AD-A037 116 MITRE CORP BEDFORD MASS F/G 8/2
GEOGRAPHIC DATA BASE DEVELOPMENT.(U)
JAN 77 A M MOLLOY F19628-76-C-0001
UNCLASSIFIED MTR-3312 ESD-TR-76-360 NL

1 OF 2
AD
A037116

ESD-TR-76-360

MTR-3312



# GEOGRAPHIC DATA BASE DEVELOPMENT

JANUARY 1977



Prepared for

DEPUTY FOR DEVELOPMENT PLANS
ELECTRONIC SYSTEMS DIVISION
AIR FORCE SYSTEMS COMMAND
UNITED STATES AIR FORCE
Hanscom Air Force Base, Bedford, Massachusetts

Approved for public release; distribution unlimited.

Project No. 7090

Prepared by
THE MITRE CORPORATION
Bedford, Massachusetts
Contract No. F19628-76-C-0001

## REVIEW AND APPROVAL

**This technical report has been reviewed and is approved for publication.**

Jack Segal

JACK SEGAL, GS-14
Project Engineer

FOR THE COMMANDER

Ronald E. Byrne, Jr.

RONALD E. BYRNE, JR., Colonel, USAF
Director, Advanced Planning
Deputy for Development Plans

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

# REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

| | |
|---|---|
| 1. REPORT NUMBER: ESD-TR-76-360 | 2. GOVT ACCESSION NO. |
| 3. RECIPIENT'S CATALOG NUMBER | |
| 4. TITLE (and Subtitle): GEOGRAPHIC DATA BASE DEVELOPMENT. | 5. TYPE OF REPORT & PERIOD COVERED |
| | 6. PERFORMING ORG. REPORT NUMBER: MTR-3312 |
| 7. AUTHOR(s): A. M. Molloy | 8. CONTRACT OR GRANT NUMBER(s): F19628-76-C-0001 |
| 9. PERFORMING ORGANIZATION NAME AND ADDRESS: The MITRE Corporation, Box 208, Bedford, MA 01730 (187 p.) | 10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: Project No. 7090 |
| 11. CONTROLLING OFFICE NAME AND ADDRESS: Deputy for Development Plans, Electronic Systems Division, AFSC, Hanscom Air Force Base, Bedford, MA 01731 | 12. REPORT DATE: JAN [redacted] 77 |
| | 13. NUMBER OF PAGES: 186 |
| 14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office): Technical rept., | 15. SECURITY CLASS. (of this report): UNCLASSIFIED |
| | 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE |

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

COMPUTER SOFTWARE
DATA BASE
GEOGRAPHY
MAP PROJECTION

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

Project 7090, Operations/Intelligence Techniques Experimentation, has as an objective the development of data processing techniques for effective applications of intelligence data. Because information derived from intelligence data is often positional, one way to derive information is to display the data over a map background.

(continued)

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE 235 050 UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

20. ABSTRACT (concluded)

Although map data bases for the world have already been digitized, in order that a map background be effective in an operational environment, there must be a direct relationship between the amount and types of map feature data displayed, and the scale of the display. This report discusses the issues involved in creating custom data bases and provides documentation on the set of FORTRAN routines implemented for that purpose.

## ACKNOWLEDGMENTS

This report has been prepared by The MITRE Corporation under Project No. 7090. The contract is sponsored by the Electronic Systems Division, Air Force Systems Command, Hanscom Air Force Base, Massachusetts

Richard H. Bullen, Jr., D. Elliott Bell and W. Reid Gerhart contributed significantly towards the design and implementation of several parts of the geographic data base preprocessor. In addition, Dr. Bell provided guidance in writing this report.

TABLE OF CONTENTS

TABLE OF CONTENTS (Concluded)

## LIST OF ILLUSTRATIONS

## SECTION I

### INTRODUCTION

Project 7090, Operations/Intelligence Techniques Experimentation, has as its primary objective the development of man/machine interface, graphics display, and data processing techniques for effective applications of intelligence data in support of operational missions. The problems of processing, interpreting and applying intelligence data - data from multiple reporting systems whose output differ in content and format - are numerous. Further complications arise in trying to use this data to develop a unified consistent real-time picture; the separate systems usually report in different time spaces and on independent non-continuous attributes of the force-elements/forces involved. A significant portion of this available data, however, is reported in formatted messages, contains primarily positional information and has time and positional errors which are either tolerable or can easily be corrected.

Fundamental to any $C^2$ system handling positional geographic-based data, is the ability to display this data over a map background. Existing $C^2$ systems possess this capability to varying degrees; however, none of them has the total set of needed capabilities. Systems with an air situation display function normally offer fast response to operator requests but provide only minimal geographic background; alternatively ground-oriented systems provide the desired geographic data (sometimes too much) but suffer from slow response to "simple" change of context commands (e.g., translate to a new area of the map, zoom in on the current center point).

A basic map display system must not only provide for rapid display of operations/intelligence data in the context of major physical and political boundaries, but must also provide for the display of multiple classes of feature data - e.g., lines of communication



PRECEDING PAGE BLANK-NOT FILMED

(rivers, railroads, roads), bases, cities, terrain. In addition, it must automatically display this background information in a manner consistent with the scale of the area being displayed. When a large area is being pictured the amount of feature data and the degree of detail in the basic geography should be minimized - e.g., only major rivers and cities are shown, while small islands and inlets are ignored. If a small region is being shown, then all available boundary data as well as all cities, islands, and inlets in the region should be displayed. Finally, the map display system must provide rapid response to the "simple" change of context commands.

For FY76, therefore, a primary objective of Project 7090 has been the design of a map display facility which we call a Geographical Data Display Environment (GDDE). A major requirement for our system was a digitized geographical data base. *A wide variety of map features* were desirable in this data base, including topography, rivers, roads, railroads, cities, and military bases in addition to the usual geopolitical boundaries. However, only the geopolitical boundaries--coastlines and national boundaries--were available in digitized form at the outset. Thus our development of a geographical data base had two parts: the immediate utilization of the digitized coastlines and boundaries available in a data base called World Data Bank I (WDBI) and the development of a digitizing capability for the generation of the remaining map features.

The utilization of WDBI was constrained in several ways. First of all, special-purpose data structures and data manipulation techniques were likely for the GDDE, because of the goal of providing rapid response to "simple" change-of-context commands. This fact implied that the WDBI data format would not be useable and that we would have to reformat WDBI data in GDDE form. A second structure involves the amount of detail that must be left in the maps. Two factors influenced this situation: the raster-scan displays we used and the special emphasis in the GDDE development on the relation of scale and detail. This restriction requires very finely tuned tools for controlling the amount of detail present in a derivative map. The last constraint is that we must be able to provide special handling for various anomalies that arise, either from the data itself or from our manipulation of it. Altogether, we require a special set of tools that can modify WDBI to create custom map data bases for Europe, our initial area of interest, while retaining the ability to focus on other regions of the world or even to use different input data bases.

The set of tools we required was implemented as a package of FORTRAN programs. This report documents that package. Section II provides a general description of our process of generating maps for use in an interactive computer graphics environment. Section III is a user's guide for the package, and Section IV is a programmer's guide.

## SECTION II

## THE MAP GENERATION PROCESS

### INTRODUCTION

The process of generating maps for use in an interactive computer graphics environment has five components. These components and their inter-relationships are shown in Figure 1. A discussion of each component is given in this section.

### DATA CREATION

The data creation process consists of obtaining a set of geographic coordinates (latitude and longitude) for the region to be mapped. To date we have not had to digitize data ourselves. The data creation process consisted of making some initial modifications on an already existing data base, World Data Bank I.

The data contained in World Data Bank I (WDBI) was digitized in 1973 by the Federal Systems Division of IBM. There are over 100,000 points in the data base, and, as suggested by its name, the range of the points is the entire globe. The data is organized in three files: coastline and islands, boundaries and lakes, and an index to the first two files. The record format is given in Appendix A, Table I.

In this format World Data Bank I was unacceptable for use in our raster graphics lab. Two modifications were made to the data before the tape could be read in our lab. The first consisted of blocking the data at 6400 bytes instead of 80. This was essential to work with the data due to an eight tenths inch magnetic tape inter-record gap. The second change was an EBCDIC to ASCII conversion. Both of these steps were performed on an IBM 370/158.

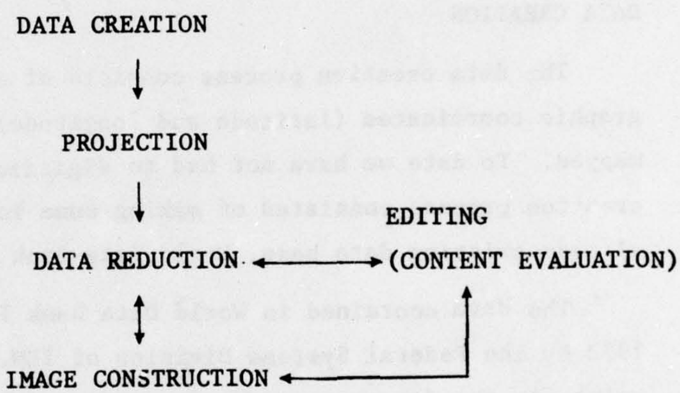



Figure 1. Map Generation Process

The data could then be handled in our minicomputer facility. The contents of the three files were printed so that an estimate of the number and location of points in the European region we wished to map could be made. On examining the index contained in the third file it was found to be inadequate. A thorough sort was not completed on this file, and the index itself is neither complete nor totally accurate. WDBI line segments are referenced by a map code of the area (continent, country, region) to which they belong. In many cases, however, the index would be needed to find the location of a line segment given the line segment number. In this respect the index was inefficient. To solve this problem the index was inverted and sorted by line segment numbers. (Its format is given in Appendix A, Table IV.)

For all subsequent work with WDBI it was necessary to keep only the latitude and longitude in radians for each data point. Reduced tapes for both coastline and boundary data were made. For each data point a 2-word line segment number and the latitude and longitude in radians were retained (see Appendix A, Table II). The data was converted from ASCII to binary and remained blocked at 6400 bytes, (i.e., 400 16-byte records per block). The input tapes for the projection component of the map generation package are the reduced coastline and boundary data files.

## PROJECTION

The projection process consists of transforming a set of geographic coordinates into Cartesian coordinates. Since this is a transformation from the globe to the plane, that is, from three-space to two-space, some stretching and/or shrinking must occur in the region which is being mapped. The stretching and shrinking can result in distortion of the shape, area, distance and direction measurements or any combination of these features in the mapped region. The type and extent of distortion is related to several characteristics of the region (e.g., its size, shape and location on the globe) as well as the type of projection which is used.

