VERTICAL LINE.
      IF (XOUT.EQ.XIN) GO TO 10
```

```
C
C     FIND THE POSSIBLE BOUNDARY IN THE X DIRECTION.
      XBND=0.0
      IF (XIN.LT.XOUT) XBND=XMAX-0.001
      YBND=YIN+(YOUT-YIN)*(XBND-XIN)/(XOUT-XIN)
C     CHECK FOR A NEAR-VERTICAL LINE.
      CALL DVCHK(J)
      IF (J.EQ.1) GO TO 10
C     SEE IF (XBND,YBND) IS WITHIN THE MAP.
      IF (YBND.GE.0.0.AND.YBND.LE.YMAX) RETURN
C
C     NO. THE BOUNDARY POINT IS IN THE Y DIRECTION.
C     CHECK FOR A HORIZONTAL LINE.
      IF (YOUT.EQ.YIN) GO TO 30
      YBND=0.0
      IF (YIN.LT.YOUT) YBND=YMAX-0.001
      XBND=XIN+(XOUT-XIN)*(YBND-YIN)/(YOUT-YIN)
C     CHECK FOR A NEAR-HORIZONTAL LINE.
      CALL DVCHK(J)
      IF (J.EQ.1) GO TO 30
      RETURN
C
C     VERTICAL OR NEAR-VERTICAL LINE.
   10 CONTINUE
      XBND=XIN
      YBND=YOUT
      IF (YOUT.LT.0.0) YBND=0.0
      IF (YOUT.GE.YMAX) YBND=YMAX-0.001
      RETURN
C
C     HORIZONTAL OR NEAR-HORIZONTAL LINE.
   30 CONTINUE
      YBND=YIN
      XBND=XOUT
      IF (XOUT.LT.0.0) XBND=0.0
      IF (XOUT.GE.XMAX) XBND=XMAX-0.001
      RETURN
      END
/*
```

Procedure: DEFGRAL
Executes program: DEFGRAL
Purpose: To define a grid on an Albers equal area projection.
Reads files: none
Writes file: SEDDS.&STUDY.GENERAL
Text page reference: 11

Procedure listing:

```
//DEFGRAL PROC D=CCDXXX
//*
//* REQUIRED: &STUDY
//*
//* CLEAR THE OLD FILE, IF ANY.
//C EXEC PGM=IEFBR14
//CLEAR DD UNIT=3330,VOL=SER=&D,DISP=(OLD,DELETE),
// DSN=SEDDS.&STUDY..GENERAL
//G EXEC PGM=DEFGRAL
//STEPLIB DD UNIT=3330,VOL=SER=&D,DISP=SHR,DSN=SEDDS.PGMLIB
//FT05F001 DD DDNAME=SYSIN
//FT06F001 DD SYSOUT=A
//FT10F001 DD UNIT=3330,VOL=SER=&D,DISP=(NEW,KEEP),
// DCB=(RECFM=VBS,LRECL=6440,BLKSIZE=6444),SPACE=(TRK,(1,1),RLSE),
// DSN=SEDDS.&STUDY..GENERAL
```

Program listing:

```
C     PROGRAM DEFGRAL.
C     DEFINES THE GRID IN ALBERS COORDINATES.
C
C     THE GRID IS DEFINED BY SPECIFING BOUNDARIES OF LATITUDE AND
C        LONGITUDE. THE SPECIFIED AREA WILL BE CENTERED ON THE GRID,
C        AND THE GRID WILL EXACTLY COVER ALL OF THE AREA. THE GRID
C        WILL ALSO INCLUDE SOME AREA NOT SPECIFIED, AND THIS WILL HAVE
C        TO BE REMOVED OR CHECKED FOR BY OTHER PROGRAMS.

      REAL STUDY(2)
      REAL X(100),Y(100),XG(100),YG(100)
      REAL LAMO,LAMB,LAMT
      WRITE (6,6000)
 6000 FORMAT (' PROGRAM DEFGRAL.')
C
C     READ THE LIMITS OF THE GRID.
      READ (5,5001) STUDY,SLATMI,SLATMA,SLONMI,SLONMA
 5001 FORMAT (2A4,2X,4F10.2)
C
C     CONVERT DEGREES TO SECONDS.
      SLATMI=SLATMI*3600.0
      SLATMA=SLATMA*3600.0
```

```
      SLONMI=SLONMI*3600.0
      SLONMA=SLONMA*3600.0
C
C     CONVERT LAT/LON TO ALBERS.
      CALL CLLAL1 (SLATMI,SLONMA,X1,Y1,IERR)
      CALL CLLAL1 (SLATMA,SLONMA,X2,Y2,IERR)
      CALL CLLAL1 (SLATMA,SLONMI,X3,Y3,IERR)
      CALL CLLAL1 (SLATMI,SLONMI,X4,Y4,IERR)
C
C     FIND THE DIMENSIONS OF THE GRID, IN ALBERS COORDINATES.
      CALL CLLAL1 (SLATMI,(SLONMA+SLONMI)/2.0,X5,Y5,IERR)
      XV1=X5-0.5*(X1+X4)
      YV1=Y5-0.5*(Y1+Y4)
      XO=X1+XV1
      YO=Y1+YV1
      XR=X4+XV1
      YB=Y4+YV1
      XL=XO+0.5*(X2+X3)-X5
      YT=YO+0.5*(Y2+Y3)-Y5
      WRITE (6,6001) XO,YO,XR,YB,XL,YT
 6001 FORMAT (' THE THREE CORNER POINTS ARE:'/
     1   ' LOWER LEFT:  ',2E16.7/
     2   ' LOWER RIGHT: ',2E16.7/
     3   ' UPPER LEFT:  ',2E16.7)
C
C     PLACE THE CORNER POINTS INTO VECTORS FOR CXYXY1.
      X(1)=XO
      Y(1)=YO
      X(2)=XR
      Y(2)=YB
      X(3)=XL
      Y(3)=YT
C
C     DEFINE THE CORNER POINTS OF THE GRID. ONE SIDE WILL BE 480 CELLS.
      XSCALE=SQRT((XR-XO)**2+(YB-YO)**2)/480.0
      YSCALE=SQRT((XL-XO)**2+(YT-YO)**2)/480.0
      IF (XSCALE.LT.YSCALE) GO TO 10
      XG(2)=480.0
      YG(2)=0.0
      XG(3)=0.0
      YG(3)=480.0*YSCALE/XSCALE
      GO TO 12
   10 CONTINUE
      XG(2)=480.0*XSCALE/YSCALE
      YG(2)=0.0
      XG(3)=0.0
      YG(3)=480.0
   12 CONTINUE
      XG(1)=0.0
```

```
      YG(1)=0.0
C
C     DETERMINE THE SCALE OF THE GRID (KM PER CELL).
      SCALE=SQRT((XR-XO)**2+(YB-YO)**2)/(1000.0*XG(2))
      WRITE (6,6002) SCALE
 6002 FORMAT ('OTHE SCALE IS ',F8.3,' KILOMETERS PER CELL.')
C
C     FIND THE TRANSFORMATION FOR ALBERS INTO GRID.
      N=3
      IPRT=1
      CALL CXYXYD (X,Y,XG,YG,N,A1,B1,C1,A2,B2,C2,IPRT,IERR)
C
C     CALCULATE THE CORNER POINTS OF THE GRID IN LAT/LON.
      CALL CALLL1 (XO,YO,PHIO,LAMO,IERR)
      CALL CALLL1 (XR,YB,PHIR,LAMB,IERR)
      CALL CALLL1 (XL,YT,PHIL,LAMT,IERR)
      PHIO=PHIO/3600.0
      LAMO=LAMO/3600.0
      PHIR=PHIR/3600.0
      LAMB=LAMB/3600.0
      PHIL=PHIL/3600.0
      LAMT=LAMT/3600.0
      WRITE (6,6003) XG(1),YG(1),PHIO,LAMO,XG(2),YG(2),PHIR,LAMB,
     1   XG(3),YG(3),PHIL,LAMT
 6003 FORMAT ('OTHE MATRIX HAS THE FOLLOWING BOUNDARIES:'/
     1 (' (',F6.2,',',F6.2,') EQUALS LAT ',F6.2,' LONG ',F6.2))
C
C     LIMITS OF THE GRID.
      IX=XG(2)
      IY=YG(3)
C
C     WRITE THE GRID DEFINITION ON A DISK.
      WRITE (10) STUDY,IX,IY,A1,B1,C1,A2,B2,C2,
     1   SLATMI,SLATMA,SLONMI,SLONMA
      STOP
      END
/*
```

Procedure: DEFGRME
Executes program: DEFGRME
Purpose: Define a grid on a Mercator projection.
Reads file: none
Writes file: SEDDS.&STUDY.GENERAL
Text page reference: 11

Procedure listing:

```
//DEFGRME PROC D=CCDXXX
//*
//* REQUIRED: &STUDY
//*
//* CLEAR THE OLD FILE, IF ANY.
//C EXEC PGM=IEFBR14
//CLEAR DD UNIT=3330,VOL=SER=&D,DISP=(OLD,DELETE),
// DSN=SEDDS.&STUDY..GENERAL
//G EXEC PGM=DEFGRME
//STEPLIB DD UNIT=3330,VOL=SER=&D,DISP=SHR,DSN=SEDDS.PGMLIB
//FT05F001 DD DDNAME=SYSIN
//FT06F001 DD SYSOUT=A
//FT10F001 DD UNIT=3330,VOL=SER=&D,DISP=(NEW,KEEP),
// DCB=(RECFM=VBS,LRECL=6440,BLKSIZE=6444),SPACE=(TRK,(1,1),RLSE),
// DSN=SEDDS.&STUDY..GENERAL
```

Program listing:

```
C     PROGRAM DEFGRME
C
C     DEFINES THE LIMITS OF THE BASE MAP, USING A MERCATOR PROJECTION,
C        AND ESTABLISHES THE APPROPRIATE GRID CONVERSIONS.
C     THE GRID IS ALWAYS ALIGNED WITH THE X-AXIS EAST-WEST (EAST
C        POSITIVE) AND THE Y-AXIS NORTH-SOUTH (NORTH POSITIVE). THE
C        LONGER AXIS IS ASSIGNED A LENGTH OF 480 GRID UNITS, AND THE
C        OTHER AXIS IS TRUNCATED TO AN INTEGER GRID VALUE.
C
C     INPUT:
C        CARD NUMBER 1.
C           COL. 01-08 STUDY (A4)
C           COL. 09-10 BLANK
C           COL. 11-30 LATITUDE BOUNDARIES, DEGREES (2F10.2)
C           COL. 31-50 LONGITUDE BOUNDARIES, DEGREES (2F10.2)
C
C
      REAL XM(3),YM(3),XG(3),YG(3)
      REAL STUDY(2)
      INTEGER UIN/5/,UOUT/6/,UGEN/10/
C
C     READ THE LIMITS OF LATITUDE AND LONGITUDE.
```

```
      READ (UIN,5001) STUDY,SLAT1,SLAT2,SLON1,SLON2
 5001 FORMAT (2A4,2X,4F10.2)
C
C     FIND MAXIMUMS AND MINIMUMS, AND CONVERT TO SECONDS.
      SLATMA=3600.0*AMAX1(SLAT1,SLAT2)
      SLATMI=3600.0*AMIN1(SLAT1,SLAT2)
      SLONMA=3600.0*AMAX1(SLON1,SLON2)
      SLONMI=3600.0*AMIN1(SLON1,SLON2)
C
C     FIND MERCATOR COORDINATES OF THE ORIGIN (POINT1), THE END POINT
C        OF THE X-AXIS (POINT2), AND THE END POINT OF THE Y-AXIS
C        (POINT3). THESE ARE THE REFERENCE POINTS.
      CALL CLLME (SLATMI,SLONMA,XM(1),YM(1),IERR)
      IF (IERR.NE.0) WRITE (UOUT,6099)
      CALL CLLME (SLATMI,SLONMI,XM(2),YM(2),IERR)
      IF (IERR.NE.0) WRITE (UOUT,6099)
      CALL CLLME (SLATMA,SLONMA,XM(3),YM(3),IERR)
      IF (IERR.NE.0) WRITE (UOUT,6099)
C
C     THE FOLLOWING VALUES ARE ALWAYS THE SAME.
      XG(1)=0.0
      YG(1)=0.0
      XG(3)=0.0
      YG(2)=0.0
C
C     SEE WHICH AXIS IS LONGER.
      IF ((ABS(XM(2)-XM(1))).LT.(ABS(YM(3)-YM(1)))) GO TO 20
C
C     THE X-AXIS IS LONGER. PROPORTION THE Y-AXIS.
      XG(2)=480.0
      Y=ABS((YM(3)-YM(1))*480.0/(XM(2)-XM(1)))
      IY=Y
      YG(3)=IY
C     RECOMPUTE THE MERCATOR AND LAT-LONG VALUE FOR THE TRUNCATED AXIS.
      YM(3)=YM(1)+ABS(YG(3)/Y)*(YM(3)-YM(1))
      CALL CMELL (XM(3),YM(3),SLATMA,DUMMY,IERR)
      GO TO 30
C
C     THE Y-AXIS IS LONGER. PROPORTION THE X-AXIS.
   20 CONTINUE
      YG(3)=480.0
      X=ABS((XM(2)-XM(1))*480.0/(YM(3)-YM(1)))
      IX=X
      XG(2)=IX
C     RECOMPUTE THE MERCATOR AND LAT-LONG VALUE FOR THE TRUNCATED AXIS.
      XM(2)=XM(1)+ABS(XG(2)/X)*(XM(2)-XM(1))
      CALL CMELL (XM(2),YM(2),DUMMY,SLONMI,IERR)
      IF (IERR.NE.0) WRITE (UOUT,6099)
C
```

```
C     CONVERT LAT-LONG BACK TO DEGREES.
   30 CONTINUE
      SLATMI=SLATMI/3600.0
      SLATMA=SLATMA/3600.0
      SLONMI=SLONMI/3600.0
      SLONMA=SLONMA/3600.0
C
C     FIND THE CONVERSION COEFFICIENTS FROM MERCATOR TO GRID.
C     USES THE SAME FORMAT AS CXYXY1.
      A1=-XG(2)*XM(1)/(XM(2)-XM(1))
      B1=XG(2)/(XM(2)-XM(1))
      C1=0.0
      A2=-YG(3)*YM(1)/(YM(3)-YM(1))
      B2=0.0
      C2=YG(3)/(YM(3)-YM(1))
C
C     WRITE THE GRID DIMENSIONS,CONVERSION COEFFICIENTS, AND THE
C        LATITUDE-LONGITUDE BOUNDARIES.
      IX=XG(2)
      IY=YG(3)
      WRITE (UGEN) STUDY,IX,IY,A1,B1,C1,A2,B2,C2,
     1   SLATMI,SLATMA,SLONMI,SLONMA
C
C     PRINT THE REFERENCE POINTS.
      WRITE (UOUT,6001) STUDY
 6001 FORMAT ('1MERCATOR GRID DEFINITION FOR ',2A4,'.'/
     *   ' COMPUTED BY PROGRAM DEFGRME, VERSION 1.0'/
     1   ' THE REFERENCE POINTS ARE AS FOLLOWS:'//
     2   11X,'LATITUDE  LONGITUDE    MERCATOR COORDINATES    GRID ',
     3   'COORDINATES'/11X,'(DEGREES) (DEGREES)',7X,'X',11X,'Y',11X,
     4   'X',6X,'Y')
      WRITE (UOUT,6002)
     1   SLATMI,SLONMA,XM(1),YM(1),XG(1),YG(1),
     2   SLATMI,SLONMI,XM(2),YM(2),XG(2),YG(2),
     3   SLATMA,SLONMA,XM(3),YM(3),XG(3),YG(3)
 6002 FORMAT (
     1   'OPOINT 1   ',F9.4,2X,F9.4,4X,F9.0,2X,F9.0,5X,F5.0,2X,F5.0/
     2   'OPOINT 2   ',F9.4,2X,F9.4,4X,F9.0,2X,F9.0,5X,F5.0,2X,F5.0/
     3   'OPOINT 3   ',F9.4,2X,F9.4,4X,F9.0,2X,F9.0,5X,F5.0,2X,F5.0)
 6099 FORMAT ('0********** WARNING - ERROR IN MERCATOR CONVERSION ',
     1   'ROUTINES. **********')
      STOP
      END
/*
```

Procedure: DEFPOLY
Executes program: DEFPOLY
Purpose: Form a matrix in numbered storage notation.
Reads files: SEDDS.ME.&FILE
SEDDS.ME.PNAMES.&FILE
Writes file: SEDDS.&sale.DA.SEGMATRX
Page reference: 15

Procedure listing:

```
//DEFPOLY PROC TM=3,RG=640K,D=CCD836,
// FILEOUT=NULLFILE,PTAPE=LLFILE,V=1
//C EXEC PGM=IEFBR14
//CLEAR1 DD UNIT=3330,VOL=SER=&D,DISP=(OLD,DELETE),
// DSN=SEDDS.&STUDY..DEFPOLY.IDS.&FILE
//CLEAR2 DD UNIT=3330,VOL=SER=&D,DISP=(OLD,DELETE),
// DSN=SEDDS.&STUDY..DEFPOLY.DRAX.&FILE
//A EXEC PGM=DEFPOLY1,TIME=&TM,REGION=&RG
//STEPLIB DD UNIT=3330,VOL=SER=&D,DISP=SHR,DSN=SEDDS.PGMLIB
//FT05F001 DD DDNAME=SYSIN
//FT06F001 DD SYSOUT=A
//FT20F001 DD UNIT=3330,VOL=SER=&D,DISP=SHR,
// DSN=SEDDS.&STUDY..GENERAL
//FT40F001 DD UNIT=3330,VOL=SER=&D,DISP=SHR,
// DSN=SEDDS.&STUDY..ME.&FILE
//FT50F001 DD UNIT=3330,VOL=SER=&D,DISP=SHR,
// DSN=SEDDS.&STUDY..ME.PNAMES.&FILE
//FT60F001 DD UNIT=3330,VOL=SER=&D,DISP=(NEW,KEEP),
// DCB=(RECFM=VBS,LRECL=X,BLKSIZE=6444),SPACE=(TRK,(10,5),RLSE),
// DSN=SEDDS.&STUDY..DEFPOLY.IDS.&FILE
//FT75F001 DD UNIT=3330,VOL=SER=&D,
// DCB=(RECFM=VBS,LRECL=404,BLKSIZE=408),SPACE=(TRK,(5,5),RLSE),
// DSN=&&TRASH
//FT90F001 DD UNIT=3330,VOL=SER=&D,DISP=(NEW,KEEP),
// DCB=DSORD=DA,SPACE=(1800,(256)),
// DSN=SEDDS.&STUDY..DEFPOLY.DRAX.&FILE
//FT99F001 DD UNIT=3330,VOL=SER=&D,
// DCB=(RECFM=VBS,LRECL=X,BLKSIZE=6444),SPACE=(TRK,(50,50),RLSE),
// DSN=&&BIGMAT
```

Program listing:

```
C                       PROGRAM DEFPOLY1
C
C                   CREATES A SET OF DISJOINT POLYGONS STORED IN NUM.
C        INPUT:
C                   CARD INPUT:  THE FIRST CARD HAS A FOUR DIGIT INTEGER.
C                   IF ZERO, WE GET MINIMAL OUTPUT. IF POSITIVE AND LESS
C                   THAN TEN, WE GET PRINTOUT DURING THE MANIPULATIONS
```

```
C                  OF NUM.  IF TEN OR MORE, WE GET PRINTOUT INFO FROM
C                  PPLOT2 WHILE A PLOT IS GENERATED.
C                  THE SECOND CARD (OPTIONAL) HAS FOUR I4 FIELDS.  THEY
C                  SPECIFY, IN ORDER, THE MINIMUM LEVEL AT WHICH WE
C                  HAVE DIGITIZED BOUNDARY POINTS, IF WE WANT TO STORE
C                  NUM IN A DIRECT ACCESS SET, IF WE WANT TO PLOT NUM,
C                  AND IF WE WANT TO FILL ANY 'HOLES' THAT MAY HAVE
C                  OCCURRED IN NUM.
C
C                  FILE20: WE READ THE STUDY NAME,THE DIMENSIONS OF THE
C                  GRID AND THE COEFFICIENTS NEEDED TO PRODUCE THE GRID
C                  GRID COORDINATES.
C                  FILE50 : FROM FILE 50 NAMEIT READS A SEQUENCE OF
C                  POINTS (MERCATOR COORDINATES) AND THEIR NUMBERS.
C
C                  FILE 40: WE READ THE MERCATOR COORDINATES OF THE
C                  BOUNDARY POINTS.
C       OUTPUT:
C                  FILE 60 : PPLOT2 OUTPUTS ON FILE60 A SEQUENCE OF
C                  POINTS (GRID COORDINATES) AND THEIR #.
C
C                  FILE90 : IS A DIRECT ACCESS FILE ON WHICH STUFIT
C                  WRITES NUM IN 30 BY 30 BLOCKS.
C
C                  FILE 99 : WE WRITE OUT THE MATRIX NUM.
      LOGICAL*1 BUG
      INTEGER*2 NUM
      COMMON /COEFF/ A1,B1,C1,A2,B2,C2
      COMMON /DEBUG/ BUG
      COMMON /DIMEN/IXLOW,IYLOW,IXUP,IYUP
      COMMON /MATRIX/ NUM(500,500)
      EQUIVALENCE    (XO(2),XI(1))     ,    (YO(2),YI(1))
      DIMENSION XO(32),XI(31),YO(32),YI(31)
      DIMENSION STUDY(2)
      READ(5,5000) KNOW
      READ(20) STUDY,IX,IY,A1,B1,C1,A2,B2,C2
      WRITE(6,6000) STUDY,IX,IY,A1,B1,C1,A2,B2,C2
      IXLOW=10
      IYLOW=10
      IXUP=IX+IXLOW
      IYUP=IY+IYLOW
      A1=A1+IXLOW
      A2=A2+IYLOW
      WRITE(6,6000)STUDY,IXUP,IYUP,A1,B1,C1,A2,B2,C2
      READ(5,5000,ERR=10,END=10)     LEVMIN,LSTUF,LPLOT,LHOLE
   10 CONTINUE
      IF (LEVMIN.LE.0) LEVMIN=11
      WRITE(6,5000) LEVMIN,LSTUF,LPLOT,LHOLE
      CALL ZERO2(NUM,250000)
```

```
      MALL = 0
      READ(40)
      NREC = 1
      BUG = (KNOW.GE.7)
  100 CONTINUE
C                   READ FROM 40 TILL YOU FIND THE BEGINNING OF A BOUNDARY.
      NREC = NREC+1
      READ(40,END=900,ERR=990) N,LEVEL,XI,YI
      IF (LEVEL.LT.LEVMIN) GO TO 100
      M = IABS(N)
      CALL CXYXY3(XI,YI,XI,YI,A1,B1,C1,A2,B2,C2,M)
      IF  (BUG)       WRITE(6,6200) N,((XI(L),YI(L)),L=1,M)
C                   GO TRACE THE BOUNDARY OF THE POLYGON.
      IF (N.LT.0) CALL TRACK1(XI,YI,M)
      IF (N.GT.0) CALL TRACK1(XO,YO,M+1)
C                   THE STRING MAY CONSIST OF MULTIPLE RECORDS.  THERE-
C                   FORE, STORE THE LAST POINT.
      XO(1) = XI(M)
      YO(1) = YI(M)
      GO TO 100
  900 CONTINUE
C                   DO YOU NEED TO PATCH SOME POLYGON BOUNDARIES ?
      CALL PATCH1
      BUG = (KNOW.GT.0)
C                   GO GET THE NAMES OF THE POLYGONS.
      CALL NAMEIT
C                   ATTACH UNAFFILIATED BOUNDARY POINTS TO THE NEAREST
C                   POLYGON
      IF (LHOLE.NE.0) CALL UNHOLE
      CALL PATCH3
      IF (LSTUF.NE.0) CALL STUFIT
      BUG = (KNOW.GE.10)
      IF (LPLOT.NE.0) CALL PPLOT2
      STOP
  990 CONTINUE
      WRITE(6,6100) NREC
      STOP
 5000 FORMAT( 10I4)
 6000 FORMAT(2X,2A4, 2I4,6F16.4)
 6100 FORMAT( T12,'ERROR IN UNIT 40 WHILE READING RECORD',I5,' .')
 6200 FORMAT( T10,I6 / 8(2X,F12.2))
 7000 FORMAT( T12,'MALL =',I4 / 12(2X,F8.3))
      STOP
      END
      SUBROUTINE ACTION(I,J,KVAL)
      INTEGER*2 IX,IY
      INTEGER*2 NUM
      COMMON /MATRIX/NUM(500,500)
      COMMON /STACK/ IX(2500),IY(2500),IN,IOUT
```

```
      WRITE(6,1000) I,J,KVAL
 1000 FORMAT(T20,'ACTION I,J,KVAL',10I5)
      IX(1) = I
      IY(1) = J
      IN = 1
      IOUT = 2
      NUM(I,J) = KVAL
      CALL FILIN3(0)
      RETURN
      END
      SUBROUTINE CRUMBS
      INTEGER*2 LX,LY
      INTEGER*2 IX,IY
      INTEGER*2 NUM
      COMMON /MATRIX/ NUM(500,500)
      COMMON /STACK/ IX(2500),IY(2500),IN,IOUT
      COMMON /SHELF/ LX(100) ,LY(100) ,LMANY,LREC
      WRITE(6,1000) LREC,LMANY
      IF(LMANY.EQ.1) GO TO 300
      KMANY = LMANY-1
      DO 100 I=1,KMANY
      IX(I) = LX(I)
      IY(I) = LY(I)
      WRITE(6,2000) IX(I),IY(I),NUM(IX(I),IY(I))
  100 CONTINUE
      IN = 1
      IOUT = LMANY
      CALL FILIN3(0)
  300 IF (LREC.EQ.0) GO TO 900
      REWIND 75
      DO 500 I=1,LREC
      READ(75) LX,LY
      DO 400 I=1,100
      IX(I) = LX(I)
      IY(I) = LY(I)
      WRITE(6,2000) IX(I),IY(I),NUM(IX(I),IY(I))
  400 CONTINUE
      IN = 1
      IOUT = 101
      CALL FILIN3(0)
  500 CONTINUE
  900 CONTINUE
      RETURN
 2000 FORMAT( T20,'CRUMBS : IX,IY, AND NUM ARE = ',5I6)
 1000 FORMAT( T20,'CRUMBS WAS ENTERED WITH LREC AND LMANY EQUAL TO',
     *  10I5)
      END
      SUBROUTINE DEFER(I,J,KVAL)
      INTEGER*2 LX,LY
```

```
      INTEGER*2 NUM
      COMMON /MATRIX/NUM(500,500)
      COMMON /SHELF/ LX(100) ,LY(100) ,LMANY,LREC
      NUM(I,J) = KVAL
      IF (LMANY.LE.100) GO TO 200
      WRITE(75) LX,LY
      LMANY = 1
      LREC = LREC+1
  200 CONTINUE
      LX(LMANY) = I
      LY(LMANY) = J
      LMANY = LMANY+1
      RETURN
      END
      SUBROUTINE FILIN3(IVAL)
C                  SUBROUTINE FILIN3 USES A CIRCULAR QUEUE IN ORDER
C                  TO NAME OUR POLYGON.  AT EACH STEP WE REMOVE THE
C                  FIRST ELEMENT AND EXAMINE ITS NEIGHBORS.  THOSE
C                  WHICH HAVE NO ID ( NUM NOT POSITIVE) ARE GIVEN THE
C                  ID OF THE POLYGON.  MOREOVER WE ADD ONTO THE QUEUE
C                  THOSE NEIGHBORS WHOSE NUM VALUE WAS (PRIOR TO THE
C                  RENAMING) ZERO.  WHEN THE QUEUE EMPTIES, WE ARE DONE.
      INTEGER*2 NUMBER
      INTEGER*2 IX,IY,NUM
      COMMON /STACK/ IX(2500),IY(2500),IN,IOUT
      COMMON /MATRIX/NUM(500,500)
      COMMON /DIMEN/IXLOW,IYLOW,IXUP,IYUP
      MX=IXUP+10
      NY=IYUP+10
      NMAX = 2500
      LALL = 0
  100 IF (IN.EQ.IOUT) GO TO 999
      K = IX(IN)
      L = IY(IN)
      NUMBER = NUM(K,L)
      IF (IN.EQ.NMAX) LALL=LALL+NMAX
      IF (IN.EQ.NMAX) WRITE(6,6100) IOUT
      IN = IN+1
      IF (IN.GT.NMAX) IN = 1
      IF (K.EQ.1) GO TO 200
      IF (NUM(K-1,L).GT.IVAL) GO TO 200
      IF (NUM(K-1,L).LT.IVAL) GO TO 150
      IX(IOUT) = K-1
      IY(IOUT) = L
      IOUT = IOUT+1
      IF (IOUT.GT.NMAX) IOUT=1
      IF (IOUT.EQ.IN) GO TO 800
  150 NUM(K-1,L) = NUMBER
  200 IF (L.EQ.1) GO TO 300
```

```
      IF (NUM(K,L-1).GT.IVAL) GO TO 300
      IF (NUM(K,L-1).LT.IVAL) GO TO 250
      IX(IOUT) = K
      IY(IOUT) = L-1
      IOUT = IOUT+1
      IF (IOUT.GT.NMAX) IOUT=1
      IF (IOUT.EQ.IN) GO TO 800
  250 NUM(K,L-1) = NUMBER
  300 IF (K.EQ.MX) GO TO 400
      IF (NUM(K+1,L).GT.IVAL) GO TO 400
      IF (NUM(K+1,L).LT.IVAL) GO TO 350
      IX(IOUT) = K+1
      IY(IOUT) = L
      IOUT = IOUT+1
      IF (IOUT.GT.NMAX) IOUT=1
      IF (IOUT.EQ.IN) GO TO 800
  350 NUM(K+1,L) = NUMBER
  400 IF (L.GE.NY) GO TO 100
      IF (NUM(K,L+1).GT.IVAL) GO TO 100
      IF (NUM(K,L+1).LT.IVAL) GO TO 450
      IX(IOUT) = K
      IY(IOUT) = L+1
      IOUT = IOUT+1
      IF (IOUT.GT.NMAX) IOUT=1
      IF (IOUT.EQ.IN) GO TO 800
  450 NUM(K,L+1) = NUMBER
      GO TO 100
  800 WRITE(6,6000) IN,IOUT,IX(IN),IY(IN)
      IN = IN+1
      IF (IN.GT.NMAX) IN = 1
      GO TO 100
  999 CONTINUE
      LALL = LALL+IN-1
      IF (IVAL.GE.0) WRITE(6,6200) NUMBER,LALL
      RETURN
 6000 FORMAT( T12,'DANGER OF OVERFLOWING. IN AND IOUT ARE',2I5 ,
     *  '. AND THE POINT(',I4,',',I4,') WAS DITCHED')
 6100 FORMAT( T12,'IN EQUALS NMAX AND IOUT IS',I5)
 6200 FORMAT(T20,'FILIN3 : POLYGON #',I5,' HAS',I6,' POINTS.')
      END
      SUBROUTINE GET60(XN,YN,NUMBER,KREC)
      COMMON /COEFF/A1,B1,C1,A2,B2,C2
      DIMENSION XN(100),YN(100),NUMBER(100)
      KREC = 0
      MREC = 0
  100 CONTINUE
C                   READ SOME POINTS AND THEIR ID NUMBERS.
C                   IF END OF FILE OR IF NO POINTS, YOU ARE DONE.
      READ(50,ERR=980,END=980) MP,XN,YN,NUMBER
```

```
      IF (MP.LE.0) GO TO 980
      KREC = KREC+1
C                  GET THE GRID COORDINATES.
      CALL CXYXY3(XN,YN,XN,YN,A1,B1,C1,A2,B2,C2,MP)
      IF (KREC.GE.MREC) CALL PATCH2(XN,YN,NUMBER,KREC,MREC)
      WRITE(60) MP,XN,YN,NUMBER
      GO TO 100
  980 CONTINUE
      RETURN
      END
      SUBROUTINE MARGIN
      COMMON /DIMEN/ IXLOW,IYLOW,IXUP,IYUP
      DIMENSION BX(5),BY(5)
      BX(1) = IXLOW
      BX(2)=IXUP
      BX(3)=IXUP
      BX(4) = IXLOW
      BX(5) = IXLOW
      BY(1) = IYLOW
      BY(2) = IYLOW
      BY(3)=IYUP
      BY(4)=IYUP
      BY(5) = IYLOW
      CALL INIT(1)
      CALL SCAL(0.05,0.05)
      B=1.0-IXLOW*0.05
      C=1.0-IYLOW*0.05
      CALL ZER(B,C)
      CALL FONT2(3.0,3.0,0.0)
      CALL WAIT
      CALL LINES(BX,BY,5,0,0)
      CALL WAIT
      RETURN
      END
      SUBROUTINE NAMEIT
C                  FILLS THE INTERIOR AND BOUNDARY BLOCKS OF EACH
C                  POLYGON WITH THE POLYGON'S ID #.
      INTEGER*2 NUM
      INTEGER*2 LX,LY
      COMMON /COEFF/ A1,B1,C1,A2,B2,C2
      COMMON /MATRIX/ NUM(500,500)
      COMMON /SHELF/ LX(100) ,LY(100) ,LMANY,LREC
      DIMENSION XN(100),YN(100),NUMBER(100)
      DATA MX,NY/500,500/
      LMANY = 1
      LREC = 0
      CALL GET60(XN,YN,NUMBER,KREC)
      REWIND 60
      DO 300 IREC = 1,KREC
```