For very small regions, that is, regions extending over only a few degrees of latitude and longitude, the amount of stretching and shrinking is also small regardless of the projection which is used. It is possible in such cases to simply plot geographic coordinates. On the other hand for a region extending over many degrees of latitude and longitude, projection type has a definite impact on the resultant representation of the region.

The European region which is mapped for the project is of the latter type. It extends from 12° W longitude to 39° E and 30° N to 68° N latitudes. A secant conic projection, also called a conic with two standard parallels, was chosed for this region. The decision to use this projection was based on several considerations including the following:

- simple representation of parallels and meridians for ease in judging direction between points;
- no scale distortion along meridians so the region to be mapped has no longitudinal restriction;
- small percentage of stretching and/or shrinking along the parallels relative to other projections for the European region (a comparison of scale error among several projections is given in Appendix B);
- ability to control where minimum and maximum distortion occurs within the region through choice of standard parallels;
- simplicity of calculations (details on the mathematics of

the projection are given in the programmers guide under SUB-SET).

Figure 2 gives a breakdown of the projection process. There are actually three steps to be performed. The first is to make a first cut on the world data base to obtain a region slightly larger than that to be mapped. Points within this first cut are projected into the plane via a secant cone. The final step scissors the quadrangle so that it forms a rectangle. A scope photograph of the projected European data is shown in Figure 3. A geographic coordinate grid has been overlayed to aid in determining direction. Because a conic projection was used the meridians appear as radial lines from the apex of the cone and at their true angle, and the parallels are concentric arcs with the apex of the cone as their center.

Although the secant conic is the only projection in the package to date, the projection component can be modified to map data by any projection. Modifications would be necessary to only one subroutine.

DATA REDUCTION

The third component in the map generation process is data reduction. This step is essential for two reasons. The first is a limit on the amount of storage available for map data. The equipment in our graphics lab includes two minicomputers, an Interdata Model 4 and an Interdata Model 70. It is a result of the minicomputer environment and the nature of the application (i.e., map data is essential only as a background for military situation overlays) that this restriction is imposed.

The second reason for reducing the number of points in the data base is the limited resolution of the display medium. Figures 4 - 6

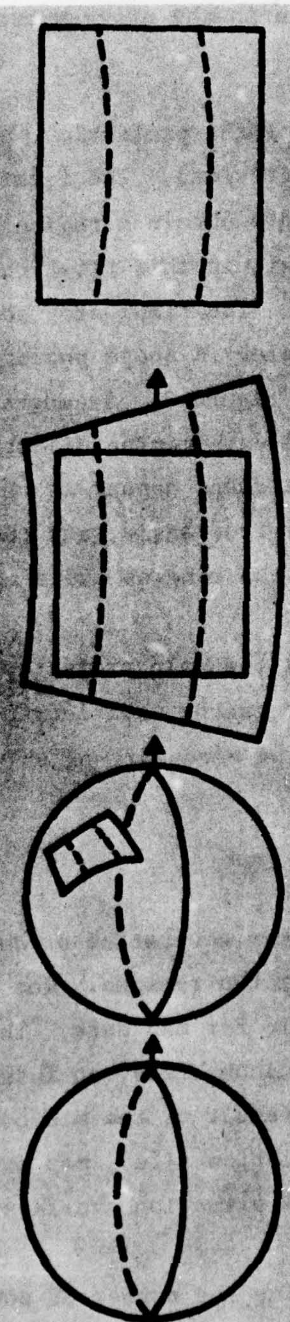


Figure 2. Projection Process

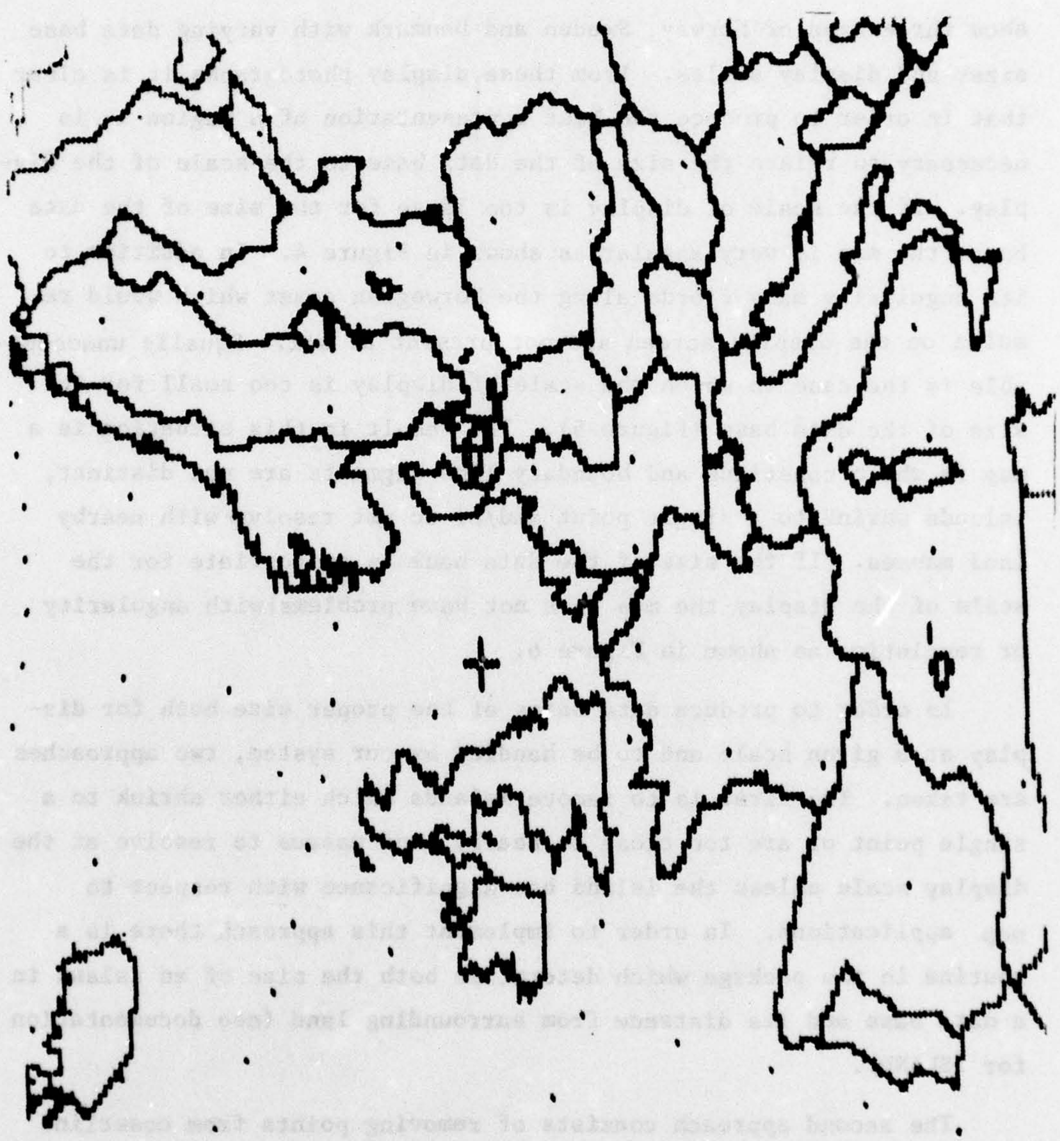


**Figure 3. Europe: Secant Conic Projection**

show three maps of Norway, Sweden and Denmark with varying data base sizes and display scales. From these display photographs it is clear that in order to produce the best representation of a region it is necessary to relate the size of the data base to the scale of the display. If the scale of display is too large for the size of the data base, the map is very angular as shown in Figure 4. In addition to its angularity many fjords along the Norwegian coast which would resolve on the display screen are not present at all. Equally unacceptable is the case in which the scale of display is too small for the size of the data base (Figure 5). The result in this situation is a map in which coastline and boundary line segments are not distinct, islands shrink to a single point and/or do not resolve with nearby land masses. If the size of the data base is appropriate for the scale of the display the map does not have problems with angularity or resolution as shown in Figure 6.

In order to produce data bases of the proper size both for display at a given scale and to be handled by our system, two approaches are taken. The first is to remove islands which either shrink to a single point or are too close to nearby land masses to resolve at the display scale unless the island has significance with respect to map applications. In order to implement this approach there is a routine in the package which determines both the size of an island in a data base and its distance from surrounding land (see documentation for ISLAND).

The second approach consists of removing points from coastline and boundary line segments. Points should be removed if they do not resolve at the display scale and consequently are unnecessarily being stored, or if they create bunching which results in a less than optimal representation of the region.