```
      READ(60) MP,XN,YN,NUMBER
      DO 200 K=1,MP
      I = XN(K)+1
C                   MAKE SURE THE POINT IS ON THE MAP.
      IF (I.LT.1.OR.I.GT.MX) GO TO 200
      J = YN(K)+1
      IF (J.LT.1.OR.J.GT.NY) GO TO 200
      IF (NUMBER(K).LE.0) GO TO 200
      NUMB = NUMBER(K)
      IVAL = NUM(I,J)
      IF (IVAL.GT.0) WRITE(6,1000) I,J,NUMB,IVAL
      IF (IVAL.EQ.-1) CALL DEFER(I,J,NUMBER(K))
      IF (IVAL.EQ.0) CALL ACTION(I,J,NUMBER(K))
  200 CONTINUE
  300 CONTINUE
      WRITE(99)NUM
      CALL CRUMBS
      CALL SWEEP
      RETURN
 1000 FORMAT( T20,'ERROR IN DIGITIZING.  POINT (',I4,',',I4,'),THE ',
     *  'BASIS OF POLYGON ',I4,  ' HAS ALREADY BEEN NAMED',I4)
      END
      SUBROUTINE PATCH1
      DIMENSION X(2),Y(2)
  100 READ(15,1000,END=900) X(1),Y(1),X(2),Y(2)
      CALL TRACK1(X,Y,2)
      GO TO 100
  900 CONTINUE
      RETURN
 1000 FORMAT( 4F10.2)
      END
      SUBROUTINE PATCH2(XN,YN,NUMBER,KREC,MREC)
      DIMENSION XN(100),YN(100),NUMBER(100)
      COMMON /TRASH/ INDEX,X1,Y1,NUMB
C                   ARE YOU ON THE RIGHT RECORD ?
  100 IF (MREC.NE.KREC) GO TO 500
C                   IF YES UPDATE VALUES.
      IF (X1.NE.0.0)   XN(INDEX) = X1
      IF (Y1.NE.0.0)   YN(INDEX) = Y1
      IF (NUMB.NE.0) NUMBER(INDEX) = NUMB
  500 CONTINUE
C                   READ ANOTHER CARD AND EXIT IF YOU READ BEYOND RECORD
C                   KREC.
      READ(16,1000,END=980) MREC,INDEX,X1,Y1,NUMB
C                   MAKE SURE THAT INDEX IS A MEANINGFUL NUMBER.
      IF (INDEX.LE.0.OR.INDEX.GT.100) GO TO 500
      IF (MREC.GT.KREC) RETURN
      GO TO 100
  980 CONTINUE
```

```
C                  IF END OF FILE, GIVE MREC A VALUE BIG ENOUGH SO
C                  THAT YOU WILL NOT REENTER PATCH2
      MREC = 999999
      RETURN
 1000 FORMAT( 2I10,2F10.2,I10)
      END
      SUBROUTINE PATCH3
      INTEGER*2 NUM
      INTEGER OLDNME
      COMMON /MATRIX/ NUM(500,500)
  100 CONTINUE
      READ(17,1000,END=980) I1,I2,J1,J2,NEWNME,OLDNME
      DO 300 J=J1,J2
      DO 200 I=I1,I2
      IF (NUM(I,J).EQ.OLDNME) NUM(I,J) = NEWNME
  200 CONTINUE
  300 CONTINUE
      GO TO 100
  980 CONTINUE
      RETURN
 1000 FORMAT( 6I10)
      END
      SUBROUTINE PPLOT2
C                  THIS SUBROUTINE CREATES A GERBER PLOT OF THE OUTPUT
C                  OF PROGRAM DEFPOLY.
      INTEGER*2 NUM
      INTEGER PEN,TPEN
      LOGICAL*1 BUG
      COMMON /DEBUG/ BUG
      COMMON /DIMEN/IXLOW,IYLOW,IXUP,IYUP
      COMMON /MATRIX/ NUM(500,500)
      DIMENSION XN(100),YN(100),NUMBER(100)
      CALL MARGIN
      L1=IXLOW+1
      L2=IXUP-1
      N1=IYLOW+1
      N2=IYUP-1
C                  DRAW ALL VERTICAL BOUNDARY LINES
      DO 200 I=L1,L2
      PEN=0
      DO 100 J=N1,IYUP
      TPEN=0
      IF(NUM(I,J).NE.NUM(I+1,J))TPEN=1
      IF(TPEN.EQ.PEN) GO TO 100
      IF (BUG) WRITE(6,7000) I,J,PEN,TPEN,NUM(I,J),NUM(I+1,J)
      K=J-1
      IF(PEN.EQ.0)CALL MOV(FLOAT(I),FLOAT(K))
      IF(PEN.EQ.1)CALL LINE2(FLOAT(I),FLOAT(K))
      PEN=TPEN
```

```
  100 CONTINUE
      IF(PEN.EQ.1) CALL LINE2(FLOAT(I),FLOAT(IYUP))
  200 CONTINUE
C                   DRAW ALL HORIZONTAL BOUNDARY LINES.
      DO 400 J=N1,N2
      PEN=0
      DO 300 I=L1,IXUP
      TPEN=0
      IF(NUM(I,J).NE.NUM(I,J+1))TPEN=1
      IF(PEN.EQ.TPEN)GO TO 300
      IF (BUG) WRITE(6,7000) I,J,PEN,TPEN,NUM(I,J),NUM(I,J+1)
      K=I-1
      IF(PEN.EQ.0)CALL MOV(FLOAT(K),FLOAT(J))
      IF(PEN.EQ.1)CALL LINE2(FLOAT(K),FLOAT(J))
      PEN=TPEN
  300 CONTINUE
      IF(PEN.EQ.1) CALL LINE2(FLOAT(IXUP),FLOAT(J))
  400 CONTINUE
      REWIND 60
  500 CONTINUE
C                   READ THE ID NUMBERS OF ALL LAND SEGMENTS.
      READ(60,ERR=990,END=999) JALL,XN,YN,NUMBER
      IF (JALL.LE.0) GO TO 980
C                   PUT THE IDHS ON THE MAP.
      DO 600 I=1,JALL
      IF (XN(I).LE.0.0.OR.YN(I).LE.0.0) GO TO 600
      IF (XN(I).GE.M.OR.YN(I).GE.N) GO TO 600
      WRITE(6,2000) XN(I),YN(I),NUMBER(I),I
      LX = XN(I)+1.0
      LY = YN(I)+1.0
      NAME = NUM(LX,LY)
      IF (NAME.NE.NUMBER(I)) WRITE(6,3000) NAME,NUMBER(I)
      IF(NAME.LT.10000)
     1CALL FXPLT(XN(I),YN(I),0.0,NAME)
  600 CONTINUE
      GO TO 500
  980 CONTINUE
      WRITE(6,1000)
      GO TO 999
  990 CONTINUE
      WRITE(6,1500)
  999 CONTINUE
      CALL DONE
      RETURN
 1000 FORMAT( 12X,'BAD JALL WHILE READING 60')
 1500 FORMAT( 12X,'ERROR WHILE READING ON DEVICE 60')
 2000 FORMAT( 12X,'THIS IS PPLOT',2(2X,F12.5),2I6)
 3000 FORMAT( T20,'NAME ',I5,'NUMBER(I)',I5,' DIFFER.  RECHECK YOUR ',
     *    'MAP')
```

```
 7000 FORMAT( T12,'PPLOT2,I,J,PEN,TPEN,NUM',10I5)
      END
      SUBROUTINE SEABRD(IX,IY,IN,IOUT)
      INTEGER*2 IX(2500),IY(2500),NUM
      COMMON /DIMEN/IXLOW,IYLOW,IXUP,IYUP
      COMMON /MATRIX/ NUM(500,500)
      NY=IYUP+10
      MX=IXUP+10
      DO 100 I=1,MX
      IF (NUM(I,1).GT.0) GO TO 50
      IY(IOUT) = 1
      IX(IOUT) = I
      IOUT = IOUT+1
   50 IF (NUM(I,NY).GT.0) GO TO 100
      IX(IOUT) = I
      IY(IOUT) = NY
      IOUT = IOUT+1
  100 CONTINUE
      DO 200 J=1,NY
      IF (NUM(1,J).GT.0) GO TO 150
      IX(IOUT) = 1
      IY(IOUT) = J
      IOUT = IOUT+1
  150 IF (NUM(MX,J).GT.0) GO TO 200
      IX(IOUT) = MX
      IY(IOUT) = J
      IOUT = IOUT+1
  200 CONTINUE
      RETURN
      END
      SUBROUTINE STUFIT
C                  WILL CREATE A DIRECT ACCESS DATA SET OF 256 RECORDS,
C                  EACH OF THEM CONTAINING A 30 BY 30 BLOCK OF MATRIX
C                  NUM.
      INTEGER*2 LSEG(30,30)
      INTEGER*2 NUM
      COMMON /DIMEN/IXLOW,IYLOW,IXUP,IYUP
      COMMON /MATRIX/ NUM(500,500)
      DEFINE FILE90(256,450,U,IP90)
      DO 300 IP=1,256
      LY = (IP-1)/16
      KX = IP-16*LY-1
      KX=30*KX+IXLOW
      LY=30*LY+IYLOW
      DO 200 I=1,30
      DO 100 J=1,30
      LSEG(I,J) = NUM(I+KX,J+LY)
      IF(LSEG(I,J).GE.10000) LSEG(I,J)=0
  100 CONTINUE
```

```
  200 CONTINUE
      WRITE(90'IP) LSEG
  300 CONTINUE
      RETURN
      END
      SUBROUTINE SWEEP
C                   ALLOCATES THE UNATTACHED BOUNDARY POINTS.
      INTEGER*2 NUM
      COMMON /DIMEN/IXLOW,IYLOW,IXUP,IYUP
      COMMON /MATRIX/NUM(500,500)
      M1=IXLOW+1
      N1=IYLOW+1
      LSWEEP = 0
      LNEW=250000
  100 CONTINUE
      LOLD=LNEW
      LNEW=0
      DO 400 J=N1,IYUP
      DO 300 I=N1,IXUP
      IF (NUM(I,J).GE.0) GO TO 300
      K = NUM(I-1,J)
      IF (K.GT.0) GO TO 200
      K = NUM(I,J-1)
      IF (K.GT.0) GO TO 200
      K = NUM(I+1,J)
      IF (K.GT.0) GO TO 200
      K = NUM(I,J+1)
      IF(K.GT.0)GO TO 200
      LNEW=LNEW+1
      GO TO 300
  200 NUM(I,J) = K
      LSWEEP = LSWEEP+1
  300 CONTINUE
  400 CONTINUE
      WRITE(6,6000) LSWEEP
      IF(LNEW.GT.0.AND.LNEW.LT.LOLD)GO TO 100
      RETURN
 6000 FORMAT( T12,'SWEEP FOUND '     ,I6,' MEANDERING BOUNDARY POINTS.')
      END
      SUBROUTINE TRACK1(XV,YV,NPOINT)
C     MARKS EACH CELL ON THE LINE BETWEEN TWO POINTS.
C                   THIS IS A VARIATION OF SLACK'S 'TRACK'.  INSTEAD OF
C                   TWO POINTS THE INPUT IS A VECTOR.
      INTEGER*2 NUM
      COMMON /MATRIX/NUM(500,500)
      DIMENSION XV(1),YV(1)
      DATA MX,NY/500,500/
      KPOINT = NPOINT-1
      DO 800 ILINE=1,KPOINT
```

```
      XL = XV(ILINE)
      YL = YV(ILINE)
      YI = YV(ILINE+1)
      XI = XV(ILINE+1)
      LX=XL
      LY=YL
      IX=XI
      IY=YI
C     TRACK THRU EACH CELL BETWEEN THE OLD POINT
C        AND THE NEW POINT.
      IF (IX.EQ.LX) GO TO 30
C     NON-VERTICAL PATH.
      SLOPE=(YI-YL)/(XI-XL)
      IXDO=IABS(IX-LX)+1
      IXADD=ISGN(IX-LX)
      XADD=IXADD
      IYADD=ISGN(IY-LY)
      KX=LX-IXADD
      XN=LX
      IF (XADD.LT.0.0) XN=XN+1.0
      DO 20 KK=1,IXDO
      XN=XN+XADD
      KX=KX+IXADD
      IF (KK.EQ.IXDO) XN=XI
      KY=(XN-XL)*SLOPE+YL
      IYDO=IABS(KY-LY)+1
      LY=LY-IYADD
      DO 10 LL=1,IYDO
      LY=LY+IYADD
C     RECORD THE TRACK THRU THIS CELL.
      IF (KX.LT.0.OR.KX.GE.MX.OR.LY.LT.0.OR.LY.GE.NY) GO TO 10
      NUM(KX+1,LY+1) = -1
   10 CONTINUE
      LY=KY
   20 CONTINUE
      GO TO 800
C     VERTICAL PATH.
   30 CONTINUE
      IF (IX.LT.0.OR.IX.GE.MX) GO TO 800
      IYADD=ISGN(IY-LY)
      IYDO=IABS(IY-LY)+1
      LY=LY-IYADD
      DO 40 LL=1,IYDO
      LY=LY+IYADD
C     RECORD THE TRACK THRU THIS CELL.
      IF (LY.LT.0.OR.LY.GE.NY) GO TO 40
      NUM(IX+1,LY+1) = -1
   40 CONTINUE
  800 CONTINUE
```

```
      RETURN
      END
      SUBROUTINE UNHOLE
C                  UNHOLE USES AN ALGORITHM SIMILAR TO FILIN3'S IN
C                  ORDER TO IDENTIFY ALL MAP HOLES.  THE HOLES, IF ANY,
C                  TAKE THEIR ID NUMBERS FROM ADJOINING POLYGONS.
      INTEGER*2 IX,IY,NUM
      COMMON /DIMEN/IXLOW,IYLOW,IXUP,IYUP
      COMMON /MATRIX/ NUM(500,500)
      COMMON /STACK/ IX(2500),IY(2500),IN,IOUT
      NMAX = 2500
      IN=1
      IOUT = 1
C                  CALL SEABRD TO INITIALIZE THE STACK WITH THE SEA
C                  BLOCKS LOCATED AT THE PERIPHERY OF THE MAP.
      CALL SEABRD(IX,IY,IN,IOUT)
      NSQ = IOUT-1
      IF (NSQ.LE.0) GO TO 225
      DO 220 K=1,NSQ
      NUM(IX(K),IY(K)) = 0
  220 CONTINUE
  225 CONTINUE
      NY=IYUP+10
      MMX=IXUP+9
      NNY=IYUP+9
      DO 250 J=2,NNY
      DO 240 I=2,MMX
      IF (NUM(I,J).GT.0) GO TO 240
      NUM(I,J)=-9
      NSQ = NSQ+1
  240 CONTINUE
  250 CONTINUE
      IF (NSQ.LE.0) RETURN
      WRITE(6,1200) NSQ,IOUT
  280 IF (IN.EQ.IOUT) GO TO 900
      K = IX(IN)
      L = IY(IN)
      NUM(K,L) = 0
      NSQ = NSQ-1
      IN = IN+1
      IF (IN.GT.NMAX) IN=1
      IF (K.EQ.1) GO TO 300
      IF (NUM(K-1,L).NE.-9) GO TO 300
      IX(IOUT) = K-1
      IY(IOUT) = L
      IOUT = IOUT+1
      NUM(K-1,L) = 0
      IF (IOUT.GT.NMAX) IOUT=1
      IF (IOUT.EQ.IN) GO TO 800
```

```
300 IF (L.EQ.1) GO TO 400
    IF (NUM(K,L-1).NE.-9) GO TO 400
    NUM(K,L-1) = 0
    IX(IOUT) = K
    IY(IOUT) = L-1
    IOUT = IOUT+1
    IF (IOUT.GT.NMAX) IOUT=1
    IF (IOUT.EQ.IN) GO TO 800
400 IF (K.EQ.MX) GO TO 500
    IF(NUM(K+1,L).NE.-9) GO TO 500
    NUM(K+1,L) = 0
    IX(IOUT) = K+1
    IY(IOUT) = L
    IOUT = IOUT+1
    IF (IOUT.GT.NMAX) IOUT=1
    IF (IOUT.EQ.IN) GO TO 800
500 IF (L.EQ.NY) GO TO 280
    IF (NUM(K,L+1).NE.-9) GO TO 280
    NUM(K,L+1) = 0
    IX(IOUT) = K
    IY(IOUT) = L+1
    IOUT = IOUT+1
    IF (IOUT.GT.NMAX) IOUT=1
    IF (IOUT.EQ.IN) GO TO 800
    GO TO 280
800 WRITE(6,1000) IX(IN),IY(IN),IN
    NUM(IX(IN),IY(IN)) = 0
    IN = IN+1
    IF (IN.GT.NMAX) IN = 1
    NSQ = NSQ-1
    GO TO 280
900 CONTINUE
    WRITE(6,1100) NSQ
    IF (NSQ.LE.0) RETURN
    L1=IXLOW+1
    M1=IYLOW+1
    DO 940 J=M1,IYUP
    DO 920 I=L1,IXUP
    IF(NUM(I,J).GE.0)GO TO 920
    NUM(I,J)=NUM(I+1,J)
    IF(NUM(I,J).GE.0)GO TO 920
    NUM(I,J)=NUM(I,J+1)
    IF(NUM(I,J).GE.0)GO TO 920
    NUM(I,J)=NUM(I-1,J)
    IF(NUM(I,J).GE.0)GO TO 920
    NUM(I,J)=NUM(I,J-1)
```

```
      IF(NUM(I,J).GE.0)GO TO 920
      NUM(I,J)=0
  920 CONTINUE
  940 CONTINUE
      RETURN
 1000 FORMAT( T12,'DANGER OF OVERFLOW.POINT (',I4,',',I4,') WAS DITCHED'
     *  ,' AND IN (',I4,') WILL BE AUGMENTED BY ONE.')
 1100 FORMAT( T12,'NSQ, THE # OF HOLES IS',I6)
 1200 FORMAT( T12,'UNHOLE: THE NUMBER OF PROSPECTIVE HOLES AND THE '
     * ,'NEXT FREE QUEUE ENTRY ARE ',2I5)
      END
```

Procedure: DIGIMAPS
Executes program: DATAREAD
Purpose: Read a digitizer tape and convert it to IBM codes.
Reads: Digitizer tape.
Writes files: SEDDS.DIG.MAP.&FILE
SEDDS.DIG.CTL.&FILE
SEDDS.DIG.PNAMES.&FILE (optional)
Text page reference: 7

Procedure listing:

```
//DIGIMAPS PROC  U1=2,UVOL=CCDXXX,
//*
//* REQUIRED: &STUDY,&FILE
//*
// TAPEIN=ANY,FILE=WHAT,STUDY=WHOKNOWS,PNAMES=PNAMES,V=2
//*CLEAR THE OLD FILES, IF ANY.
//C  EXEC  PGM=IEFBR14
//CLEAR1  DD  UNIT=3330,VOL=SER=&UVOL,DISP=(OLD,DELETE),
// DSN=SEDDS.&STUDY..DIG.MAP.&FILE
//CLEAR2  DD  UNIT=3330,VOL=SER=&UVOL,DISP=(OLD,DELETE),
// DSN=SEDDS.&STUDY..DIG.CTL.&FILE
//CLEAR3  DD  UNIT=3330,VOL=SER=&UVOL,DISP=(OLD,DELETE),
// DSN=SEDDS.&STUDY..DIG.&PNAMES..&FILE
//*READ THE DIGITIZER TAPE.
//A  EXEC  PGM=DATAREAD,TIME=&U1,REGION=120K
//STEPLIB  DD  UNIT=3330,VOL=SER=&UVOL,DISP=SHR,DSN=SEDDS.PGMLIB
//FT05F001  DD  DDNAME=SYSIN
//FT06F001  DD  SYSOUT=A
//*DIGITIZER TAPE.
//FT20F001    DD  UNIT=TAPELO,VOL=SER=&TAPEIN,
// DCB=(RECFM=F,BLKSIZE=512,DEN=2),DISP=(SHR,PASS),
// LABEL=(&V,BLP)
//FT25F001    DD  UNIT=3330,VOL=SER=&UVOL,
// DISP=(NEW,KEEP),DCB=(RECFM=VBS,LRECL=X,BLKSIZE=6444),
// SPACE=(TRK,(10,5),RLSE),DSN=SEDDS.&STUDY..DIG.MAP.&FILE
//FT30F001    DD  UNIT=3330,VOL=SER=&UVOL,
// DISP=(NEW,KEEP),DCB=(RECFM=VBS,LRECL=X,BLKSIZE=6444),
// SPACE=(TRK,(10,5),RLSE),DSN=SEDDS.&STUDY..DIG.CTL.&FILE
//FT45F001    DD  UNIT=3330,VOL=SER=&UVOL,
// DISP=(NEW,KEEP),DCB=(RECFM=VBS,LRECL=X,BLKSIZE=6444),
// SPACE=(TRK,(10,5),RLSE),DSN=SEDDS.&STUDY..DIG.&PNAMES..&FILE
//* IF THERE ARE NO POLYGONS,(PNAMES=NO), THEN GET RID
//* OF THE CORRESPONDING FILE.
//EPILOG  EXEC  PGM=IEFBR14
//E1  DD  UNIT=3330,VOL=SER=&UVOL,DISP=(OLD,DELETE),
// DSN=SEDDS.&STUDY..DIG.NO.&FILE
//
```

Program listing:

```
C                       PROGRAM  DATAREAD
C                   THIS PROGRAM WILL INTERPRET THE OUTPUT OF THE DIGI-
C                   TIZER (PDP TAPE) AND WILL CREATE TWO FILES.
C                   FILE 25 WILL CONTAIN THE MAP ITSELF WHILE FILE 30
C                   WILL CONTAIN CONTROL POINTS.
C         INPUT:
C           A.      THE TAPE FROM THE DIGITIZER
C           B.OPTIONAL  CARD INPUT ALTERING THE VALUES OF IBUG AND D.
C                   IF IBUG.NE.0 DETAILED OUTPUT WILL BE PRINTED.  D
C                   IS THE CONSTANT USED IN THE ENCODING OF THE
C                   CONTROL POINTS (DEFAULT VALUE=1000.0)
C         TOTAL     NUMBER OF HALFWORDS AVAILABLE FOR PROCESSING.
      LOGICAL*1 BLIP,LPUT(600),FLAG
      INTEGER*2 K2PUT(300)
      INTEGER TOTAL,SKIP,TYPE
      COMMON /CONTRL/XMIN,XMAX,YMIN,YMAX,PCON(100,4)
      COMMON /INDICS/ ITOP,TOTAL,JPOINT,MREST,SKIP,LEVEL,LEVMAX,NALL,
     * FLAG
      COMMON /ODBALL/ D,IOF,IBUG,ILINES,IRECIN,IRCOUT,BLIP
      COMMON /ONEMOR/ IALL,JALL,XN(100),YN(100),NUMBER(100)
      COMMON /POINTS/ KPUT(150)
      DIMENSION IPUT(128)
      EQUIVALENCE  (KPUT(23),IPUT(1),K2PUT(45),LPUT(89))
C                   CHECK IF THE TAPE IS O.K.
      CALL TPECHK(&50)
      STOP
   50 CONTINUE
C                   INITIALIZE THE PROGRAM VARIABLES
      FLAG = .TRUE.
      IALL = 1
      JALL = 1
      ILINES = 0
      IRECIN = 0
      IRCOUT = 0
      XMAX = 0.0
      YMAX = 0.0
      BLIP = .FALSE.
      IOF = 2**31
      ITOP = 45
      JPOINT = 1
      MREST = 0
      TOTAL = 0
      SKIP = 0
      LEVEL = 0
      LEVMAX = 1
      TYPE = 0
      NALL = 1
```

```
      IBUG = 0
      D = 1000.0
      READ(5,9000,END=100,ERR=100) IBUG,D
      IF (D.EQ.0.0) D=1000.0
  100 CONTINUE
      READ(20,1000,END=120) IPUT
      IRECIN = IRECIN+1
  120 CONTINUE
      TOTAL = 301-ITOP
C                  ARE THERE ANY POINTS TO BE RECORDED ?
  200 IF (MREST.GT.0) CALL RECORD(&100)
C                  IF NOT GO TO START TO FIND ANOTHER STRING.  IF NONE,
C                  FOLD YOUR OPERATION (&999).  OTHERWISE CONTINUE
C                  PROCESSING.
      CALL START(&100,&200,&999)
      GO TO 100
  999 CONTINUE
      NALL = NALL-1
C                  RECORD THE COLLECTED INFORMATION AND IF SO REQUESTED
C                   , PRINT IT AS WELL.
      WRITE(30) NALL,LEVMAX,XMIN,XMAX,YMIN,YMAX,PCON
      JALL = JALL-1
      IALL = IALL-1
      WRITE(45) IALL,XN,YN,NUMBER
      IF(IBUG.NE.0) WRITE(6,6000) JALL,IALL,(XN(I),YN(I),NUMBER(I),
     *  I=1,IALL)
      IF (IBUG.GT.0)
     *WRITE(6,5000) NALL,LEVMAX,XMIN,XMAX,YMIN,YMAX,
     *  (PCON(I,1),PCON(I,2),PCON(I,3),PCON(I,4),I=1,NALL)
      WRITE(6,1500) IBUG, D ,XMIN,XMAX,YMIN,YMAX,IRECIN,IRCOUT,ILINES
      STOP
 1000 FORMAT( 128A4)
 1500 FORMAT(12X,'DATAREAD JUST TERMINATED' /
     *    12X,'IBUG =',I3,', AND D =', F9.2 /
     *  12X,'XMIN,XMAX,YMI AND YMAX ARE RESPECTIVELY EQUAL TO',4F12.2 /
     * 12X,'WE READ',I5,' BLOCKS AND WE WROTE',I5,'. WE MET',I5,
     *  ' STRINGS.')
 5000 FORMAT( 2I6/8(1X,F14.4))
 6000 FORMAT( 12X,'IALL=',I6,' AND JALL=',I6 /
     *  12X,2(2F15.3,I10))
 9000 FORMAT( I10,F20.8)
      END
      SUBROUTINE CONINF
      LOGICAL*1 LPUT,LL(8)
      COMMON /CONTRL/XMIN,XMAX,YMIN,YMAX,PCON(100,4)
      COMMON /INDICS/ ITOP,TOTAL,JPOINT,MREST,SKIP,LEVEL,LEVMAX,NALL
      COMMON /ONEMOR/ IALL,JALL,XN(100),YN(100),NUMBER(100)
      COMMON /ODBALL/ D,IOF,IBUG,ILINES,IRECIN,IRCOUT
      COMMON /POINTS/ LPUT(600)
```

```
      DIMENSION A(4)
      DIMENSION II(2)
      EQUIVALENCE (LL(1) , II(1))
      M = 2*ITOP
      DO 200 K=1,2
      DO 100 I=1,4
      LL(2*I-1)= LPUT(M)
      LL(2*I) = LPUT(M-1)
      M = M+2
  100 CONTINUE
      II(1) = II(1)-IOF
      II(2) = II(2)-IOF
      A(2*K) = II(2)
      A(2*K-1) = II(1)
  200 CONTINUE
      IF (LEVEL.GT.1) GO TO 400
      IF (XMIN.NE.A(1)) GO TO 300
      IF (YMIN.NE.A(2)) GO TO 300
      XMAX = A(3)
      YMAX = A(4)
      GO TO 900
  300 CONTINUE
      IF (NALL.GT.100) GO TO 900
      PCON(NALL,1) = A(1)
      PCON(NALL,2) = A(2)
      PCON(NALL,3) = (A(3)-A(1))/D
      PCON(NALL,4) = (A(4)-A(2))/D
      IF (IBUG.NE.0)
     *WRITE(6,7500) NALL,LEVEL,PCON(NALL,1),PCON(NALL,2),
     *  PCON(NALL,3),PCON(NALL,4)
      NALL = NALL+1
      GO TO 900
  400 CONTINUE
      IF (LEVEL.GT.2) GO TO 800
      IF (A(1).NE.A(3)) GO TO 900
      NUMBER(IALL) = (A(4)-A(2))/D
      XN(IALL) = A(1)
      YN(IALL) = A(2)
      IF (IALL.EQ.100) WRITE(45) IALL,XN,YN,NUMBER
      IF (IBUG.NE.0) WRITE(6,6000) IALL,JALL,XN(IALL),YN(IALL),
     *  NUMBER(IALL)
      IF (IALL.EQ.100) IALL=0
      IALL = IALL+1
      JALL = JALL+1
  800 CONTINUE
  900 CONTINUE
      RETURN
 6000 FORMAT( 12X,2I6 / 12X,2(2F15.3,I10))
 7500 FORMAT(12X,2I6,4F15.3)
```

```
      END
      SUBROUTINE COPYUP
      INTEGER TOTAL,SKIP,TYPE
      LOGICAL*1 LPUT(600)
      LOGICAL*1 FLAG
      INTEGER*2 I2PUT(300)
      EQUIVALENCE  (KPUT(1) , I2PUT(1))
      EQUIVALENCE(KPUT(1) , LPUT(1))
      COMMON /POINTS/ KPUT(150)
      COMMON /INDICS/ ITOP,TOTAL,JPOINT,MREST,SKIP,LEVEL,LEVMAX,NALL,
     * FLAG
C                  MOVE THE UNUSED INFO FROM THE BOTTOM OF INPUT TO THE
C                  TOP.  SET ITOP=POSITION OF FIRST HALF WORD
C                  CONTAINING INFO.
      KTOP = 44-TOTAL
      IF (TOTAL.EQ.0) GO TO 400
      ITOP = ITOP-1
      DO 300 I=1,TOTAL
      I2PUT(KTOP+I) = I2PUT(ITOP+I)
  300 CONTINUE
  400 CONTINUE
      ITOP = KTOP+1
      RETURN
      END
      SUBROUTINE DISKBY
      LOGICAL*1 FLAG
      COMMON /COORD/ X(31),Y(31)
      COMMON /INDICS/ ITOP,TOTAL,JPOINT,MREST,SKIP,LEVEL,LEVMAX,NALL,
     * FLAG
      COMMON /ODBALL/ D,IOF,IBUG,ILINES,IRECIN,IRCOUT,BLIP
C                  RECORD THE INFO.  IF THIS IS THE FIRST RECORD OF A
C                  STRING, FLAG = TRUE.
      NPT = JPOINT-1
      MPT = NPT
      IF (FLAG) NPT = -NPT
      IF (FLAG) ILINES=ILINES+1
      IRCOUT = IRCOUT+1
      WRITE(25) NPT,LEVEL,X,Y
      IF (IBUG.NE.0)
     *WRITE(6,2000) NPT,LEVEL,(X(N),Y(N),N=1,MPT)
      JPOINT = 1
      FLAG = (MREST.EQ.0)
      RETURN
 2000 FORMAT(12X,2I5 / 8(F14.2,1X))
      END
      SUBROUTINE OFFSET
C                  FINDS THE ORIGIN OF THE MAP
      LOGICAL*1 BLIP
      LOGICAL*1 LL(8)
```

```
      LOGICAL*1 LPUT(600)
      COMMON /INDICS/ ITOP
      EQUIVALENCE   (IOR(1) , LL(1))
      COMMON /CONTRL/XMIN,XMAX,YMIN,YMAX,PCON(100,4)
      COMMON /ODBALL/ D,IOF,IBUG,ILINES,IRECIN,IRCOUT,BLIP
      DIMENSION IOR(2)
      COMMON /POINTS/ LPUT(600)
      M = 2*ITOP+44
      LL(1) = LPUT(M)
      LL(2) = LPUT(M-1)
      LL(3) = LPUT(M+2)
      LL(4) = LPUT(M+1)
      LL(5) = LPUT(M+4)
      LL(6) = LPUT(M+3)
      LL(7) = LPUT(M+6)
      LL(8) = LPUT(M+5)
      XMIN = IOR(1)-IOF
      YMIN = IOR(2)-IOF
      IF (IBUG.NE.0) WRITE(6,8000) IOF,XMIN,YMIN
      RETURN
 8000 FORMAT( 12X,'OFFSET=',I10,' AND ORIGIN IS',2F15.2)
      END
      SUBROUTINE RECORD(*)
      LOGICAL*1 LPUT(600)
      INTEGER TOTAL,SKIP,TYPE
      LOGICAL*1 FLAG
      LOGICAL*1 LL(8)
      INTEGER II(2)
      EQUIVALENCE  (II(1),LL(1))
      EQUIVALENCE(KPUT(1) , LPUT(1))
      COMMON /COORD/ X(31),Y(31)
      COMMON /INDICS/ ITOP,TOTAL,JPOINT,MREST,SKIP,LEVEL,LEVMAX,NALL,
     * FLAG
      COMMON /ODBALL/ D,IOF,IBUG,ILINES,IRECIN,IRCOUT,BLIP
      COMMON /POINTS/ KPUT(150)
C                  THIS SUBROUTINE DECODES THE INPUT TAPE AND OBTAINS
C                  THE COORDINATES OF THE MAP POINTS
C                  IF NOT ENOUGH INFO, GO AND READ ONE MORE RECORD
  200 CONTINUE
      IF (TOTAL.LT.4) GO TO 900
C                  ELSE RECORD ANOTHER POINT AND DECREMENT TOTAL,MREST,
C                  ITOP AND JPOINT ACCORDINGLY.
      M = 2*ITOP
      DO 300 I=1,4
      LL(2*I-1) = LPUT(M)
      LL(2*I) = LPUT(M-1)
      M = M+2
  300 CONTINUE
      ITOP = ITOP+4
```