After the projection process the Cartesian coordinates of the

Figure 4. Map of Scandinavia With Too Little Data

Figure 5. Map of Scandinavia With Too Much Data

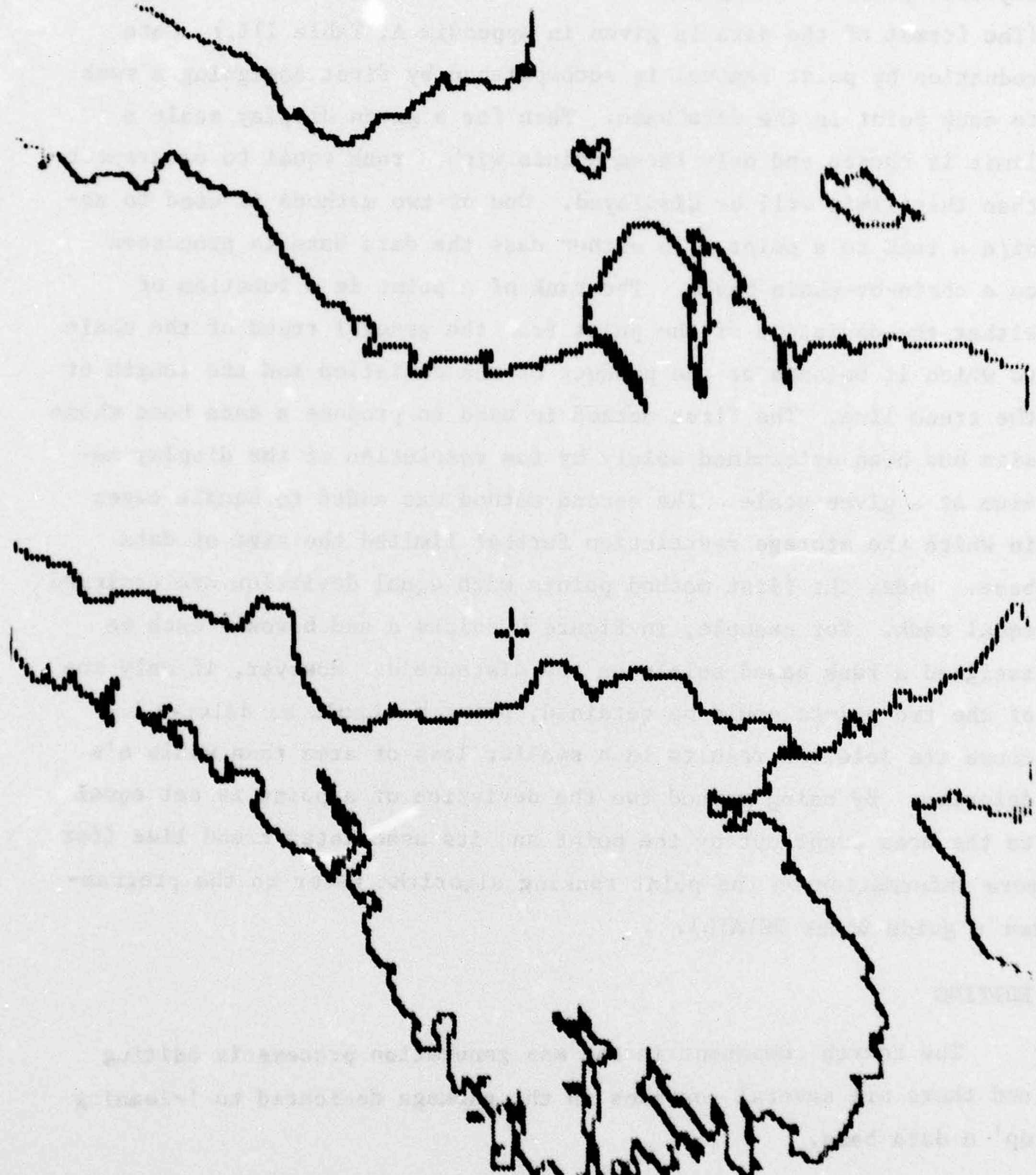


Figure 6. Map of Scandinavia With Appropriate Data

data points are stored in chains. A chain is defined as a group of adjacent points forming part of a coastline or boundary line segment. (The format of the data is given in Appendix A, Table III.) Data reduction by point removal is accomplished by first assigning a rank to each point in the data base. Then for a given display scale a limit is chosen and only those points with a rank equal to or greater than that limit will be displayed. One of two methods is used to assign a rank to a point.[2] In either case the data base is processed on a chain-by-chain basis. The rank of a point is a function of either the deviation of the point from the general trend of the chain to which it belongs or the product of the deviation and the length of the trend line. The first method is used to produce a data base whose size has been determined solely by the resolution of the display medium at a given scale. The second method was added to handle cases in which the storage restriction further limited the size of data base. Under the first method points with equal deviation are assigned equal rank. For example, in Figure 7 points a and b would each be assigned a rank based solely on the distance d. However, if only one of the two points could be retained, point b should be deleted because its deletion results in a smaller loss of area than would a's deletion. By using method two the deviation of a point is set equal to the area swept out by the point and its associated trend line (for more information on the point ranking algorithm refer to the programmer's guide under DETAIL).

EDITING

The fourth component in the map generation process is editing and there are several routines in the package dedicated to 'cleaning up' a data base.

Two types of errors were found in the raw WDBI data. As expected

[2] The method which was originally implemented for rank assignment is in use at the Harvard Lab for Computer Graphics and Spatial Analysis and is described in "POLYVRT User's Manual", Version 1.1.