```
      TOTAL = TOTAL-4
      X(JPOINT) = II(1)-IOF
      Y(JPOINT) = II(2)-IOF
      MREST = MREST-1
      JPOINT = JPOINT+1
C                  IF THE STRING IS DONE (MREST=0) OR IF WE HAVE A FULL
C                  DISK RECORD, RECORD THE INFO.
      IF (MREST.EQ.0.OR.JPOINT.EQ.32) CALL DISKBY
C                  IF STRING IS DONE RETURN. OTHERWISE CONTINUE PROCE-
C                  SSING
      IF (MREST.EQ.0) GO TO 999
      GO TO 200
  900 CALL COPYUP
      RETURN1
  999 RETURN
      END
      SUBROUTINE START(*,*,*)
C                  READS THE BEGINING OF EACH STRING AND PROCESSES THE
C                  STRING ACCORDINGLY.
      LOGICAL*1 BLIP,FLAG,LL(8),LS(4),LPUT(600)
      INTEGER*2 J2,K2PUT(300)
      INTEGER II(2)
      EQUIVALENCE  (J2 , LL(1))
      EQUIVALENCE (II(1) , LL(1))
      EQUIVALENCE  (SKIP,LS(1))
      INTEGER TOTAL,SKIP,TYPE
      EQUIVALENCE (KPUT(1) , K2PUT(1))
      COMMON /CONTRL/XMIN,XMAX,YMIN,YMAX,PCON(100,4)
      COMMON /INDICS/ ITOP,TOTAL,JPOINT,MREST,SKIP,LEVEL,LEVMAX,NALL,
     * FLAG
      COMMON /ODBALL/ D,IOF,IBUG,ILINES,IRECIN,IRCOUT,BLIP
      COMMON /POINTS/ KPUT(150)
      EQUIVALENCE(KPUT(1) , LPUT(1))
      NFIVE = 5
      IF (NFIVE.EQ.0) GO TO 999
C                  IF PART OF THE INPUT IS TO BE SKIPPED, CALL SKIPIN
  100 IF(SKIP.GT.0) CALL SKIPIN
C                  IF NOT ENOUGH INFO CALL COPYUP AND GO TO RECORD
C                  SOME MORE.
      IF (TOTAL.LT.40) GO TO 900
C                  ELSE FIND LEVEL,TYPE AND # OF WORDS OF PRESENT
C                  STRING.IF J2 IS NEGATIVE, EITHER WE MET THE END OF
C                  THE TAPE (J2=-1) OR A NON ACTIVE STRING TO BE
C                  SKIPPED.
      LL(1) = LPUT(2*ITOP)
      LL(2) = LPUT(2*ITOP-1)
      LS(3) = LPUT(2*ITOP+2)
      LS(4) = LPUT(2*ITOP+1)
      IF (J2.LT.0) GO TO 800
```

```
      LEVEL = J2-(J2/64)*64
      TYPE = J2/256
      IF(TYPE.LT.3) GO TO 990
      IF (TYPE.GT.4) GO TO 990
      IF (LEVEL.GT.LEVMAX) LEVMAX = LEVEL
      IF (TYPE.EQ.3) GO TO 600
      IF (TYPE.EQ.4) CALL STRING
      GO TO 680
  600 CONTINUE
      ITOP = ITOP+19
      TOTAL = TOTAL-19
      IF (LEVEL.LE.NFIVE) GO TO 700
      MREST = 2
      SKIP = SKIP-17-4*MREST
  680 RETURN2
  700 CONTINUE
      CALL CONINF
      ITOP = ITOP+8
      TOTAL = TOTAL-8
      SKIP = SKIP-25
      RETURN2
  800 CONTINUE
      IF (J2.EQ.-1) RETURN3
      LL(1) = LPUT(2*ITOP+2)
      LL(2) = LPUT(2*ITOP+1)
      SKIP = J2+2
      GO TO 100
  900 CALL COPYUP
      RETURN1
  990 CONTINUE
      IF (TYPE.EQ.8) CALL OFFSET
      IF (BLIP) WRITE(6,5000) TYPE
      BLIP = (TYPE.EQ.10)
      GO TO 800
  999 CONTINUE
      RETURN
 5000 FORMAT( 15X,'ERROR OCCURED.  TYPE=' I5)
      END
      SUBROUTINE SKIPIN
C                 WILL DISCARD THE FIRST SKIP HALFWORDS.
      LOGICAL*1 FLAG
      INTEGER TOTAL,SKIP,TYPE
      DIMENSION IPUT(128)
      COMMON /INDICS/ ITOP,TOTAL,JPOINT,MREST,SKIP,LEVEL,LEVMAX,NALL,
     * FLAG
      COMMON /POINTS/ KPUT(150)
      EQUIVALENCE  (KPUT(23) , IPUT(1))
  100 CONTINUE
      IF  (SKIP.LE.TOTAL) GO TO 500
```

```
      SKIP = SKIP-TOTAL
      READ(20,1000,END=950) IPUT
      ITOP = 45
      TOTAL = 256
      GO TO 100
  500 CONTINUE
      ITOP = ITOP+SKIP
      TOTAL = TOTAL-SKIP
      SKIP = 0
      GO TO 999
  950 CONTINUE
      WRITE(6,5000)
      STOP
  999 RETURN
 1000 FORMAT( 128A4)
 5000 FORMAT(15X,'END OF FILE 20 WHILE IN SKIPIN')
      END
      SUBROUTINE STRING
C                 SETS UP THE PARAMETERS IN ORDER TO RECORD A STRING.
C                 (TYPE=4).
      LOGICAL*1 FLAG
      INTEGER TOTAL,SKIP,TYPE
      INTEGER*2 K2PUT(300)
      LOGICAL*1 LL(2),LPUT(600)
      INTEGER*2 J2
      EQUIVALENCE  (J2,LL(1))
      EQUIVALENCE  (KPUT(1) , LPUT(1))
      COMMON /POINTS/ KPUT(150)
      COMMON /INDICS/ ITOP,TOTAL,JPOINT,MREST,SKIP,LEVEL,LEVMAX,NALL,
     * FLAG
      EQUIVALENCE (KPUT(1) , K2PUT(1))
      ITOP = ITOP+17
      LL(1) = LPUT(2*ITOP)
      LL(2) = LPUT(2*ITOP-1)
      MREST = J2
      ITOP = ITOP+1
      TOTAL = TOTAL-18
      SKIP = SKIP-16-4*MREST
      RETURN
      END
      SUBROUTINE TPECHK(*)
      INTEGER*2 I
      LOGICAL*1 A(512)
      READ(20,2000,END=999) I
      REWIND 20
      J = I/256
      I = I-256*J
      IF (I.EQ.9) RETURN1
      WRITE(6,6100) I
```

```
      DO 100 J=1,25
      READ(20,1000,END=800) A
      WRITE(6,6200) J
      WRITE(6,6000) (A(2*K),A(2*K-1),K=1,256)
  100 CONTINUE
  800 CONTINUE
  999 RETURN
 1000 FORMAT( 128A1,128A1,128A1,128A1)
 2000 FORMAT( A2)
 6000 FORMAT( 12(2X,4Z2))
 6100 FORMAT( T12,'THE TAPE IS BAD (FIRST RECORD HAS TYPE ',I4,').',
     *  /  T12,'THE FIRST 25 RECORDS WILL BE PRINTED.')
 6200 FORMAT( / T12,'THIS IS RECORD # ',I4, /)
      END
```

Procedure: PLOT
Executes program: PLOT
Purpose: Plot a digitized map.
Reads file: SEDDS.&STUDY.%&FILE
Writes: Gerber 4477 plot tape.
Text page reference:

Procedure listing:

```
//PLOT PROC U=CCDXXX,FM=1
//G EXEC PGM=PLOT,REGION=120K,TIME=3
//STEPLIB DD UNIT=3330,VOL=SER=&U,DISP=SHR,DSN=SEDDS.PGMLIB
//FT05F001 DD DDNAME=SYSIN
//FT06F001 DD SYSOUT=A
//FT10F001 DD UNIT=3330,VOL=SER=&U,DISP=SHR,
// DSN=SEDDS.&STUDY..&FILEIN
//FT13F001 DD UNIT=TAPE10,VOL=SER=&PTAPE,
// LABEL=(&FM,SL),DISP=(NEW,PASS),
// DCB=(RECFM=FB,LRECL=72,BLKSIZE=1008,DEN=2),
// DSN=PLOTTAPE
//FT90F001 DD UNIT=3330,VOL=SER=&U,DISP=SHR,
// DSN=SEDDS.&STUDY..BASEMAP
```

Program listing:

```
C     PROGRAM PLOT

C     PLOTS AN OBJECT AND A BASEMAP IF INDICATED
C     INPUT
C       CARD NUMBER 1:
C         COL. 01-08 STUDY (2A4)
C         COL. 09 BLANK
C         COL. 10-14 X-SCALE (F5.2)
C         COL. 15 BLANK
C         COL. 16-20 Y-SCALE (F5.2)
C       CARD NUMBER 2:
C         COL. 01-10 X-ORIGIN (F10.2)
C         COL. 11-20 Y-ORIGIN (F10.2)
C         COL. 21 BLANK
C         COL. 22 BASEMAP (I1) 0=NO, 1=YES
C         COL. 23 BLANK
C         COL. 24 LAT.-LONG. (I1) 0=NO, 1=YES
C         COL. 25 BLANK
C         COL. 26-27 OBJECT FILE (I2)
C         COL. 28 BLANK
C         COL. 29-32 GRID (I4)
C         COL. 33 BLANK
C         COL. 34-41 FILE NAME (2A4)
C
```

```
      REAL STUDY(2),FILE(2)
      INTEGER CARD/5/,PRINT/6/,UBASE/90/
      INTEGER UPLOT,BMP
      READ(CARD,5000)STUDY,XS,YS
 5000 FORMAT(2A4,2(1X,F5.2))
      CALL INIT(1)
      CALL SCAL(XS,YS)
  100 READ(CARD,5001,END=999)XORIG,YORIG,BMP,ILL,UPLOT,IGR,FILE
 5001 FORMAT(2(F10.2),2(1X,I1),1X,I2,1X,I4,1X,2A4)
      WRITE(PRINT,6000)XORIG,YORIG,BMP,ILL,UPLOT,IGR,FILE
 6000 FORMAT(' XORIG= ',F10.2,' YORIG= ',F10.2,' BMP= ',I1,' ILL= ',I1,
     1  ' UPLOT= ',I2,' IGR= ',I4,' FILE= ',2A4)
      IF(BMP.EQ.1)CALL PLOTMP(STUDY,XORIG,YORIG,IGR,ILL,UBASE)
      CALL PLOTMP(STUDY,XORIG,YORIG,0,0,UPLOT)
      CALL ALPHA2(XORIG,(YORIG-19.0),5.0,0.0,8,FILE)
      WRITE(PRINT,6001)FILE
 6001 FORMAT(' FINISHED FILE ',2A4)
      GO TO 100
C     RELEASE THE PLOTTER
  999 CONTINUE
      CALL DONE
      STOP
      END
```

Procedure: TGTMATRX
Executes program: TGTMATRX
Purpose: Form a matrix in compact storage notation.
Reads file: SEDDS.ME.%FILEa
SEDDS.ME.&FILEb concatenated
Other concatenated files.
Writes file: SEDDS.&STUDY.DA.TGTMATRX
Page reference: 15

Procedure listing:

```
//TGTMATRX PROC D=CCD836,TM=3,RG=500K,
//C EXEC PGM=IEFBR14
//CLEAR1 DD UNIT=3330,VOL=SER=&D,DISP=(OLD,DELETE),
// DSN=SEDDS.&STUDY..TARGETS
//G EXEC PGM=TGTMATRX,TIME=&TM,REGION=&RG
//STEPLIB DD UNIT=3330,VOL=SER=&D,DISP=SHR,DSN=SEDDS.PGMLIB
//FT05F001 DD DDNAME=SYSIN
//FT06F001 DD SYSOUT=A
//FT20F001 DD UNIT=3330,VOL=SER=&D,DISP=SHR,
// DSN=SEDDS.&STUDY..GENERAL
//FT40F001 DD UNIT=3330,VOL=SER=&D,DISP=(NEW,DELETE),
// DCB=(RECFM=VBS,LRECL=X,BLKSIZE=6444),
// SPACE=(TRK,(10,5),RLSE,DSN=SHADOW
//FT50F001 DD UNIT=3330,VOL=SER=&D,DISP=(NEW,KEEP),
// DCB=DSORG=DA,SPACE=(3600,(256)),
// DSN=SEDDS.&STUDY..TARGETS
```

Program listing:

```
C                       PROGRAM MAPOBJ.TWO
C
C       A.CARD READER :   THE FIRST CARD HAS ONE I4 FIELD, LEVMIN.
C                 ONLY POINTS AT LEVEL LEVMIN AND ABOVE WILL BE
C                 BOUNDARY POINTS.
C                 THE SECOND CARD HAS A SINGLE I4 FIELD, NOBJ,THE NUMBER
C                 OF OBJECTS THAT WE ARE GOING TO PROCESS
C                 EACH SUBSEQUENT CARD HAS THREE I4 FIELDS, KOBJ,
C                 THE OBJECT'S ID #,NFILES, THE # OF FILES OVER WHICH
C                 THE OBJECT IS DISTRIBUTED, AND KNOW, WHICH IF NOT
C                 ZERO, ASKS FOR SOME EXTRA PRINTOUT FOR THE GIVEN OBJECT.
C       B.FILE 20     :   THE STUDY NAME,THE GRID-TRANSFORMATION COEFFI-
C                 CIENTS AND THE GRID DIMENSIONS.
C       C.FILE 30     :   THE BOUNDARY POINTS (LEVEL NO LESS THAN
C                 LEVMIN) AND THE SHADOW POINTS (THE REST)  FILE30 IS
C                 A CONCATENATED FILE.
C                               OUTPUT
C       FILE 50 : A DIRECT ACCESS FILE OF 256 RECORDS, EACH CORRESPOND-
C                 ING TO A 30 BY 30 SUBBLOCK OF THE GRID.
```

```
C                   ( SEE ALSO SUBROUTINE NOUT).
      LOGICAL*1 A,JPT
      LOGICAL*1 BUG
      COMMON /DEBUG/ BUG
      COMMON /DIMEN/ IX,IY
      COMMON /DIVERS/ KOBJ,MOBJ,NOBJ,MFILE,NFILES,JALL
      COMMON /MAP/ A(480,480)
      COMMON /PARMS/ A1,B1,C1,A2,B2,C2,IX,IY,LEVMIN
      COMMON /POINTS/ JPT(16,16)
      DIMENSION LMAP(30,30),STUDY(2)
      DATA LMAP /900*0/
      DEFINE FILE50(256,900,U,IP50)
      READ(20) STUDY,IX,IY,A1,B1,C1,A2,B2,C2
      WRITE(6,6025) STUDY,A1,B1,C1,A2,B2,C2,IX,IY
      READ(5,1000) LEVMIN
      WRITE(6,6050) LEVMIN
      DO 10 LREC=1,256
      WRITE(50'LREC) LMAP
   10 CONTINUE
      MBYTE = 480*480
      READ(5,1000,END=999,ERR=999) NOBJ
      WRITE(6,6075) NOBJ
      IF (NOBJ.LE.0) GO TO 990
      DO 900 MOBJ=1,NOBJ
  100 READ(5,1000) KOBJ,NFILES,KNOW
      WRITE(6,6080) KOBJ,NFILES,KNOW
      IF (NFILES.LE.0) GO TO 850
      IF (KOBJ.LT.0.OR.KOBJ.GT.31) GO TO 860
C                   (RE)INITIALIZE ALL OBJECT DEPENDENT PARAMETERS.
      JALL = 0
      BUG = (KNOW.NE.0)
      CALL ZERO1(A,MBYTE)
      CALL ZERO1(JPT,256)
      WRITE(6,6500) KOBJ
      REWIND 40
      DO 800 MFILE=1,NFILES
C                   FOR EACH FILE, CALL MAPOBJ
      WRITE(6,6000) MFILE
      CALL MAPOBJ
  800 CONTINUE
C                   JALL IS THE NUMBER OF SHADOW POINTS.
      WRITE(6,3128) JALL
C                   FILLER FILLS THE OBJECT'S INTERIOR.
      CALL FILER2
C                   NOUT OUTPUTS THE RESULTS OF FILLER IN FILE 50.
      CALL NOUT
C                   IF KNOW > 9 , PRINT OUT THE RESULTS OF FILER2.
      IF (KNOW.GT.9) CALL FLOUTY
C                   GO GET THE NEXT OBJECT.
```

```
      GO TO 900
  850 WRITE(6,6200) KOBJ
      GO TO 900
  860 WRITE(6,6300) KOBJ
  900 CONTINUE
      GO TO 999
  990 WRITE(6,6100)
  999 CONTINUE
      STOP
 1000 FORMAT( 20I4)
 3128 FORMAT( T12,'JALL,THE NUMBER OF SHADE POINTS, IS',I4)
 6000 FORMAT( 12X,'WE ENTERED FILE #',I3)
 6025 FORMAT( T12,2A4,6(2X,D12.4),2I4)
 6050 FORMAT( T12,' ONLY LEVELS ABOVE',I4,' WILL BE BOUNDARIES.')
 6075 FORMAT( T12,'NOBJ=',I3)
 6080 FORMAT( T12,'KOBJ,NFILES, AND KNOW ARE',3I5,'.  IF KNOW IS NOT',
     *  ' ZERO, WE WILL HAVE EXTENSIVE OUTPUT.')
 6200 FORMAT( 10X,'NO FILES (NFILES<1) FOR OBJECT #',I3)
 6100 FORMAT( 10X,'NO INPUT OBJECTS!!  NOBJ<1')
 6300 FORMAT( 12X,'THE VALUE OF KOBJ (',I4,') IS UNACCEPTABLE.  MOVE ',
     *  'ONTO THE NEXT OBJECT')
 6500 FORMAT( 12X,'WE START PROCESSING OBJECT #',I3)
      END
      SUBROUTINE FILER2
C                  FILLER READS THE SHADOW POINTS AND,IF THEY DO NOT
C                  BELONG TO THE BOUNDARY, CALLS FILLIN
      LOGICAL*1 A
      COMMON /DIVERS/ KOBJ,MOBJ,NOBJ,MFILE,NFILES,JALL
      COMMON /MAP/ A(480,480)
      COMMON /PARMS/ A1,B1,C1,A2,B2,C2,IX,IY
      DIMENSION X(31),Y(31)
      REWIND 40
  100 IF (JALL.LE.0) GO TO 999
      READ(40,ERR=980,END=980) M,LEVEL,X,Y
      JALL = JALL-M
      DO 200 K=1,M
      I = X(K)+1
      IF (I.LT.1.OR.I.GT.IX) GO TO 200
      J = Y(K)+1
      IF (J.LT.1.OR.J.GT.IY) GO TO 200
      IF (A(I,J)) GO TO 200
      CALL FILIN2(I,J)
  200 CONTINUE
      GO TO 100
  980 CONTINUE
  999 CONTINUE
      JALL = 0
      RETURN
      END
```

```
      SUBROUTINE FILIN2(I,J)
C                  FILIN2 WORKS ON A 'CIRCULAR' QUEUE (IX,JY). POINTER
C                  IN POINTS AT THE QUEUE'S FIRST ELEMENT AND POINTER
C                  IOUT POINTS AT THE FIRST FREE ELEMENT.  AT EVERY STEP:
C               A)  WE REMOVE THE FIRST ELEMENT.
C               BE  WE ADD TO THE QUEUE ALL ITS NEIGHBORS WHICH HAVE
C                   VALUE .FALSE. (ARE NOT BOUNDARY POINTS) AND WE SET
C                   THEIR VALUE .TRUE. .
C                  THE ROUTINE IS EXITED WHEN IN=IOUT.
      LOGICAL*1 A,JPT
      INTEGER*2 IX(2500),JY(2500)
      COMMON /MAP/ A(480,480)
      COMMON /PARMS/ R(6),MX,NY
      COMMON /POINTS/ JPT(16,16)
      NMAX = 2500
C                  INITIALIZE THE QUEUE
      IX(1) = I
      JY(1) = J
      A(I,J) = .TRUE.
      IN = 1
      IOUT = 2
C                  START PROCESSING.
  100 IF (IN.EQ.IOUT) GO TO 999
      K = IX(IN)
      L = JY(IN)
      JPT((K-1)/30+1,(L-1)/30+1) = .TRUE.
      IF (IN.EQ.NMAX) WRITE(6,6100) IOUT
      IN = IN+1
      IF (IN.GT.NMAX) IN = 1
      IF (K.EQ.1) GO TO 200
      IF (A(K-1,L)) GO TO 200
      A(K-1,L) = .TRUE.
      IX(IOUT) = K-1
      JY(IOUT) = L
      IOUT = IOUT+1
      IF (IOUT.GT.NMAX) IOUT=1
      IF (IOUT.EQ.IN) GO TO 800
  200 IF (L.EQ.1) GO TO 300
      IF (A(K,L-1)) GO TO 300
      A(K,L-1) = .TRUE.
      IX(IOUT) = K
      JY(IOUT) = L-1
      IOUT = IOUT+1
      IF (IOUT.GT.NMAX) IOUT=1
      IF (IOUT.EQ.IN) GO TO 800
  300 IF (K.EQ.MX) GO TO 400
      IF (A(K+1,L)) GO TO 400
      A(K+1,L) = .TRUE.
      IX(IOUT) = K+1
```

```
      JY(IOUT) = L
      IOUT = IOUT+1
      IF (IOUT.GT.NMAX) IOUT = 1
      IF (IOUT.EQ.IN) GO TO 800
  400 IF (L.GE.NY) GO TO 100
      IF (A(K,L+1)) GO TO 100
      A(K,L+1) = .TRUE.
      IX(IOUT) = K
      JY(IOUT) = L+1
      IOUT = IOUT+1
      IF (IOUT.GT.NMAX) IOUT = 1
      IF (IOUT.NE.IN) GO TO 100
C                   ENTERED IN THE UNLIKELY EVENT THAT OVERFLOW MENACES.
  800 WRITE(6,6000) IN,IOUT,IX(IN),JY(IN)
      IN = IN+1
      IF (IN.GT.NMAX) IN = 1
      GO TO 100
  999 CONTINUE
      RETURN
 6000 FORMAT( T12,'DANGER OF OVERFLOWING. IN AND IOUT ARE',2I5 ,
     *  '. AND THE POINT(',I4,',',I4,') WAS DITCHED')
 6100 FORMAT( T12,'IN EQUALS NMAX AND IOUT IS',I5)
      END
      SUBROUTINE FLOUTY
C                   THIS ROUTINE WILL PRINT THE 30 BY 30 BLOCKS THAT
C                   OVERLAP WITH THE CURRENTLY PROCESSED OBJECT.
      LOGICAL*1 A,JPT
      COMMON /POINTS/ JPT(16,16)
      COMMON /MAP/ A(480,480)
      WRITE(6,8128)
      WRITE(6,6005) ((JPT(I1,17-I2),I1=1,16),I2=1,16)
      DO 500 I=1,16
      DO 400 J=1,16
      IF (.NOT.JPT(I,J)) GO TO 400
      K1 = 30*(I-1)+1
      K2 = 30*I
      L2 = 30*J+1
      WRITE(6,6500) I,J,((A(K,L2-L),K=K1,K2),L=1,30)
  400 CONTINUE
  500 CONTINUE
      RETURN
 6005 FORMAT(T12,'THIS IS PJT'/ (12X,16L1))
 6500 FORMAT(T12,2I10          / (12X,30L1))
 8128 FORMAT( T12,'THIS IS A BLOCK VIEW OF THE MAP. THOSE BLOCKS WHICH
     *OVERLAP WITH THE OBJECT WILL BE TEES.'  /  T12,
     *  'EACH BLOCK IS 30 BY 30 GRID CELLS.')
      END
      SUBROUTINE MAPOBJ
C                   TRANSFORMS THE INPUT POINTS TO GRID COORDINATES AND
```

```
C                   MARKS THE BOUNDARY OF THE OBJECT.
      LOGICAL*1 JPT
      LOGICAL*1 BUG
      COMMON /DEBUG/ BUG
      COMMON /DIVERS/ KOBJ,MOBJ,NOBJ,MFILE,NFILES,JALL
      COMMON /PARMS/ A1,B1,C1,A2,B2,C2,IX,IY,LEVMIN
      COMMON /POINTS/ JPT(16,16)
      DIMENSION X(31),Y(31)
C                   THE FIRST RECORD CONTAINS NO POINTS.
      READ(30)
   50 CONTINUE
      READ(30) N,LEVEL,X,Y
C                   IF N=0, THIS IS THE LAST RECORD FOR THE CURRENT FILE.
      IF (N.EQ.0) GO TO 999
      M = IABS(N)
C                   GET THE GRID COORDINATES.
      CALL CXYXY3(X,Y,X,Y,A1,B1,C1,A2,B2,C2,M)
      IF (BUG) WRITE(6,6123) N,LEVEL,(X(I),Y(I),I=1,M)
      IF (LEVEL.GE.LEVMIN) GO TO 80
C                   IF SHADOW (LEVEL<LEVMIN) RECORD THE NUMBER OF POINTS
C                   AND GO GET THE NEXT RECORD.
      JALL = JALL+M
      WRITE(40) M,LEVEL,X,Y
      GO TO 50
   80 CONTINUE
      J = 1
      IF (N.GT.0) GO TO 100
      XO = X(1)
      YO = Y(1)
      IF (M.EQ.1) GO TO 50
      J = 2
  100 CONTINUE
C                   ELSE GO AND MARK ALL BOUNDARY CELLS.
      DO 200 I=J,M
      CALL NTRACK(XO,YO,X(I),Y(I))
      XO = X(I)
      YO = Y(I)
  200 CONTINUE
      GO TO 50
  999 RETURN
 6123 FORMAT( T12,'FROM MAPOBJ YOU GET N,LEVEL,AND GRID COORDINATES',
     *  2I5, /, 12(2X,F8.2))
      END
      SUBROUTINE NOUT
C                   COMBINES THE MAP OF THE OBJECT WITH WHAT HAS ALREADY
C                   BEEN RECORDED IN LMAP.
      LOGICAL*1 A,JPT
      COMMON /DIVERS/ KOBJ,MOBJ,NOBJ,MFILE,NFILES,JALL
      COMMON /MAP/ A(480,480)
```

```
      COMMON /POINTS/ JPT(16,16)
      DIMENSION LMAP(30,30)
      DATA N16 /16/
      DATA N30 /30/
      LOBJ = 2**(31-KOBJ)
      DO 500 I=1,16
      DO 400 J=1,16
      IF (.NOT.JPT(I,J)) GO TO 400
      IP = (J-1)*N16+I
C                  EACH RECORD OF FILE 50 REPRESENTS A 30 BY 30 GRID
C                  BLOCK.  LMAP IS CONSTRUCTED IN SUCH A WAY, THAT IF
C                  THE (I,J) BLOCK OF THE BLOCK BELONGS TO OBJECT # LOBJ
C                  ,THEN ,AND ONLY THEN THE (LOBJ-1)-TH BINARY DIGIT OF
C                  LMAP(I,J) IS 1. THE SIGN BIT OF LMAP(I,J) IS THEN
C                  THE ZERO-TH DIGIT AND IT REPRESENTS LAND.
      READ(50'IP) LMAP
      KUP = (I-1)*N30
      LUP = (J-1)*N30
      DO 300 K=1,30
      DO 200 L=1,30
      IF (A(K+KUP,L+LUP)) LMAP(K,L) = LOR(LMAP(K,L),LOBJ)
  200 CONTINUE
  300 CONTINUE
      WRITE(50'IP) LMAP
  400 CONTINUE
  500 CONTINUE
      RETURN
      END
      SUBROUTINE NTRACK(XL,YL,XI,YI)
C     SUBROUTINE TRACK -- VERSION 1.0 -- 14 APR 77.
C     J. R. SLACK, U. S. GEOLOGICAL SURVEY.
C     MARKS EACH CELL ON THE LINE BETWEEN TWO POINTS.
      LOGICAL*1 T/.TRUE./
      LOGICAL*1 A,JPT
      COMMON /MAP/ A(480,480)
      COMMON /PARMS/ A1,B1,C1,A2,B2,C2,MX,NY
      COMMON /POINTS/ JPT(16,16)
      LX=XL
      LY=YL
      IX=XI
      IY=YI
C     TRACK THRU EACH CELL BETWEEN THE OLD POINT
C        AND THE NEW POINT.
      IF (IX.EQ.LX) GO TO 30
C     NON-VERTICAL PATH.
      SLOPE=(YI-YL)/(XI-XL)
      IXDO=IABS(IX-LX)+1
      IXADD=ISGN(IX-LX)
      XADD=IXADD
```

```
      IYADD=ISGN(IY-LY)
      KX=LX-IXADD
      XN=LX
      IF (XADD.LT.0.0) XN=XN+1.0
      DO 20 KK=1,IXDO
      XN=XN+XADD
      KX=KX+IXADD
      IF (KK.EQ.IXDO) XN=XI
      KY=(XN-XL)*SLOPE+YL
      IYDO=IABS(KY-LY)+1
      LY=LY-IYADD
      DO 10 LL=1,IYDO
      LY=LY+IYADD
C     RECORD THE TRACK THRU THIS CELL.
      IF (KX.LT.0.OR.KX.GE.MX.OR.LY.LT.0.OR.LY.GE.NY) GO TO 10
      A(KX+1,LY+1) = T
      JPT(KX/30+1,LY/30+1) = T
   10 CONTINUE
      LY=KY
   20 CONTINUE
      RETURN
C     VERTICAL PATH.
   30 CONTINUE
      IYADD=ISGN(IY-LY)
      IYDO=IABS(IY-LY)+1
      LY=LY-IYADD
      DO 40 LL=1,IYDO
      LY=LY+IYADD
C     RECORD THE TRACK THRU THIS CELL.
      IF (IX.LT.0.OR.IX.GE.MX.OR.LY.LT.0.OR.LY.GE.NY) GO TO 40
      A(IX+1,LY+1) = T
      JPT(IX/30+1,LY/30+1) = T
   40 CONTINUE
      RETURN
      END
```

# Appendix B

## Map Conversion Subroutines

Disclaimer for Distribution of Programs

Although these programs have been tested by the Geological Survey, United States Department of the Interior, no warranty, expressed or implied, is made by the Geological Survey as to the accuracy and functioning of the programs and related program material, nor shall the fact of distribution constitute any such warranty, and no responsibility is assumed by the Geological Survey in connection therewith.

Subroutine: CALLL1
Purpose: To convert a point from Albers equal area coordinates
for the continental United States to geographical
coordinates.

```
      SUBROUTINE CALLL1 (X,Y,PHI,LAMDA,IERR)
C     CONVERT ALBERS EQUAL AREA PROJECTION TO LATITUDE/LONGITUDE.
C        THIS SUBROUTINE ASSUMES THAT THE X,Y POINTS CORRESPOND TO
C        A PROJECTION WITH STANDARD PARALLELS OF 29.5 AND 45.5 DEGREES,
C        WITH RHOO EQUAL TO ZERO. IT IS INTENDED FOR USE WITH MAPS
C        OF THE CONTINENTAL UNITED STATES.
C
C     THIS PROGRAM WAS DESIGNED FOR SINGLE PRECISION, HIGH SPEED
C        CONVERSIONS.
C
C     ARGUMENTS:
C        X,Y: ALBERS COORDINATES.
C        PHI,LAMDA: LAT/LONG (SECONDS)
C        IERR: ERROR FLAG (NONZERO EQUALS ERROR)
C
      REAL LAMDA
C     PARAMETERS OF THE CLARKE 1866 SPHEROID.
      REAL BETA2/0.602837/
      REAL GAMMA/0.134661985E15/
      REAL THETA/0.15091816E15/
C     SECONDS TO RADIANS.
      REAL SECRAD/4.848137E-6/
C
C     CLEAR THE ERROR FLAG.
      IERR=0
C
C     CALCULATE LATITUDE AND LONGITUDE.
      RHOSQ=X**2+Y**2
      LAMDA=ARCOS(X/(-SQRT(RHOSQ)))/BETA2
      PHI=ARSIN((THETA-RHOSQ)/GAMMA)
C
C     CONVERT RADIANS TO SECONDS.
      LAMDA=LAMDA/SECRAD
      PHI=PHI/SECRAD
C
      RETURN
      END
```

```
Subroutine: CLLAL1
Purpose: To convert a point from geographical to Albers
           equal area coordinates for the continental
           United States.

      SUBROUTINE CLLAL1 (PHIS,LAMDAS,X,Y,IERR)
C
C     CONVERT LATITUDE/LONGITUDE TO ALBERS EQUAL AREA PROJECTION,
C        USING THE STANDARD PARALLELS OF 29.5 AND 45.5 DEGREES.
C        THIS MAP PROJECTION IS COMMONLY USED FOR THE CONTINENTAL
C        UNITED STATES.
C
C     LATITUDE/LONGITUDE IS IN SECONDS.
C     THE ALBERS X,Y HAS PHIO EQUAL TO ZERO.
C     THIS SUBROUTINE WAS DESIGNED FOR SINGLE PRECISION, HIGH SPEED
C        CONVERSIONS.
C
C     ARGUMENTS:
C        PHIS,LAMDAS: LATITUDE, LONGITUDE (SECONDS).
C        X,Y: ALBERS COORDINATES.
C        IERR: ERROR FLAG (NONZERO EQUALS ERROR).
C
      REAL LAMDA,LAMDAS
C     AUTHALIC RADIUS.
C     SECONDS TO RADIANS.
      REAL SECRAD/4.848137E-6/
C     PARAMETERS OF THE CLARKE 1866 SPHEROID, WITH STANDARD
C        PARALLELS OF 29.5 AND 45.5 DEGREES NORTH, AND RHOO
C        EQUAL TO ZERO. THIS IS THE STANDARD PROJECTION USED FOR THE
C        CONTINENTAL UNITED STATES.
      REAL BETA2/0.602837/
      REAL RHO1SQ/0.84607444E14/
      REAL GAMMA/0.13466195E15/
      REAL SP1/0.49242356/
C
C     CLEAR THE ERROR FLAG.
      IERR=0
C
C     CHECK FOR VALID LATITUDE.
      IF (PHIS.LT.324000.0) GO TO 10
      IERR=1
      GO TO 90
   10 CONTINUE
C     CONVERT SECONDS TO RADIANS.
      PHI=PHIS*SECRAD
      LAMDA=LAMDAS*SECRAD
C
C     ALBERS PROJECTION EQUATIONS.
C     REFERENCE: MAP PROJECTIONS; PETER RICHARDUS AND RON K. ADLER;
```

```
C        NORTH HOLLAND PUBLISHING CO.; AMSTERDAM; 1972;
C        PAGE 166.
C     EQUATIONS MODIFIED BY K. LANFEAR TO FIT STANDARD MAP
C        AND TO ACHEIVE HIGHER SPEED.
      RHO=SQRT(RHO1SQ+GAMMA*(SP1-SIN(PHI)))
      BL=BETA2*LAMDA
      X=-RHO*COS(BL)
      Y=RHO*SIN(BL)
C
   90 CONTINUE
      RETURN
      END
```

Subroutine: CLLME
Purpose: To convert a point from geographical to Mercator coordinates.