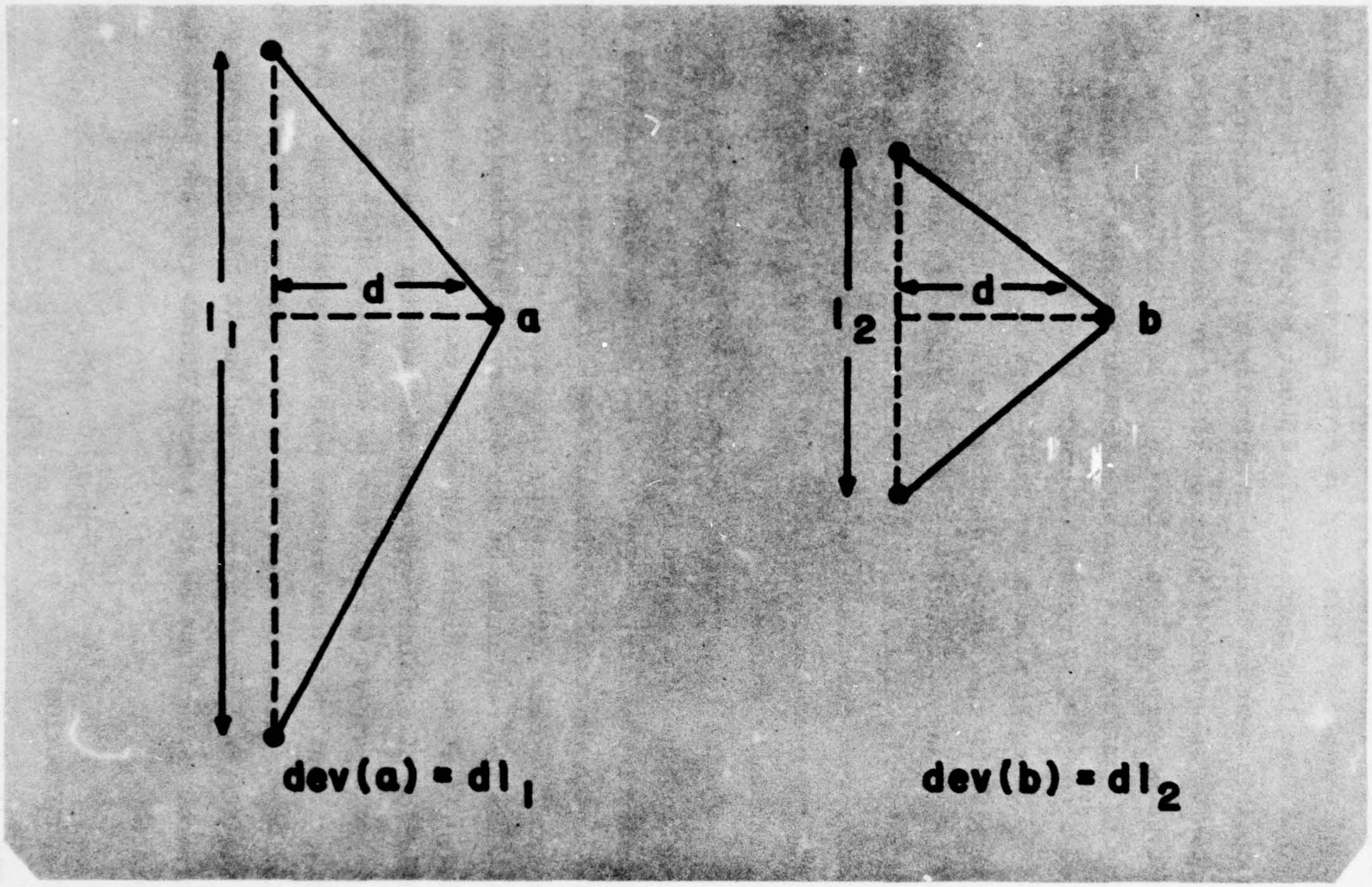


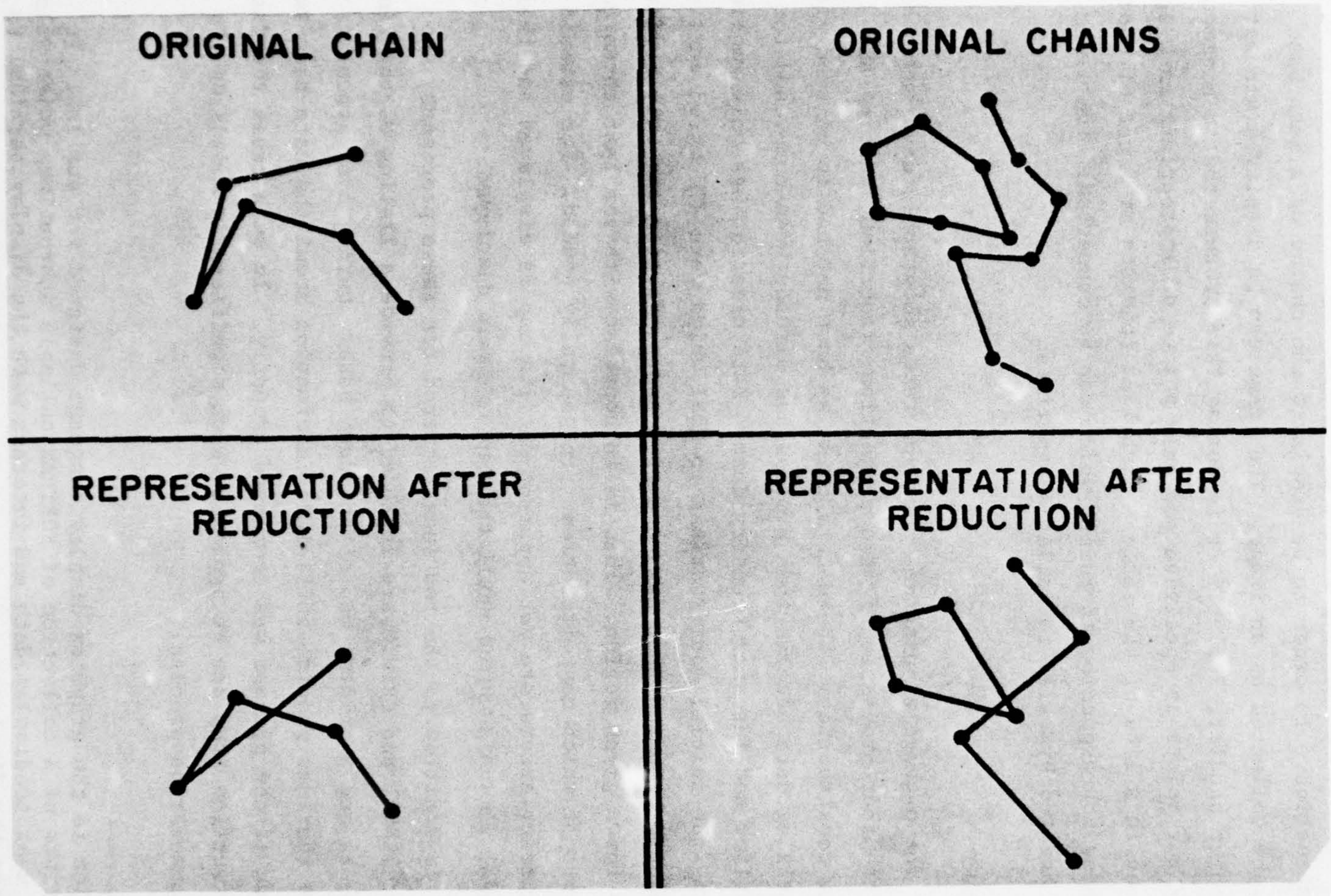

Figure 7. Product Metric

in creating a large geographic data base digitizing errors were made. A typical example of this type of error is unmatched coordinates of end points from line segments of adjacent countries. In such cases the endpoints must be changed to coincide either by altering the coordinates of an existing point or adding or deleting a point. The other type of error is the creation of artificial endpoints, that is, forming two or more line segments from an internal boundary or coastline of a country. In order to create a world data base with enough detail so as to be able to map small regions reasonably well, the maps from which the data is digitized must be detailed (i.e., large scale). For any extensive region, an example is our European region, the data must be digitized from several sheets. Some coastline and boundaries must inevitably be split. Because of our storage restriction and the overhead associated with storing a line segment, broken chains should be joined.

In addition to correcting errors in World Data Bank I data, clean-up routines are needed to produce data bases tailored for a particular display scale. For example, entire chains may need to be deleted which represent islands too small to appear. In data bases whose size has been determined by the amount of available storage rather than the resolution of the display certain display anomalies infrequently arise. Figure 8 shows two examples of the problem. In (a) deletion of a point results in the creation of a loop within the line segment, and in (b) deletion of an entire inlet results in the intersection of that line segment with a neighboring one. Due to the complexity of adding an algorithm to the point ranking routine to prevent this type of error occurring, and due to the infrequency of its occurrence, corrections on the rank of points are made during the editing process.

## IMAGE CONSTRUCTION

The final component in the map generation process is image




(a) Chain Crossover (b) Chain Intersection

Figure 8. Display Anomalies

construction. In order to be displayed the chains of a data base must be collected as an image. The image can be displayed via our graphics handler, Pallet.[3] It is during this process that a minimum rank for points is specified and only points of appropriate rank are placed in a line of the image. Each line consists of a set of coordinates corresponding to points which can be connected. The lines are created by calls to Pallet subroutines.

## SUMMARY

The package of programs that generates geographic data bases covers the four processes of data projection, reduction, editing and image construction. Up to this point we have not had to contend with digitizing data because this process has already been completed for coastline and internal boundary data. Data creation has only implied making the initial modifications to WDBI which were discussed earlier. It should be noted that data creation and projection are only performed once for a given region. This is in contrast with the last three components of which many iterations are generally needed. For example digitizing errors are not apparent until a map is displayed and their removal is not assured until the map is again displayed.

In addition to the routines required for these processes, two others, ENDPT and PRINTM, are included to provide a listing of the line segments and a printer map of the data base. During some processes these routines provide sufficient information about the data base to decide if the process was completed properly. In such cases the image construction step can be bypassed which significantly speeds up the map generation process.

---

[3]Pallet is the graphics display language designed for our lab. It consists of a collection of FORTRAN callable subroutines which allow the user to display data and interact with the display terminal to change the display.

## SECTION III

## USER'S MANUAL

### INTRODUCTION

This section provides the instructions for using programs in the map generation package. There are eleven routines in the package covering the areas of data projection, creation and reduction, image construction, data base content evaluation and editing. A breakdown of the routines in the package by these areas is given in Figure 9.

For each routine a brief description is given as well as the user inputs and program output.[4] Appendix A is referenced for the input/output data specifications. A sample run is also included.

---

[4] Logical unit to external device assignments are listed for each program for each device invoked by the program. For devices not used by the program it is assumed that the standard initial assignments have been made.

| **PROCESS** | **PROGRAM(S)** |
|---|---|
| **DATA CREATION** | **BOX** |
| **PROJECTION** | **SUBSET** |
| **REDUCTION** | **DETAIL**<br>**EXCLUD**<br>**ISLAND** |
| **IMAGE CONSTRUCTION** | **CMAP** |
| **CONTENT EVALUATION** | **ENDPT**<br>**PRINTM** |
| **EDITING** | **JOIN**<br>**EDITDB**<br>**MERGE** |

**Figure 9. Breakdown of Map Generation Routines**

DATA CREATION

BOX

Box creates a chain of points which provide a rectangular outline for the map. The coordinates of the corners of the rectangle may be specified by the user, or calculated in the program to be the minimum and maximum x- and y- coordinates of points in the data base.

Instructions

To execute BOX:

- make logical unit assignments

| | |
|---|---|
| AS 0195 | input tape on drive 95 |
| AS 0285 | output tape on drive 85 |
| AS 0313 | printer |
| AS 0510 | Carousel terminal |

- load BOX from loader

LG 2E00

LO 61 BOX

END

- start execution

ST 5000

Input

Data: projected map data tape, format as listed in Appendix A, Table III.

User: In addition to specifying the logical units for input and output tapes, the user must enter the coordinates of the lower left and upper right hand corners of the rectangular outline. If the user would like these coordinates to be calculated by BOX as the minimum and maximum x- and y- coordinates of points in the data base, corresponding coordinates of the two points should be the same. A rank within the range specified in DETAIL for the points created by BOX must also be entered. All data is entered from the Carousel.

**Output**

**Data:** formatted the same as the input tape: the final chain on the tape is the box chain and is identified as such by a three in the third field of the chain ID.

**Printer:** at the conclusion of the run the total number of chains which have been processed and the minimum and maximum x- and y- coordinates of points in the data base are printed.

**Sample Run**

```
ST 5000
 ENTER 2 BOX COORDINATES 2F5.3

-.398-.263
0.2780.410
ENTER INPUT DEVICE NUMBER(NN)
01
ENTER OUTPUT DEVICE NUMBER(NN)
02
ENTER DEVICE LEVEL FOR BOX
11




EOJ
```

DATA PROJECTION

SUBSET

Subset combines three processes: projection of data points from the globe to the plane, subsetting the data base, retaining only those points within a user-specified area and reformatting the chains which are retained for ease in handling in subsequent routines. In order to decrease processing time the data is first subjected to a geographic coordinate test. Only those points inside a specified geographic range are projected and the coordinates thus determined are checked against a Cartesian coordinate range. Those points inside this range are retained. [5]

Instructions

To execute SUBSET:

- make logical unit assignments

| | |
|---|---|
| AS 0695 | input tape on drive 95 |
| AS 0485 | output tape on drive 85 |
| AS 0313 | printer |
| AS 0104 | card reader |

- load SUBSET from loader

LG 2E00

LO 61 SUBSET

END

- start execution

ST 5000

Input

Data: WDBI data, format as given in Appendix A, Table I

User: The user must specify the following which is read from four cards:

[5] The planar region describes a rectangle and is referred to as such in the sample run.

- a single digit integer identifying the type of data:[6]
  - 1 = coastline
  - 2 = boundary
- two three-digit integers specifying the offset into the the file for the region to be mapped, and the number of blocks to be processed.
- four floating point numbers, format F6.2, which specify the latitude (North to South) and longitude (West to East) limits for the initial test
- four floating point numbers, format F8.5, which specify the planar limits; min, max x-coordinates, followed by min, max y-coordinates

Output

Data: The output data from SUBSET is formatted as specified in Appendix A, Table III. For each chain of points there is a header for identification, followed by the Cartesian coordinates of the elements of the chain, followed by an end of chain marker.

Printer: The input data which was read from cards is printed for verification. As each line segment is processed diagnostics are printed which include:

- the number of crossings of the geographic coordinage range and where these crossings occur;
- if a line segment leaves the Cartesian coordinate range, where it leaves and if it re-enters, where it re-enters;

[6] The type of data may be coastline, boundary box, other. For appropriate integer code see Appendix A, Table III.

- the number of generated chains from a line segment. (A chain consists of a series of points inside the planar region. Thus, from a single line segment in WDBI several chains could be generated if the line segment leaves and re-enters the region.)

At the conclusion of the run the following summary statistics are printed:

- total number of line segments inside the specified region
- number of line segments processed and totally outside the region
- total number of data points inside the region
- number of generated chains

<u>Sample Run</u>

Card Reader Input

```
2
ØØØØ75
Ø71.ØØØ28.ØØ-25.ØØØ4Ø.ØØ
-Ø.398ØØØØ.278ØØ-Ø.263ØØØØ.41ØØØ
```

Printer Output

```
NUMBER OF BLOCKS SKIPPED =  0
NUMBER OF BLOCKS TO BE READ = 76
LAT RANGE  71.00 TO  28.00  LONG RANGE -25.00  TO 40.00
X RANGE IS -0.39800 TO  0.27800 Y RANGE IS -0.26300
 TO 0.41000
```

```
LINE SEGMENT NUMBER =3001300

LEAVING RECTANGLE AT  0.16468     0.41000 FROM  0.16476     0.41056

CROSSING LAT-LONG BOUNDARY AT         0.12404957E 01      0.48147053E 00

CROSSING LAT-LONG BOUNDARY AT         0.12385607E 01      0.49263448E 00

ENTERING RECTANGLE AT 0.16871     0.41000 FROM  0.16867     0.40991

LEAVING RECTANGLE AT  0.17470     0.41000 FROM  0.17453     0.41050

ENTERING RECTANGLE AT 0.19354     0.41000 FROM  0.19334     0.40974

  TOTAL NUMBER OF CROSSINGS OF LAT-LONG BOUNDS =  2
  NUMBER OF GENERATED CHAINS =  3
  NUMBER OF POINTS INSIDE RECTANGLE =540
  LAT RANGE IS  63.010 TO    71.075
  LONG RANGE IS       7.548 TO     31.016


  LINE SEGMENT NUMBER =3001340


  TOTAL NUMBER OF CROSSINGS OF LAT-LONG BOUNDS =  0
  NUMBER OF GENERATED CHAINS =  1
  NUMBER OF POINTS INSIDE RECTANGLE =112
  LAT RANGE IS  63.000 TO    65.937
  LONG RANGE IS    18.234 TO     24.068


 FINAL SUMMARY
 TOTAL LS INSIDE AT ALL =  55

 TOTAL LS PROCESSED BUT TOTALLY OUTSIDE RECT. = 238

 TOTAL NUMBER OF INSIDE PTS = 1579

 TOTAL NUMBER OF GENERATED CHAINS =  55
```

## DATA REDUCTION

### DETAIL

In order to reduce the size of the map data base while retaining an adequate number of points for an accurate representation of the area, points in the subsetted data base are assigned a rank. Then only those points possessing a rank greater than or equal to a specified value are displayed.

DETAIL measures the importance of a point in delineating a feature and assigns a rank based on that measure. The importance of a point is measured by its deviation from the general trend of the chain to which it belongs. The calculation of the deviation of a point from its associated trend line is the first task performed by DETAIL. Once the deviation of a point is known a rank is assigned by checking this value against a series of user specified bandwidths which define equivalence classes. A point becomes a member of that class whose bandwidth is the largest exceeded by the deviation of the point.

<u>Instructions</u>

To execute DETAIL:

- make logical unit assignments

| | |
|---|---|
| AS 0195 | input tape on drive 95 |
| AS 0685 | output tape on drive 85 |
| AS 0313 | printer |
| AS 0510 | Carousel terminal |

- load DETAIL from loader

  LG 2E00

  LO 61 SUBSET

  END

- start execution

ST 5000

Input

Data: a projected data tape, format as given in Appendix A, Table III

User: from the Carousel the user is asked to enter the following:

- a single digit integer specifying whether or not the user wants diagnostics printed for each data point:

  1 = yes

  Ø = no

- a single digit integer specifying the metric to be used for computing the deviation of points from their associated trend lines;

  1 = distance of the point from the trend line

  2 = product of distance of the point from the trend line and the trend line length.[7]

- a two digit integer specifying the number of bandwidths
- a single digit integer indicating the bandwidth specification method;

Let:

| | |
|---|---|
| bandwidth (i) | = BW(I) |
| bandwidth factor(i) | = BWF(I) |
| band multiplier | = BM |
| minimum Cartesian coordinate range | = MCR |
| bandwidth increment | = INC |
| minimum bandwidth | = MINBW |

Bandwidths are computed by one of three methods:

(1) geometrical

$BWF(1) = MINBW$

$BWF(1) = BWF(I-1) \cdot BM$

[7] Originally only metric 1 was implemented. The 'produce metric' was added as an effort towards a more even point distribution. For a description of the two metrics, see Section II, Data Reduction.

$BW(I) = BWF(I) \cdot MCR$

(2) linear

$BWF(1) = MINBW$

$BWF(1) = BWF(I-1) + INC$

$BW(I) = BWF(I) \cdot MCR$

(3) arbitrary

BWF(I) is specified by the user for all I,

$BWF(I) > BW(I-1)$

$BW(I) = BWF(I) \cdot MCR$

- for geometric bandwidths

  a floating point number, format E14.7,specifying minimum bandwidth, MINBW

  a floating point number, format F5.3, specifying a band multiplier, BM

  a floating point number, format E14.7, specifying the minimum Cartesian coordinate range of the map, MCR

- for linear bandwidths

  a floating point number, format E14.7, specifying minimum bandwidth, MINBW

  a floating point number, format E14.7, specifying bandwidth increment, INC

  a floating point number, format E14.7, specifying the minimum Cartesian coordinate range of the map, MCR

- for arbitrary bandwidths

  N floating point numbers, each having format E14.7 specifying bandwidth factors (N = number of bandwidths previously specified by the user), BWF (I), I = 1,N

<u>Output</u>

**Data:** The third and fourth fields of each data point contain the deviation of the point from its trend line and its rank respectively.

Printer: Diagnostics are printed for each chain which has been processed including:

- the original WDBI line segment number corresponding to the chain
- integer code for type of chain (coastline, boundary, etc.)
- number of points in the chain
- point distribution by bandwidth

Carousel: A summary is given for each bandwidth/equivalence class including:

- the bandwidth factor and bandwidth
- the number of points assigned to that class
- the number of points assigned to that class and those of higher priority
- the greatest number of points in a single chain assigned to that class and those of higher priority

In addition the total number of chains, total number of points, the number of points in the longest chain and the number of points in shortest chain are printed.

Sample Run

Input:

```
ST 5000
DETAIL DEFINER
ENTER 1 FOR DEBUG, ELSE 0
0
ENTER CODE FOR METRIC
ENTER 1 FOR DEVIATION FROM TREND LINE
ENTER 2 FOR PRODUCT OF DEVIATION AND TREND LINE LENGTH
1
ENTER NO. OF DETAIL LEVELS (NN, UP TO 20)
10
ENTER BANDWIDTH SPEC METHOD
ENTER 1 FOR GEOMETRIC
ENTER 2 FOR LINEAR
ENTER 3 FOR ARBITRARY
```

```
1
ENTER MIN BANDWIDTH (E14.7  )
       0.0010000
ENTER BAND MULTIPLIER (N.NNN,OVER 1.0)
1.500
ENTER MIN COORD RANGE (E14.8)
       0.68000000
```

**Output: Point Distribution**

| Seq. # | WDBIL.S.N. | type | Chain # | Cum. Chain # | #Pts. | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CHN# 66 | 3002250 | 1 | 27 | 82 | N= 12 | 0 | 8 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 2 |
| CHN# 67 | 3002280 | 1 | 29 | 84 | N= 16 | 0 | 11 | 2 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 2 |
| CHN# 68 | 3002310 | 1 | 31 | 86 | N= 37 | 0 | 24 | 2 | 5 | 0 | 3 | 0 | 0 | 0 | 1 | 2 |
| CHN# 69 | 3002380 | 1 | 32 | 87 | N= 41 | 0 | 28 | 3 | 1 | 4 | 1 | 1 | 0 | 0 | 1 | 2 |
| CHN# 70 | 3002390 | 1 | 33 | 88 | N= 17 | 0 | 13 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 2 |
| CHN# 71 | 3002520 | 1 | 36 | 91 | N=325 | 1 | 231 | 18 | 13 | 18 | 15 | 9 | 8 | 3 | 1 | 8 |
| CHN# 72 | 3002530 | 1 | 37 | 92 | N=205 | 0 | 128 | 21 | 10 | 16 | 7 | 7 | 3 | 3 | 4 | 6 |
| CHN# 73 | 3002540 | 1 | 38 | 93 | N= 16 | 0 | 11 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 2 |
| CHN# 74 | 3002560 | 1 | 39 | 94 | N= 6 | 0 | 2 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 2 |
| CHN# 75 | 3002570 | 1 | 40 | 95 | N= 12 | 0 | 7 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 2 |
| CHN# 76 | 3002580 | 1 | 41 | 96 | N= 14 | 0 | 8 | 1 | 0 | 2 | 0 | 1 | 0 | 0 | 0 | 2 |
| CHN# 77 | 3002670 | 1 | 46 | 101 | N=197 | 0 | 125 | 17 | 15 | 9 | 9 | 6 | 7 | 1 | 2 | 6 |
| CHN# 78 | 3002680 | 1 | 47 | 102 | N= 42 | 0 | 22 | 6 | 0 | 5 | 3 | 1 | 1 | 1 | 1 | 2 |
| CHN# 79 | 3002710 | 1 | 48 | 103 | N= 9 | 0 | 5 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 2 |
| CHN# 80 | 3002790 | 1 | 53 | 108 | N= 7 | 0 | 4 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 2 |

**Summarization**

| DL | BWF | BW | NP | CNP | CLC | CSC |
|---|---|---|---|---|---|---|
| 0 | 0.00000 | 0.00000 | 2 | 7550 | 476 | 4 |
| 1 | 0.00100 | 0.00068 | 4819 | 7548 | 476 | 4 |
| 2 | 0.00150 | 0.00102 | 625 | 2729 | 201 | 2 |
| 3 | 0.00225 | 0.00153 | 515 | 2104 | 142 | 2 |
| 4 | 0.00338 | 0.00230 | 412 | 1589 | 104 | 2 |
| 5 | 0.00506 | 0.00344 | 319 | 1177 | 64 | 2 |
| 6 | 0.00759 | 0.00516 | 208 | 858 | 42 | 2 |
| 7 | 0.01139 | 0.00775 | 166 | 650 | 28 | 2 |
| 8 | 0.01709 | 0.01162 | 80 | 484 | 18 | 2 |
| 9 | 0.02563 | 0.01743 | 54 | 404 | 12 | 2 |
| 10 | 0.03844 | 0.02614 | 350 | 350 | 9 | 2 |

```
   137 CHAINS PROCESSED
  7550 POINTS PROCESSED
   476 POINTS IN LONGEST CHAIN
     4 POINTS IN SHORTEST CHAIN
DONE
```

EXCLUD

EXCLUD deletes entire chains of points from the data base.

Instructions

To execute EXCLUD

- make logical unit assignments

| | |
|---|---|
| AS 0195 | input tape on drive 95 |
| AS 0685 | output tape on drive 85 |
| AS 0510 | Carousel terminal |

- load EXCLUD from loader

  LG 2E00

  LO 61 EXCLUD

  END

- start execution

  ST 8000

Input

Data: projected data base, format as given in Appendix A, Table III

User: The user must supply a list of chain numbers corresponding to the chains to be deleted. The chains must be listed in order of their appearance on the input tape.

Output

Data: specific chains have been removed from the data base.

Sample Run

```
ST 8000
ENTER INPUT DEVICE NUMBER(NN)
01
ENTER OUTPUT DEVICE NUMBER(NN)
06
```

```
ENTER CHAINS TO BE ELIMINATED(NNN),
ENTER ZERO AS LAST CHAIN
057
082
161
185
003
000
```

ISLAND

This routine is an aid in determining those islands which are too small to be displayed at a given scale. An island is deleted if it appears as a single point or if it merges with another land mass when displayed. ISLAND computes the horizontal and vertical extent of all chains in the data base whose endpoints coincide.[8] The measurements are made in terms of the Cartesian coordinates of the data. To test the proximity of an island to nearby landmasses, a second pass through the data is required. The coordinates of each point in the vicinity of an island are compared to the minimum and maximum x- and y- coordinates of the island calculated in the first pass through the data. If the distance between these points is less than a minimum value,[9] the coordinates of the points and their respective chain numbers are printed.