```
      SUBROUTINE CLLME (SLAT,SLON,X,Y,ERROR)
C
C     CONVERTS A SET OF POINTS FROM LAT-LONG TO MERCATOR.
C
C     ARGUMENTS:
C        SLAT,SLON      - INPUT LATITUDE AND LONGITUDE.
C        X,Y            - COORDINATES IN MERCATOR GRID (METERS).
C                         NEGATIVE IS SOUTHERN AND/OR EASTERN HEMISPHERES.
C
C     WARNING: THIS SUBROUTINE IS DESIGNED TO PROCESS A VERY LARGE
C        NUMBER OF POINTS, AND EMPHASIZES SPEED OVER EXTREME ACCURACY.
C
C
      REAL SECRAD/4.848137E-6/,PION4/0.7853982/
C     PARAMETERS OF THE CLARKE (1866) SPHEROID.
      REAL A/6378206.4/,EPSIL/8.227185E-2/
      REAL SECMET/30.92241/,SECRA2/2.424068E-6/,EPSIL2/4.113593E-2/
C
C     RESET THE ERROR FLAG.
      ERROR=0
C
      X=SECMET*SLON
C     ADJUST IF SOUTHERN HEMISPHERE.
      HEM=ISGN(SLAT)
      SLAT=ABS(SLAT)
C     CHECK FOR LATITUDE GREATER THAN 90 DEGREES.
      IF (SLAT.GE.324000.) ERROR=1
      ESINPH=EPSIL*SIN(SLAT*SECRAD)
      Y=A*HEM*ALOG(TAN(PION4+SLAT*SECRA2)*
     1   ((1.0-ESINPH)/(1.0+ESINPH))**EPSIL2)
C
C     RETURN THE CONVERTED DATA.
C     X AND Y ARE IN METERS.
      RETURN
      END
```

Subroutine: CLLUT
Purpose: To convert a point from grographical coordinates to
Universal Transverse Mercator (UTM) coordinates.

```
      SUBROUTINE CLLUT (Y,X,IZONE,SLAT,SLON,LERR)
C     THIS SUBROUTINE TRANSFORMS LATITUDE AND LONGITUDE TO UNIVERSAL
C     TRANSVERSE MERCATOR COORDINATES
C     NOTE: SOUTHERN HEMISPHERE LATITUDES AND EASTERN HEMISPHERE LONGITUDES
C     ARE CONSIDERED NEGATIVE AND MUST BE INPUT AS SUCH
C
      INTEGER UOUT/6/
      REAL A1/-.5104691E-2/,A2/0.2173749E-4/,A3/-0.1152053E-6/
      REAL A4/0.6545693E-9/
      REAL A5/5.0E5/,A7/0/,A8/0.9996/,A10/6367399.0/
      REAL A16/0.67688227E-2/,A15/6378206.4/,A17/0.6814949E-2/
C
      LOGICAL*4LERR
      LERR=.FALSE.
C
C
C
C     COMPUTE UTM ZONE (IZONE) AND CENTRAL MERIDIAN IN SECONDS (A9) FOR
C     GEODETIC TO UTM CONVERSION WHERE ZONE IS NOT INPUT
      IZONE=(180-SLON/3600.0)/6+1
      UTZ=30.0-IZONE
      A9=((UTZ*6.0)+3.0)*3600.0
      A6=0.0
      IF (SLAT.LT.0) IZONE=-IZONE
      IF (SLAT.LT.0) A6=1.0E7
C
C
C     THE NEXT GROUP OF STATEMENTS WAS FORMRLY SUBROUTINE TMFWD OF J380
C
C
      B10=(A9-SLON)*4.848137E-6
      IF (ABS(B10).GT.0.16) GO TO 902
      IF (ABS(SLAT).GT.302400.0) GO TO 902
      B9=SLAT*4.848137E-6
      SINP=SIN(B9)
      COSP=COS(B9)
      RN=A15/SQRT(1.0-A16*SINP**2)
      T=SINP/COSP
      TS=T**2
      TS2=TS**2
      TS3=TS2*TS
      B11=COSP**2
      B11SQ=B11**2
      B11CU=B11SQ*B11
      ETAS=A17*B11
```

```
      B1=RN*COSP
      B3=(1.0-TS+ETAS)*B1*B11/6.0
      B5=(TS2-18.0*TS+5.0+(14.0-58.0*TS)*ETAS)*B1*B11SQ/120.0
      B7=(179.0*TS2-TS3-479.0*TS+61.0)*B1*B11CU/5040.0
      B12=B10**2
      B12SQ=B12**2
      B12CU=B12SQ*B12
      X=(B7*B12CU+B5*B12SQ+B3*B12+B1)*B10*A8+A5
      B2=RN*B11*T/2.0
      B4=(ETAS*9.0+4.0*ETAS**2+5.0-TS)*B2*B11/12.0
      B6=(TS2-58.0*TS+61.0+(270.0-330.0*TS)*ETAS)*B2*B11SQ/360.0
      B8=(543.0*TS2-TS3-3110.0*TS+1385.0)*B2*B11CU/20160.0
      Y=(B8*B12SQ*B12SQ+B6*B12CU+B4*B12SQ+B2*B12)+((A4*B11CU+A3*B11SQ+
     1A2*B11+A1)*SINP*COSP+B9)*A10
      Y=(Y-A7)*A8+A6
C
C
      RETURN
C
  902 CONTINUE
C
C
C     ERROR ROUTINES
      WRITE (UOUT,6902) SLAT,SLON
 6902 FORMAT (' ERROR - INVALID COORDINATES TO SUBROUTINE CLLUT'/
     1  'LATITUDE = ',F10.2,'  LONGITUDE = ',F10.2)
      LERR=.TRUE.
      Y=0.0
      X=0.0
      RETURN
      END
```

Subroutine: CMELL
Purpose: Convert a point from Mercator to geographical coordinates.

```
      SUBROUTINE CMELL (X,Y,SLAT,SLON,ERROR)
C
C     CONVERTS A SET OF POINTS FROM MERCATOR TO LAT-LONG.
C
C     ARGUMENTS
C        X,Y            - COORDINATES IN MERCATOR GRID (METERS).
C        SLAT,SLON      - LATITUDE AND LONGITUDE (SECONDS).
C                         NEGATIVE IS SOUTHERN AND/OR EASTERN HEMISPHERES
C
C     WARNING: THIS SUBROUTINE IS DESIGNED TO PROCESS A VERY LARGE
C        NUMBER OF POINTS, AND EMPHASIZES SPEED OVER EXTREME ACCURACY.
C
C
      REAL PION4/0.7853982/
      INTEGER ERROR
C     PARAMETERS OF CLARKE (1866) SPHEROID.
      REAL A/6378206.4/
      REAL EPSIL/8.227185E-2/
      REAL SECMET/30.92241/
      REAL EPSIL2/4.113593E-2/
      REAL SECRAD/4.848137E-6/
      REAL PONE/0.6981317/,RHO/0.99565460/
C     NOTE: IT IS EXPECTED THAT THIS SUBROUTINE WILL BE CALLED
C        REPEATEDLY FOR CONVERTING FILES OF POINTS THAT ARE NOT VERY
C        FAR APART. THEREFORE, TO REDUCE ITERATIONS, THE VALUES OF
C        PONE AND RHO ARE RETAINED FROM THE PREVIOUS CALL. THE DECLARED
C        VALUES ARE USED ONLY FOR THE FIRST CALL.
C
C     RESET THE ERROR FLAG.
      ERROR=0
C
C     CHECK FOR LATITUDE LESS THAN 85 DEGREES.
      IF (Y.GT.8.7E6) GO TO 910
C
C
C     FIND LONGITUDE.
      SLON=X/SECMET
C
C
C     FIND LATITUDE.
C     ADJUST IF SOUTHERN HEMISPHERE.
      HEM=ISGN(Y)
      YTAN=EXP(ABS(Y)/A)
C
C     ITERATIVE SOLUTION FOR ELLIPTICITY OF EARTH.
   10 CONTINUE
```

```
      PTWO=2*(ATAN(YTAN/RHO)-PION4)
      IF ((ABS(1.0-PONE/PTWO)).LE.0.000001) GO TO 20
C     ITERATION CONTINUES TO SEVEN PLACE PRECISION.
      PONE=PTWO
      ESINPH=EPSIL*SIN(PTWO)
      RHO=((1.0-ESINPH)/(1.0+ESINPH))**EPSIL2
      GO TO 10
C
C     ITERATION COMPLETED.
   20 CONTINUE
      SLAT=PONE*HEM/SECRAD
      RETURN
C
C     ERROR - WRONG LATITUDE.
  910 CONTINUE
      ERROR=1
      RETURN
      END
```

```
Subroutine: CUTLL
Purpose: To convert a point from Universal Transverse Mercator
           (UTM) coordinates to geographical coordinates.

      SUBROUTINE CUTLL (X,Y,IZONE,SLAT,SLON,LERR)
C
C     CONVERTS A SET OF COORDINATES FROM UNIVERSAL TRANSVERSE MERCATOR
C        TO LATITUDE AND LONGITUDE.
C     BASED ON PROGRAM J380, PROVIDED BY THE BRANCH OF FIELD SURVEYS.
C
C     WARNINGS.................
C        THIS PROGRAM WAS DESIGNED TO TRANSFORM THE COORDINATES OF A
C     VERY LARGE NUMBER OF POINTS. CONSEQUENTLY, IT EMPHASIZES HIGH
C     SPEED OVER EXTREME ACCURACY. BESIDES SUBSTANTIAL CHANGES IN
C     THE INPUT/OUTPUT ROUTINES, THE PRINCIPAL DIFFERENCES BETWEEN
C     THIS PROGRAM AND J380 ARE AS FOLLOWS:
C        1. SINGLE PRECISION IS USED. THIS IS SUFFICIENTLY ACCURATE
C           FOR MOST SEDDS USES.
C        2. THE CONVERSION IS ONLY "ONE WAY", I. E. ANOTHER
C           PROGRAM MUST BE USED TO CONVERT FROM LAT-LONG TO UTM.
C        3. INTERNAL CODING OF J380 HAS BEEN OPTIMIZED, AND THE
C           FORTRAN IV(H) OPTIMIZER, LEVEL 2, IS USED.
C        4. ONLY THE CLARKE 1866 SPHEROID IS ALLOWED.
C        5. CERTAIN STATEMENTS HAVE BEEN RE-CODED TO REDUCE ROUNDOFF.
C
C     REQUIRES ABOUT 2.1 MILLISECONDS OF CPU TIME PER POINT,
C        OR 35 MINUTES FOR ONE MILLION POINTS.
C
C
C
C
C         IZONE - - UTM ZONE (SOUTHERN HEMISPHERE IS NEGATIVE)
C         X,Y   - - EASTING, NORTHING
C     NOTE: CUSTOMARY TO WRITE NORTHING BEFORE EASTING.
C
C
      INTEGER UOUT/6/
      REAL A5/5.0E5/,A7/0/,A8/0.9996/
C     PARAMETERS OF CLARKE 1866 SPHEROID.
C     CHANGE THESE TO USE ANOTHER SPHEROID.
      REAL A10/6367399.0/,A11/0.50787702E-2/,A12/0.30026989E-4/,
     1   A13/0.24297748E-6/,A14/0.22795696E-8/,A16/0.67688227E-2/,
     2   A15LIM/1.27564128/,A8E6/999600.0/,A15EM6/6.3782064/,
     3   B1/6356583.8/
C
C
C
C     RESET ERROR FLAG
      LERR=.FALSE.
```

```
C
C     ADJUST IF SOUTHERN HEMISPHERE.
C     NOTE: CLARKE 1866 SHPERIOD IS NOT USUALLY USED OUTSIDE OF U.S..
      A6=0.0
      IF (IZONE.LT.0) A6=1.0E7
C
C
C     COMPUTE THE CENTRAL MERIDIAN
      A9=(((30-IABS(IZONE))*6)+3)*3600
C
C     THE NEXT GROUP OF STATEMENTS WAS, FORMERLY, SUBROUTINE
C        TMINV OF J380
      B9=(A5-X)/A8E6
      IF (ABS(B9).GT.A15LIM) GO TO 902
      B10=((Y-A6)/A8+A7)/A10
      IF (ABS(B10).GT.1.47) GO TO 902
      SINW=SIN(B10)
      COSW=COS(B10)
      B12=COSW**2
      B11=(A14*B12**3+A13*B12**2+A12*B12+A11)*SINW*COSW+B10
      SINW=SIN(B11)
      COSW=COS(B11)
      RN=SQRT(1.0-A16*SINW**2)/A15EM6
      RN2=RN**2
      RN4=RN2**2
      RN6=RN2*RN4
      T=SINW/COSW
      TS=T**2
      B12=COSW**2
      ETAS=A16*B12/(1.0-A16)
      B1=RN/COSW
      B2=-T*(1.0+ETAS)*RN2  /2.0
      B3=-(1.0+2.0*TS+ETAS)*B1*RN2  /6.0
      B4=(((-6.0-ETAS*9.0)*ETAS+3.0)*TS+(6.0-ETAS*3.0)*ETAS+5.0)*
     1   T*RN4/24.0
      B5=((TS*24.0+ETAS*8.0+28.0)*TS+ETAS*6.0+5.0)*B1*RN4  /120.0
      B6=(((ETAS*45.0-45.0)*TS+ETAS*162.0-90.0)*TS-ETAS*107.0-61.0)*
     1   T*RN6/720.0
      B7=-(((TS*720.0+1320.0)*TS+662.0)*TS+61.0)*B1*RN6  /5040.0
      B8=(((TS*1575.0+4095.0)*TS+3633.0)*TS+1385.0)*T*RN4**2/40320.0
      B10=B9**2
      SLAT=((((B8*B10+B6)*B10+B4)*B10+B2)*B10+B11)*206264.8
      SLON=(((B7*B10+B5)*B10+B3)*B10+B1)*B9*206264.8+A9
C
C
      RETURN
C
C
C     ERROR ROUTINES.
```

```
 902 CONTINUE
     WRITE (UOUT,6902) IZONE,Y,X
6902 FORMAT (' ERROR - INVALID COORDINATES TO SUBROUTINE CUTLL'/
    1   ' ZONE = ',I5,'   NORTHING = ',F10.2,
    2   '   EASTING = ',F10.2)
     LERR=.TRUE.
     SLAT=0.0
     SLON=0.0
     RETURN
     END
```

# Appendix C

Plotting Subroutines

## Disclaimer for Distribution of Programs

Although these programs have been tested by the Geological Survey, United States Department of the Interior, no warranty, expressed or implied, is made by the Geological Survey as to the accuracy and functioning of the programs and related program material, nor shall the fact of distribution constitute any such warranty, and no responsibility is assumed by the Geological Survey in connection therewith.

Subroutine: ALPHA2
Purpose: Write text on a plot.
Argument list: (X,Y,YHGHT,THETA,NCHAR,ICHAR)

| | |
|---|---|
| X,Y | - location of lower left corner of text |
| YHGHT | - height of letters |
| THETA | - angle of rotation (degrees counterclockwise) |
| NCHAR | - number of characters in text |
| ICHAR | - a vector with up to 256 bytes of text |

Subroutine: ARROW2
Purpose: Draw an arrow.
Argument list: (X,Y,THETA,SIZE,NCODE)

| | |
|---|---|
| X,Y | - location of arrow (see NCODE) |
| THETA | - angle of rotation (degrees counterclockwise from horizontal) |
| SIZE | - total length of arrow, tip to tail |
| NCODE | - 1 if x,y at head<br>2 if x,y at middle<br>3 if x,y at tail |

Subroutine: CIRC
Purpose: Draw a circle.
Argument list: (X,Y,R)

| | |
|---|---|
| X,Y | - center of circle |
| R | - radius |

Subroutine: DONE
Purpose: Terminate a plot.
Argument list: none

Subroutine: INIT
Purpose: Initialize the plotting subroutine package.
Argument list: (IOFLAG)

IOFLAG - flag for output device (1 for magnetic tape)

Subroutine: LINE2
Purpose: Move to x,y with the pen down.
Argument list: (X,Y)

X,Y - coordinates of x,y

Subroutine: MOV
Purpose: Move to x,y with the pen up.
Argument list: (X,Y)

X,Y - coordinates of x,y

Subroutine: NORTH
Purpose: Draw a north arrow.
Argument list: (X,Y,THETA,SIZE)

| | |
|---|---|
| X,Y | - location of arrow center |
| THETA | - angle of rotation (degrees counterclockwise from horizontal) |
| SIZE | - length of arrow |

Subroutine: PAT1
Purpose: Draw a horizontal line in each cell that contains a specific target.