Instructions

To execute ISLAND

- make logical unit assignments

  | | |
  |---|---|
  | AS 0685 | input tape on drive 85 |
  | AS 0313 | printer |

- load ISLAND from loader

  LG 2E00

  LO 61 ISLAND

  END

- start execution

  ST 5000

[8]Some editing is necessary before running ISLAND as not all islands in WDBI have coincident endpoints. ISLAND will not perform any computations for such chains.

[9]This value was chosen in relation to the resolution of the display medium. With 256 x 240 addressable positions the minimum horizontal distance was 4/256 and the minimum vertical distance was 4/240.

Input

Data: projected data, format as given in Appendix A, Table III

Output

Printer: From the first pass, for each chain the coordinates of the first and last points as well as the WDBI line segment number and chain number are listed. For each island the minimum and maximum x- and y- coordinates are listed. From the second pass the coordinates and chain numbers of points less than the minimum allowable distance apart are printed.

Sample Run

First Pass

```
WDBLSN      Chain type  Chain #    Cum. Chain #        x-start    y-start    x-end      y-end

5001700        1          116          171             0.2780     0.3746     0.2780     0.3361
5005510        1          117          172             0.1607     0.1120     0.2279     0.2427
5005520        1          118          173             0.1855     0.1934     0.1855     0.1934  *
    MIN X IS   0.1720 AT Y =   0.1883
    MAX X IS   0.1890 AT Y =   0.1916
    MIN Y IS   0.1793 AT X =   0.1767
    MAX Y IS   0.1936 AT X =   0.1854
    DELTA X =    0.0162 DELTA Y =    0.0144

5005530        1          119          174             0.1765     0.1971     0.1765     0.1971  *
    MIN X IS   0.1731 AT Y =   0.1954
    MAX X IS   0.1839 AT Y =   0.1976
    MIN Y IS   0.1934 AT X =   0.1783
    MAX Y IS   0.2004 AT X =   0.1783
    DELTA X =    0.0100 DELTA Y =    0.0070

5005570        1          120          175             0.2557     0.2745     0.2780     0.2783
5005570        1          121          176             0.2780     0.2566     0.2554     0.2753
  0   0        3            1          177            -0.4000    -0.2700    -0.4000    -0.2700  *
    MIN X IS  -0.4000 AT Y =  -0.2700
    MAX X IS   0.2800 AT Y =  -0.2700
    MIN Y IS  -0.2700 AT X =  -0.4000
    MAX Y IS   0.4100 AT X =   0.2800
    DELTA X =    0.6800 DELTA Y =    0.6800
```

## Second Pass

| Island Chain# | x | y | other land mass chain# | x | y | Delta x | Delta y |
|---|---|---|---|---|---|---|---|
| 63 | 0.0820 | 0.3501 | 1 | 0.0912 | 0.3413 | 0.0092 | |
| 63 | 0.0820 | 0.3501 | 1 | 0.0893 | 0.3406 | 0.0072 | |
| 63 | 0.0820 | 0.3501 | 1 | 0.0880 | 0.3394 | 0.0059 | |
| 145 | 0.2246 | -0.1429 | 42 | 0.2241 | -0.1378 | 0.0005 | |
| 145 | 0.2163 | -0.1385 | 42 | 0.2241 | -0.1378 | 0.0077 | |
| 145 | 0.2185 | -0.1374 | 42 | 0.2241 | -0.1378 | | 0.0004 |
| 145 | 0.2227 | -0.1435 | 42 | 0.2241 | -0.1378 | | 0.0056 |
| 77 | 0.0175 | 0.2568 | 58 | 0.0106 | 0.2497 | | 0.0071 |
| 77 | 0.0120 | 0.2528 | 58 | 0.0106 | 0.2497 | | 0.0030 |
| 77 | 0.0175 | 0.2568 | 58 | 0.0112 | 0.2506 | | 0.0062 |
| 77 | 0.0120 | 0.2528 | 58 | 0.0112 | 0.2506 | | 0.0022 |
| 77 | 0.0175 | 0.2568 | 58 | 0.0114 | 0.2513 | | 0.0055 |
| 77 | 0.0120 | 0.2528 | 58 | 0.0114 | 0.2513 | | 0.0015 |
| 77 | 0.0175 | 0.2568 | 58 | 0.0121 | 0.2520 | | 0.0048 |
| 77 | 0.0120 | 0.2528 | 58 | 0.0121 | 0.2520 | | 0.0008 |
| 77 | 0.0175 | 0.2568 | 58 | 0.0132 | 0.2524 | | 0.0045 |
| 77 | 0.0120 | 0.2528 | 58 | 0.0132 | 0.2524 | | 0.0004 |
| 77 | 0.0175 | 0.2568 | 58 | 0.0142 | 0.2527 | | 0.0041 |
| 77 | 0.0120 | 0.2528 | 58 | 0.0142 | 0.2527 | | 0.0000 |
| 77 | 0.0175 | 0.2568 | 58 | 0.0148 | 0.2527 | | 0.0041 |
| 77 | 0.0120 | 0.2528 | 58 | 0.0148 | 0.2527 | | 0.0000 |

IMAGE CONSTRUCTION

CMAP

CMAP creates images of the data points via subroutine calls to Pallet and stores these images in a drum file. The contents of this file is properly formatted for display.

Instructions

To execute CMAP:

- make logical unit assignments

| | |
|---|---|
| AS 0195 | mount tape WRG016 |
| AS 0685 | input tape on drive 85 |
| AS 0510 | Carousel terminal |

- load COREDP

  RW DE

  BI DC00

  LO DE

- load WRG016 from tape drive 95; enter these commands from Carousel (COREDP responses not given)

  ST DC00

  LO

  01

  0080, FFFE

- load CMAP

  RW 60

  LO 60

- start execution

  ST 2E00

Input

Data: projected data,format as given in Appendix A, Table III

User: the user will be asked to supply an eight character name for the image and a two digit integer specifying the minimum rank (i.e., bandwidth supplied by DETAIL) which points must have in order to be retained in the map.

Output

Data: drum file containing an image of the data points of rank greater than or equal to that specified by the user.

Sample Run

```
ST 2E00
 ENTER NAME
MAPTHREE
 ENTER DETAIL LEVEL(NN)
11
STOP
 EOJ
```

## CONTENT EVALUATION

### ENDPT

ENDPT provides a list of the chains in the map data base and the x- and y- coordinates of the first and last points in each chain.

<u>Instructions</u>

To execute ENDPT:

make logical unit assignments

| | |
|---|---|
| AS 0685 | input tape on device 85 |
| AS 0313 | printer |

load ENDPT from loader

LG 2E00

LO 61 ENDPT

END

start execution

ST 5000

<u>Input</u>

Data: projected data base, format as given in Appendix A, Table III

<u>Output</u>

Printer: For each chain the coordinates of the first and last points are listed as well as the WDBI line segment number and an integer indicating the source of the chain (i.e., coastline, boundary)

<u>Sample Run</u>

| OLD ID | SOURCE | ORIGIN SEQ. | TOTAL SEQ. | XSTART | YSTART | XEND | YEND |
|---|---|---|---|---|---|---|---|
| 3001370 | 2 | 5 | 5 | 0.1091 | 0.3062 | 0.2279 | 0.2427 |
| 3004000 | 2 | 14 | 14 | 0.0205 | 0.1034 | 0.0150 | 0.1033 |
| 3004100 | 2 | 16 | 16 | 0.0654 | 0.0261 | 0.0950 | 0.2012 |
| 3004180 | 2 | 18 | 18 | 0.0008 | -0.0239 | 0.0306 | -0.0256 |
| 3004200 | 2 | 20 | 20 | -0.0263 | 0.0110 | -0.0106 | 0.0100 |
| 3004240 | 2 | 24 | 24 | -0.0211 | 0.0200 | -0.0213 | 0.0310 |
| 3004420 | 2 | 29 | 29 | 0.2217 | 0.0314 | 0.2200 | 0.0181 |
| 3004460 | 2 | 33 | 33 | 0.1343 | -0.0266 | 0.2015 | -0.0271 |
| 3004490 | 2 | 36 | 36 | 0.2326 | 0.0135 | 0.2700 | 0.0206 |
| 3004510 | 2 | 37 | 37 | 0.2700 | -0.0490 | 0.2494 | -0.0505 |
| 3004550 | 2 | 39 | 39 | 0.2700 | -0.0920 | 0.2659 | -0.0965 |
| 3004920 | 2 | 43 | 43 | 0.2353 | -0.1137 | 0.2027 | -0.1930 |

PRINTM

PRINTM prints a crude map from the points in the data base. Due to the low resolution of the printer the map produced by PRINTM is of limited use. It does provide a check on the extension of regions mapped with the data, and most major problems with data base (e.g., missing chains) will be made obvious.

Instructions

To execute PRINTM:

- make logical unit assignments

  AS 0685 input tapes on device 85

  AS 0313 printer

  AS 0510 Carousel terminal

- load PRINTM from loader

  LG 2E00

  LO 61 PRINTM

  END

- start execution

  ST 5000

Input

Data: projected data, format as given in Appendix A, Table III

User: The user is asked for a two digit integer which specifies the minimum rank of points to be printed.

Output

Printer: The output from PRINTM is a 60 x 100 matrix map of the data base. The horizontal and vertical extensions of the map are reflected in the rows and columns of the matrix. An element of the matrix contains an X if the coordinates of any point in the data base are within the range of that particular cell. Otherwise the element contains a blank.

Carousel:At the conclusion of the run a PAUSE is encountered. The user may change input tapes and continue the run by typing CO, and produce the map with an overlay at the same level of detail. For example, if coastline and boundary data were on separate tapes, PRINTM could produce a map of the coastlines first and then coastlines with a boundary overlay. The coastlines would be identified by the character 'X', and boundaries by the character '@'.