```
      SUBROUTINE PAT1 (XORIG,YORIG,IBX,IBY,BITP)
C
C     PLOTS A HORIZONTAL LINE IN EACH CELL THAT CONTAINS TARGET BITP.
C
      LOGICAL PENDWN
      INTEGER A(30,30)
      COMMON /PAT/A
C
C     ESTABLISH THE OFFSETS FOR THE BLOCK.
      XOFF=XORIG+(IBX-1)*30.0-1.0
      YOFF=YORIG+(IBY-1)*30.0-0.5
C
C     PROCESS ACROSS A ROW.
      DO 10 J=1,30
C     PENDWN TRUE MEANS THE PEN IS DOWN.
      PENDWN=.FALSE.
C
C     CHECK EACH ELEMENT OF THE ROW.
      DO 20 I=1,30
C
C     TEST FOR THE PRESENCE OF THE TARGET.
      IF (LAND(A(I,J),BITP).EQ.0) GO TO 30
C
C     TARGET IS PRESENT IN THIS CELL.
C     IS THE PEN UP OR DOWN?
      IF (PENDWN) GO TO 20
C     THE PEN IS UP. MOVE IT TO THE STARTING POINT.
      CALL MOV ((XOFF+I),(YOFF+J))
      PENDWN=.TRUE.
      GO TO 20
C
C     THE TARGET IS NOT PRESENT IN THIS CELL.
   30 CONTINUE
C     IF THE PEN IS DOWN, COMPLETE THE LAST LINE.
      IF (PENDWN) CALL LINE2 ((XOFF+I),(YOFF+J))
      PENDWN=.FALSE.
C
   20 CONTINUE
C
C     END OF A ROW.
C     IF THE PEN IS DOWN, COMPLETE THE LINE.
      IF (PENDWN) CALL LINE2 ((XOFF+31.0),(YOFF+J))
   10 CONTINUE
C
      RETURN
```

```
END
```

```
Subroutine: P2
Purpose: Draw a vertical line in each cell that contains a
           specific target.

      SUBROUTINE PAT2 (XORIG,YORIG,IBX,IBY,BITP)
C
C     PLOTS A VERTICAL LINE IN EACH CELL THAT CONTAINS TARGET BITP.
C
      LOGICAL PENDWN
      INTEGER A(30,30)
      COMMON /PAT/A
C
C     ESTABLISH THE OFFSETS FOR THE BLOCK.
      XOFF=XORIG+(IBX-1)*30.0-0.5
      YOFF=YORIG+(IBY-1)*30.0-1.0
C
C     PROCESS DOWN A COLUMN.
      DO 10 I=1,30
C     PENDWN TRUE MEANS THE PEN IS DOWN.
      PENDWN=.FALSE.
C
C     CHECK EACH ELEMENT OF THE COLUMN.
      DO 20 J=1,30
C
C     TEST FOR THE PRESENCE OF THE TARGET.
      IF (LAND(A(I,J),BITP).EQ.0) GO TO 30
C
C     TARGET IS PRESENT IN THIS CELL.
C     IS THE PEN UP OR DOWN?
      IF (PENDWN) GO TO 20
C     THE PEN IS UP. MOVE IT TO THE STARTING POINT.
      CALL MOV ((XOFF+I),(YOFF+J))
      PENDWN=.TRUE.
      GO TO 20
C
C     THE TARGET IS NOT PRESENT IN THIS CELL.
   30 CONTINUE
C     IF THE PEN IS DOWN, COMPLETE THE LAST LINE.
      IF (PENDWN) CALL LINE2 ((XOFF+I),(YOFF+J))
      PENDWN=.FALSE.
C
   20 CONTINUE
C
C     END OF A COLUMN.
C     IF THE PEN IS DOWN COMPLETE THE LINE.
      IF (PENDWN) CALL LINE2 ((XOFF+I),(YOFF+31.0))
   10 CONTINUE
C
      RETURN
```

```
END
```

Subroutine: PAT3
Purpose: Draw a cross in each cell that contains a specific target.

```
      SUBROUTINE PAT3 (XORIG,YORIG,IBX,IBY,BITP)
C
C     DRAWS A HORIZONTAL AND A VERTICAL LINE IN EACH CELL THAT
C        CONTAINS TARGET BITP.
C
      INTEGER A(30,30)
      COMMON /PAT/A
C
C     DRAW HORIZONTAL LINES WITH SUBROUTINE PAT1.
      CALL PAT1 (XORIG,YORIG,IBX,IBY,BITP)
C     DRAW VERTICAL LINES WITH SUBROUTINE PAT2.
      CALL PAT2 (XORIG,YORIG,IBX,IBY,BITP)
C
      RETURN
      END
```

Subroutine: PLOTMP
Purpose: Plot a digitized outline map.

```
      SUBROUTINE PLOTMP (STUDY,XORIG,YORIG,IGR,ILL,UPLOT)
C
C
C     PLOTS A DIGITIZED MAP, WITH ITS ORIGIN AT THE SPECIFIED POINT.
C     USES THE GERBER SUBROUTINE LIBRARY, AUGMENTED BY SEDDS SUBROUTINES.
C     DESIGNED TO PROCESS A FILE OF DIGITIZED DATA FROM PROGRAM CFMAGR.
C     L1,L2 ARE VARIABLES WHICH INDICATE WHICH POINTS TO PLOT
C
C
      REAL STUDY(2),FILE(15),STUDYB(2)
      REAL X(100),Y(100)
      INTEGER WHEN(7)
      INTEGER IPN(100)
      INTEGER JLEV(100)
      INTEGER UPLOT
      INTEGER UOUT/6/
C
C
C     READ THE HEADER RECORD OF THE MAP FILE
      READ (UPLOT) STUDYB,FILE,WHEN,IX,IY,A1,B1,C1,A2,B2,C2
C     INTERLOCK TO PREVENT USING THE WRONG FILE.
      IF (STUDY(1).NE.STUDYB(1).OR.STUDY(2).NE.STUDYB(2)) GO TO 990
      WRITE (UOUT,6001) FILE,WHEN,XORIG,YORIG
 6001 FORMAT ('OPLOT MAP FILE:',15A4/
     1   ' CREATED: ',I2,A3,I2,5X,I2,':',I2,':',I2,1X,A2/
     2   ' POSITION OF BASE MAP ORIGIN ON PLOT: X = ',F10.2,
     3   5X,'Y = ',F10.2)
C
C     LIMITS OF PLOT TO FLOATING POINT.
      XMAX=IX
      YMAX=IY
C
C     DRAW A BOX AROUND THE PLOT.
      CALL MOV (XORIG,YORIG)
      CALL LINE2 ((XORIG+XMAX),YORIG)
      CALL LINE2 ((XORIG+XMAX),(YORIG+YMAX))
      CALL LINE2 (XORIG,(YORIG+YMAX))
      CALL LINE2 (XORIG,YORIG)
C
C     READ AND PLOT THE MAP FILE
   20 CONTINUE
C     POINTS ARE READ IN BLOCKS OF 100.
C        N = NUMBER OF POINTS IN THE BLOCK, USUALLY 100.
C        IPN = 1 IF PEN DOWN, -1 IF PEN UP.
C              0 IF THE POINT IS TO BE IGNORED.
      READ (UPLOT,END=50) N,IPN,JLEV,X,Y
```

```
C     ZERO INDICATES END OF A FILE, BUT COULD STILL BE CONCATENATED.
      IF (N.EQ.0) GO TO 20
      DO 30 I=1,N
      IF(IPN(I).EQ.0) GO TO 30
      IF(IPN(I).LE.0) CALL MOV ((X(I)+XORIG),(Y(I)+YORIG))
      CALL SEL(10)
      IF(JLEV(N).GE.6.AND.JLEV(N).LE.10)CALL SEL(11)
      IF(IPN(I).GT.0) CALL LINE2 ((X(I)+XORIG),(Y(I)+YORIG))
   30 CONTINUE
C     GO BACK FOR ANOTHER RECORD.
      GO TO 20
C
C
C     PLOT A GRID IF IGR IS SPECIFIED.
   50 CONTINUE
      CALL SEL(10)
      CALL FONT2 (3.0,1.8,0.0)
      IF (IGR.LE.0) GO TO 90
C     CONFIRM GRID PLOT ON PRINTOUT.
      WRITE (UOUT,6002) IGR
 6002 FORMAT (' GRID SPECIFIED WITH ',I4,' UNIT DIVISIONS.')
C
C     GRID MUST BE WITHIN LIMITS.
      IF (IGR.GT.IX.OR.IGR.GT.IY) GO TO 90
C
C     PLOT LINES PERPENDICULAR TO X-AXIS.
C     NUMBER OF LINES TO DRAW.
      ILN=IX/IGR
C     DIRECTION REVERSER FOR FASTER PLOTTING.
      IDIR=1
C     START AT FIRST GRID LINE.
      XG=IGR+XORIG
      XG=IGR+XORIG
      CALL MOV(XG,YORIG)
C     DO FOR EACH LINE.
      DO 60 I=1,ILN
C     FIND THE UPPER OR LOWER Y LIMIT.
      YG=YMAX*0.5+IDIR*YMAX*0.5+YORIG
C     DRAW THE LINE.
      CALL LINE2 (XG,YG)
C     MOVE TO THE START OF THE NEXT LINE.
      XG=IGR*(I+1)+XORIG
      IF (XG.LE.(XMAX+XORIG)) CALL MOV (XG,YG)
C     REVERSE THE DIRECTION OF DRAWING FOR THE NEXT LINE.
      IDIR=-IDIR
   60 CONTINUE
C     LABEL THE GRID LINES ALONG THE X-AXIS.
      DO 61 I=1,ILN
      XG=I*IGR+XORIG
```

```
      IPLT=I*IGR
      CALL FXPLT (XG,YORIG,90.0,IPLT)
   61 CONTINUE
C
C     PLOT LINES PERPENDICULAR TO THE Y-AXIS.
C     USES THE SAME ROUTINE AS IS USED FOR THE X-AXIS.
      ILN=IY/IGR
      IDIR=1
      YG=IGR+YORIG
      CALL MOV(XORIG,YG)
      DO 70 I=1,ILN
      XG=XMAX*0.5+IDIR*XMAX*0.5+XORIG
      CALL LINE2 (XG,YG)
      YG=IGR*(I+1)+YORIG
      IF (YG.LE.(YMAX+YORIG)) CALL MOV (XG,YG)
      IDIR=-IDIR
   70 CONTINUE
      DO 71 I=1,ILN
      YG=I*IGR+YORIG
      IPLT=I*IGR
C     NUMBERS MUST NOT EXTEND BEYOND THE MAP BOUNDARIES.
      IF ((YG+3.0).LE.(YMAX+YORIG)) CALL FXPLT (XORIG,YG,0.0,IPLT)
   71 CONTINUE
C
   90 CONTINUE
C
C     SEE IF LATITUDE AND LONGITUDE MERIDIANS ARE DESIRED.
      IF (ILL.LE.0)  GO TO 190
C
C     DRAW LATITUDES AND LONGITUDES AT 1 DEGREE INTERVALS.
C
C     FIND THE WESTERN LONGITUDE AND SOUTHERN LATITUDE OF THE BASE MAP.
      XM=-A1/B1
      YM=-A2/C2
      CALL CMELL (XM,YM,SLAT,SLON)
C
C
C     MERIDIANS OF LONGITUDE.
C     STARTING LONGITUDE.
      ISLON=SLON/3600
C
C     CONVERT LONGITUDE BACK TO GRID COORDINATES.
  100 CONTINUE
      SLON=ISLON*3600
      CALL CLLME (SLAT,SLON,XM,YM,IERR)
      XG=A1+B1*XM
C
C     TEST IF LONGITUDE IS WITHIN THE BASE MAP.
      IF (XG.GT.(XMAX+0.000001)) GO TO 110
```

```
      IF (XG.LT.-0.000001) GO TO 101
C
C     MOVE THE PEN TO THE STARTING POINT.
      CALL MOV ((XG+XORIG),YORIG)
C
C     DRAW THE MERIDIAN AS A DASHED LINE.
      DO 114 I=1,IY,3
      YVAL=I
      CALL LINE2 ((XG+XORIG),(YORIG+YVAL))
      CALL LINE2 ((XG+XORIG),(YORIG+YVAL+1.0))
      CALL MOV ((XG+XORIG),(YORIG+YVAL+2.0))
  114 CONTINUE
C
C     WRITE THE LONGITUDE IN DEGREES.
C     A MAXIMUM OF 3 FIXED-POINT NUMBERS WILL BE DRAWN, ALONG WITH
C     A DEGREE SYMBOL. THIS WILL OCCUPY A SPACE 3.0 UNITS HIGH AND
C     10.2 UNITS WIDE.
      IF (ISLON.GE.100) CALL FXPLT ((XG+XORIG+1.5),(YORIG-10.2),
     1   90.0,ISLON)
      IF (ISLON.LT.100) CALL FXPLT ((XG+XORIG+1.5),(YORIG-8.4),
     1   90.0,ISLON)
      CALL CIRC ((XG+XORIG-1.0),(YORIG-1.9),0.5)
  101 CONTINUE
C
C     INCREMENT THE LONGITUDE.
      ISLON=ISLON-1
C
C     RETURN FOR ANOTHER MERIDIAN.
      GO TO 100
C
C
C     MERIDIANS OF LATITUDE.
  110 CONTINUE
C
C     STARTING LATITUDE.
      ISLAT=SLAT/3600.0
C
C     CONVERT LATITUDE BACK TO GRID COORDINATES.
  111 CONTINUE
      SLAT=ISLAT*3600
      CALL CLLME (SLAT,SLON,XM,YM,IERR)
      YG=A2+C2*YM
C
C     TEST IF LATITUDE IS WITHIN THE BASE MAP.
      IF (YG.GT.(YMAX+0.000001)) GO TO 120
      IF (YG.LT.-0.000001) GO TO 112
C
C     MOVE THE PEN TO THE STARTING POINT.
      CALL MOV (XORIG,(YG+YORIG))
```

```
C
C     DRAW THE MERIDIAN AS A DASHED LINE.
      DO 115 I=1,IX,3
      XVAL=I
      CALL LINE2 ((XORIG+XVAL),(YG+YORIG))
      CALL LINE2 ((XORIG+XVAL+1.0),(YG+YORIG))
      CALL MOV ((XORIG+XVAL+2.0),(YG+YORIG))
  115 CONTINUE
C
C     WRITE THE LATITUDE IN DEGREES.
C     SEE NOTE FOR LONGITUDES.
      IF (ISLAT.GE.100) CALL FXPLT ((XORIG-10.2),(YG+YORIG-1.5),0.0,
     1   ISLAT)
      IF (ISLAT.LT.100) CALL FXPLT ((XORIG-8.4),(YG+YORIG-1.5),0.0,
     1   ISLAT)
      CALL CIRC ((XORIG-1.9),(YG+YORIG+2.5),0.5)
  112 CONTINUE
C
C     INCREMENT THE LATITUDE.
      ISLAT=ISLAT+1
C
C     RETURN FOR ANOTHER MERIDIAN.
      GO TO 111
  120 CONTINUE
  190 CONTINUE
      RETURN
C
C
C     ERROR ROUTINE FOR A FILE MISMATCH.
  990 CONTINUE
      WRITE (UOUT,6990) STUDY,STUDYB
 6990 FORMAT ('0ERROR- SUBROUTINE BASEMP WAS CALLED WITH STUDY ',2A4,
     1   ', BUT THE BASE MAP FILE WAS FOR STUDY ',2A4,'.')
      RETURN
      END
```

Subroutine: SCAL
Purpose: Set the x-axis and y-axis scale factors.
Argument list: (X,Y)

| | |
|---|---|
| X | - scale factor for x-axis |
| Y | - scale factor for y-axis |

Subroutine: WAIT
Purpose: Temporarily halt the plotter.
Argument list: none

Subroutine: ZER
Purpose: Reset the origin of the plot.
Argument list: (X,Y)

X,Y - coordinates of the new origin, with respect to the drafting surface zero

# Appendix D

## Logic Functions

SEDDS uses three logical functions to form and check matrices in compact storage notation. These are:

```
LAND (A,B)
LOR (A,B)
LXOR (A,B)
```

A and B are each 4 byte arguments. Each function returns a 4 byte (32 bit) integer. Bit i of LAND, LOR or LXOR will be set equal to 1 under the conditions below. Bit i will be zero otherwise.

| Function | Bit i of A | and | Bit i of B |
|---|---|---|---|
| LAND | 1 | | 1 |
| LOR | 1 | | 0 |
| | 0 | or | 1 |
| | 1 | or | 1 |
| LXOR | 0 | | 1 |

# Appendix E

## Matrix Blocking Algorithms

SEDDS stores a 480 x 480 matrix in 256 blocks, each 30 x 30 elements. Using integer arithmetic, the conversion formulas between blocked storage and the full matrix are as follows:

(IX,IY) = cell of the 480 x 480 matrix.

(IBX,IBY) = block coordinates, which are defined as
IBX = IX/30 +1
IBY = IY/30 +1

IREC = record number of the 30 x30 block stored on the direct access file.
= (IBY - 1) * 16 + IBX

(ICX,ICY) = cell within the 30 x 30 block (IBX,IBY).
The relation between this cell and the 480 x 480 cells is as follows:
ICX = IX - (IX/30) * 30
ICY = IY - (IY/30) * 30
IX = (IBX -1) * 30 + ICX
IY = (IBY -1) * 30 + 31 - ICY