Sample Run

```
ST 5000
ENTER DETAIL LEVEL(NN)
04
PAUSE  99
 PAUSE
RW 6
ST 5000
ENTER DETAIL LEVEL(NN)
02
PAUSE  99
 PAUSE
```

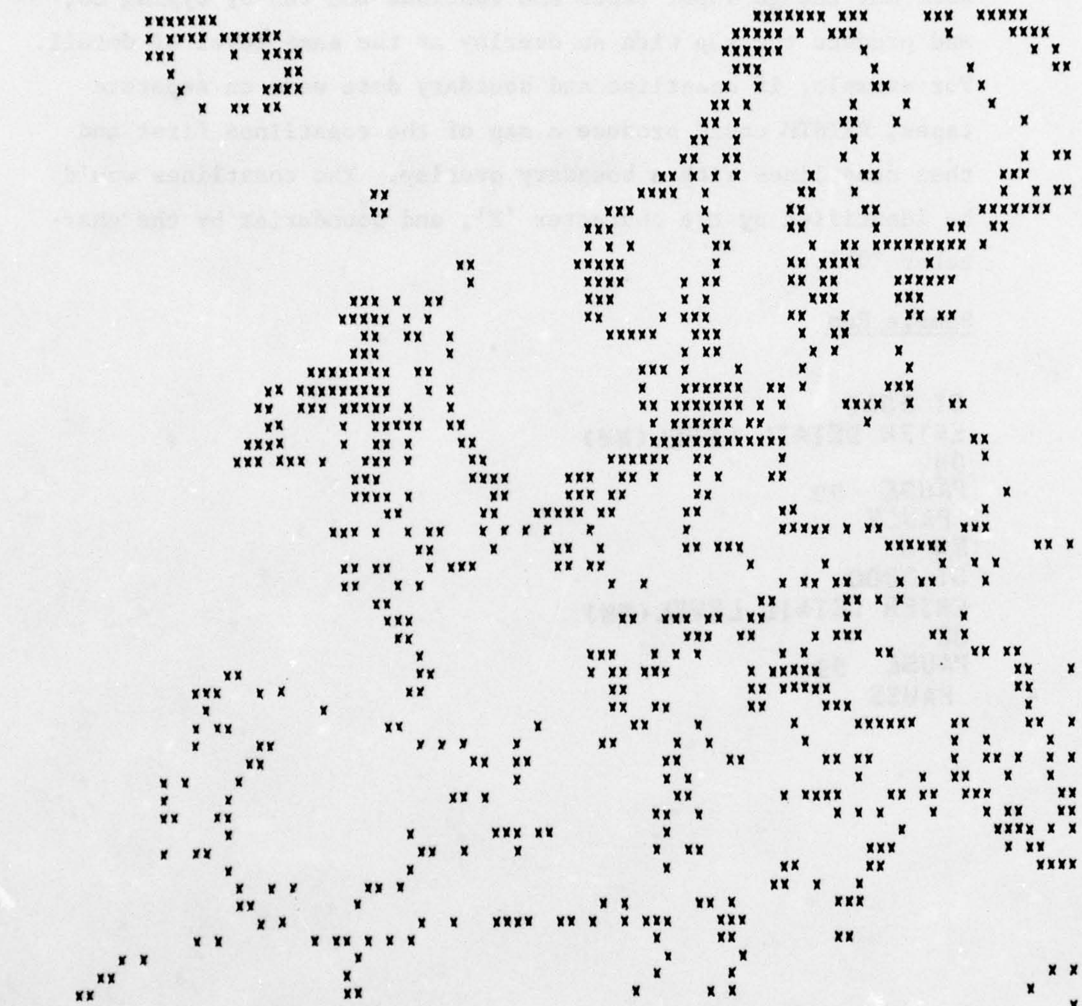


Figure 10. Sample Output: Detail Level 4

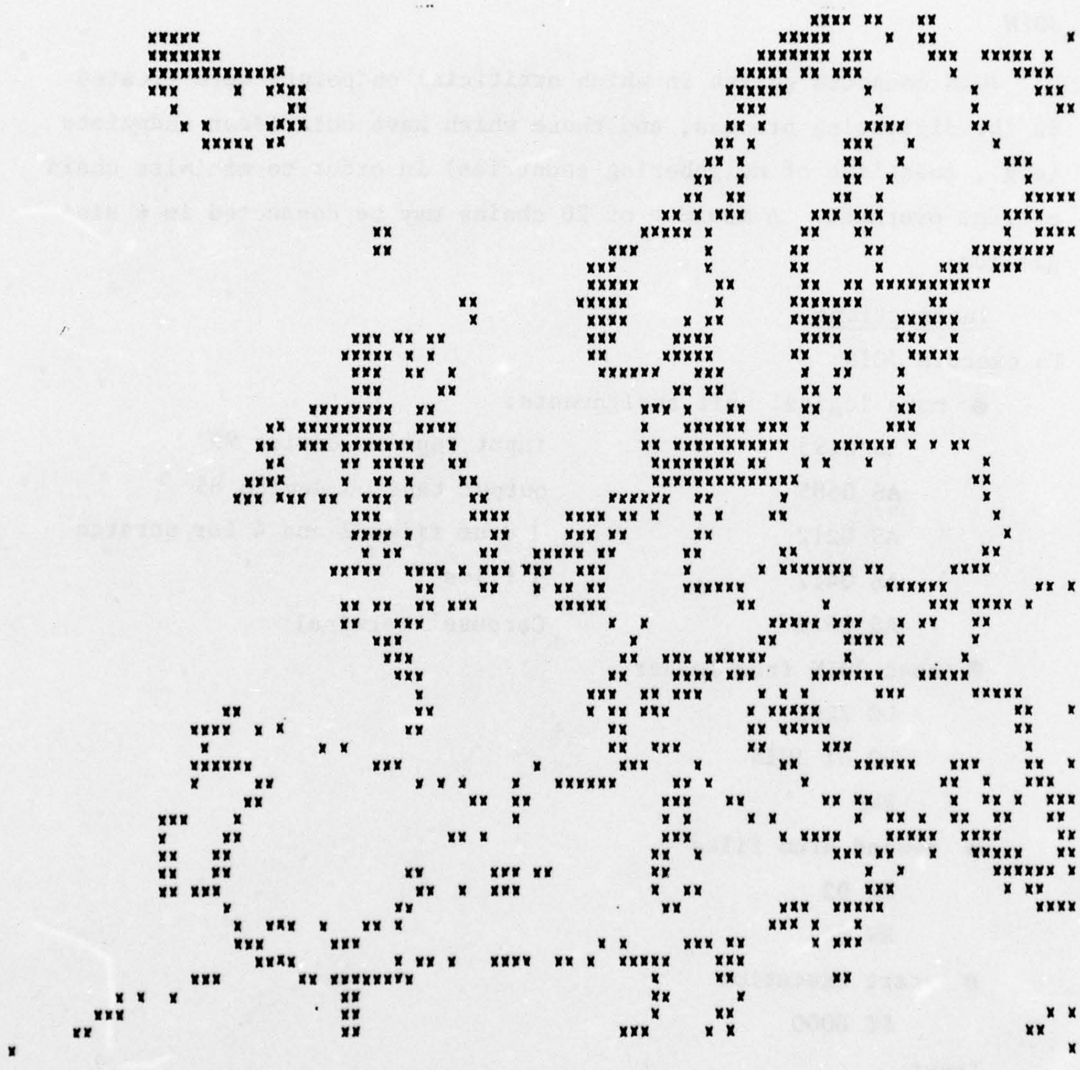


**Figure 11. Sample Output: Detail Level 2**

EDITING

JOIN

JOIN connects chains in which artificial endpoints were created in the digitizing process, and those which have coincident endpoints (e.g., coastline of neighboring countries) in order to minimize chain storage overhead. A maximum of 20 chains may be connected in a single run.

Instructions

To execute JOIN:

- make logical unit assignments:

| | |
|---|---|
| AS 0195 | input tape on device 95 |
| AS 0685 | output tape on device 85 |
| AS 0212 | drum files 2 and 4 for scratch files |
| AS 0412 | |
| AS 0510 | Carousel terminal |

- load JOIN from loader

  LG 2E00

  LO 61 JOIN

  END

- rewind drum files

  RW 02

  RW 04

- start execution

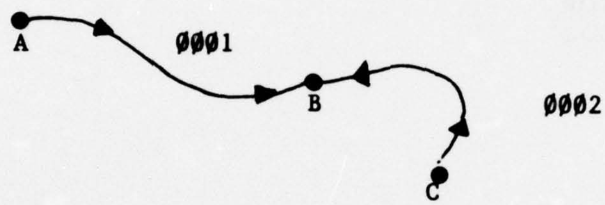
  ST 8000

Input

Data: projected data, format as given in Appendix A, Table III

User: The user must specify the input, output and scratch files. If several runs are to be made two drum files and one mag tape may be used instead of two mag tapes and one drum file to speed up the process. In addition the user will input a series of four digit integers specifying the chains to be linked. If the points of a chain are stored in reverse order to that needed for linkage, input a minus sign as the first digit of the chain number.

e.g.,




To join line segment ØØØ1 (A → B) and ØØØ2 (C → B), input:

ØØØ1

-ØØ2

ØØØØ

Output

Data: User-specified chains have been linked. The new linked chain assumes the chain number of the first chain in the series. It is placed at the end of the file. The third field in the chain ID of a joined chain will contain a four (see Appendix A, Table III). The first two fields of the chain ID, which hold the WDBI line segment number of the chain, contain zeroes because a chain produced by JOIN may be composed of points from several WDBI line segments.

Sample Run

```
ST 8000
ENTER INPUT DEVICE NUMBER(NN)
04
ENTER OUTPUT DEVICE NUMBER(NN)
06
ENTER SCRATCH DEVICE NUMBER(NN)
02
```

```
ENTER CHAINS TO BE LINKED(NNNN)
NEGATIVE NUMBER FOR OPPOSITE ORDER
ENTER ZERO AS END OF CHAINS TO BE LINKED
NULL CHAIN TO EXIT
  11
 -23
 161
0000
STOP
 EOJ
```

**EDITDB**

EDITDB functions as a point editor. With it the coordinates and/or rank of a point can be changed. Points can be added to or deleted from a chain.

Instructions

To execute EDITDB:

make logical unit assignments

| | |
|---|---|
| AS 0195 | input tape on device 95 |
| AS 0685 | output tape on device 85 |
| AS 0412 | scratch file is drum file 4 |
| AS 0510 | Carousel terminal |

load EDITDB from loader

LG 2E00

LO 61 EDITDB

END

rewind scratch file

RW 4

start execution

ST 5000

Input

Data: projected data, format as specified in Appendix A, Table III

User: The user must specify the input, output and scratch files and a series of four digit integers specifying chains to which edited points belong. Editing is performed on a chain by chain basis. Any number of points may be edited in a single chain. For each point in a chain the user specifies

- the position of the point in its chain (Note: Position of the point must reflect previous deletions and additions to the chain. If a point is deleted, positions of following points are decreased by one; simularly if a point is added, positions of following points are increased by one;

- an integer code for the type of editing to be performed;
- the new geographic coordinates and/or rank as appropriate.

**Output**

**Data:** User-specified points are edited and the chains to which they belong are written following the last chain in the data base.

**Sample Run**

```
ST 5000
ENTER INPUT DEVICE NUMBER(NN)
06
ENTER OUTPUT DEVICE NUMBER(NN)
01
ENTER SCRATCH DEVICE NUMBER(NN)
04
ENTER CHAINS TO BE EDITED(NNNN)
ENTER ZERO AS END OF CHAINS TO BE EDITED
0002
0031
0000
 EDIT CODES :
 1-CHANGE COORDINATES
 2-CHANGE DETAIL LEVEL
 3-BOTH
 4-ADD POINT
 5-DELETE POINT
 CHAIN NUMBER =     2
 ENTER POSITION OF POINT TO BE EDITED(NNN)
150
 POSITION NUMBER = 150 ENTER TASK NUMBER (N)
2
 ENTER DETAIL LEVEL (NN)
01
 FURTHER EDITING IN SAME CHAIN?
 ENTER 1=YES, 0=NO
1
 ENTER POSITION OF POINT TO BE EDITED(NNN)
151
 POSITION NUMBER = 151 ENTER TASK NUMBER (N)
2
 ENTER DETAIL LEVEL (NN)
01
```

```
 FURTHER EDITING IN SAME CHAIN?
 ENTER 1=YES, 0=NO
0
 CHAIN NUMBER =  31
 ENTER POSITION OF POINT TO BE EDITED(NNN)
674
 POSITION NUMBER = 674 ENTER TASK NUMBER (N)
5
 FURTHER EDITING IN SAME CHAIN?
 ENTER 1=YES, 0=NO
0
STOP
 EOJ
```

MERGE

MERGE combines two data tapes which were created independently by the projection routine, SUBSET, (e.g., coastline and boundary data).

Instructions

To execute MERGE:

- make logical unit assignments

| | |
|---|---|
| AS 0195 | output tape on drive 95 |
| AS 0685 | input tape on drive 85 |
| AS 0510 | Carousel terminal |

- load MERGE from loader

```
LG 2E00
LO 61 MERGE
END
```

- start execution

```
ST 5000
```

Input

Data: projected data, format as specified in Appendix A, Table III

User: The user specifies the logical units for input and output. The output tape is initially blank. The input tapes are copied to the output tape one after the other; successive input tapes are mounted on tape drive 85, and the logical unit number is re-entered. The user can merge any number of tapes using this procedure.

Output

Data: The merged tape has the same format as the individual input tapes (see Appendix A, Table III). However, the sixth field of the chain ID contains the cumulative chain number. For example, if the boundary tape was copied after the coastline tape, the new chain number of the first boundary chain would be 1 + (the number of coastline chains). On the input boundary tape this field for the first chain would contain a one.

Sample Run

```
ST 5000
ENTER INPUT DEVICE NUMBER(NN)
06
ENTER OUTPUT DEVICE NUMBER(NN)
01
ENTER INPUT DEVICE NUMBER(NN) - IF FINISHED ENTER 00
00
TOTAL NUMBER OF CHAINS  172

 EOJ
```

## SECTION IV

## PROGRAMMER'S GUIDE

### INTRODUCTION

Additional documentation is provided here on each of the routines for which operating instructions are given in Section III. The function of each routine and a description of the procedure implemented to perform that function are stated. This is followed by a list of the common blocks used and subroutines called by the program. A high-level flow chart, program listing and load module map complete the documentation. Similarly for each subroutine called by the main program, the function, procedure description, common block and subroutine list and program listing are given.

BOX

Purpose

BOX creates a chain of points which form a rectangular outline for a map.

Procedure Description

The routine copies all of the data to the output tape until the end-of-data mark is encountered.[10] The data is read into core and written to tape in 6400-byte blocks. In core it is examined four words at a time for the end of the data. The variable IHEAD identifies the section of a chain being examined(i.e., header, set of coordinates or end of chain - see Appendix A, Table III). When the end of the data has been reached the five point box chain is placed in the buffer and written to tape. A record is kept of the minimum and maximum x- any y- coordinates while the data points are being copied to the output file so that these may be used as the box coordinates should the user so specify.

Common Blocks

None

Subroutines

None

---

[10] The end of the data is signalled by two end-of-chain marks. An end-of-chain is a four word entry consisting of a floating point 1000.0 followed by three blanks.

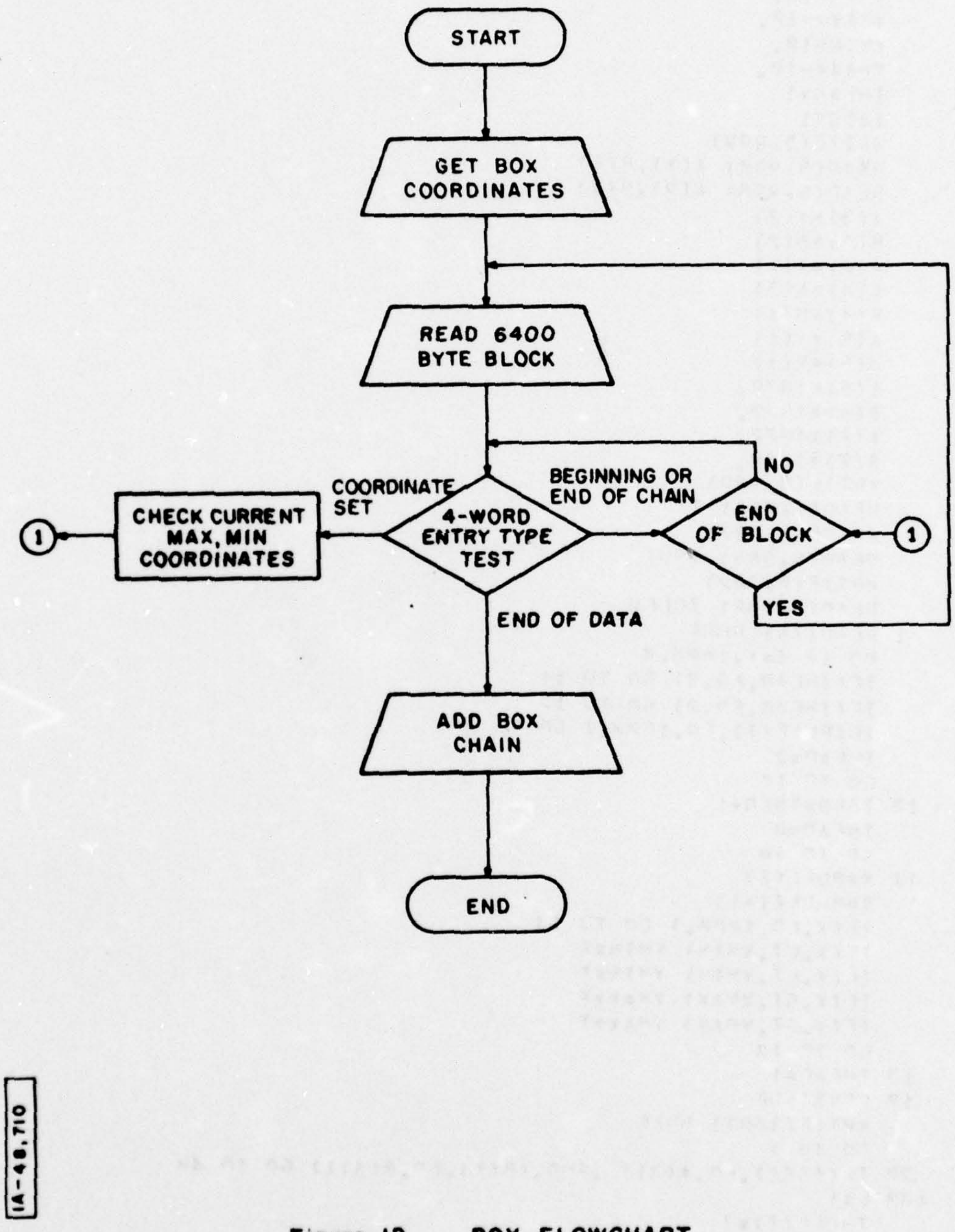



**Figure 12 BOX FLOWCHART**

```
      DIMENSION BUFF(1600),IBUFF(1600)
      DIMENSION A(7),B(7)
      EQUIVALENCE (IBUFF(1),BUFF(1))
      XMIN=10.
      XMAX=-10.
      YMIN=10.
      YMAX=-10.
      IHEAD=1
      ISEQ=1
      WRITE(5,999)
      READ(5,998) A(1),B(1)
      READ(5,998) A(2),B(2)
      A(3)=A(2)
      B(3)=B(2)
      A(2)=A(1)
      A(4)=A(3)
      B(4)=B(1)
      A(5)=A(1)
      B(5)=B(1)
      A(6)=1000.
      B(6)=1000.
      A(7)=1000.
      B(7)=1000.
      WRITE(5,989)
      READ(5,988) IN
      WRITE(5,987)
      READ(5,988) IOUT
      WRITE(5,982)
      READ(5,988) IDLEV
    1 READ(IN) BUFF
      DO 10 I=1,1600,4
      IF(IHEAD.EQ.0) GO TO 11
      IF(IHEAD.EQ.2) GO TO 12
      IF(BUFF(I).EQ.1000.) GO TO 20
      IHEAD=2
      GO TO 10
   12 ISEQ=ISEQ+1
      IHEAD=0
      GO TO 10
   11 X=BUFF(I)
      Y=BUFF(I+1)
      IF(X.EQ.1000.) GO TO 13
      IF(X.LT.XMIN) XMIN=X
      IF(Y.LT.YMIN) YMIN=Y
      IF(X.GT.XMAX) XMAX=X
      IF(Y.GT.YMAX) YMAX=Y
      GO TO 10
   13 IHEAD=1
   10 CONTINUE
      WRITE(IOUT) BUFF
      GO TO 1
   20 IF((A(1).EQ.A(3)) .AND.(B(1).EQ.B(3))) GO TO 40
  100 L=1
      IBUFF(I)=?
```

```
      IBUFF(I+1)=0
      IBUFF(I+2)=3
      IBUFF(I+3)=0
      I=I+4
      IF(I.LT.1598) GO TO 102
      WRITE(IOUT) BUFF
      I=1
  102 IBUFF(I)=1
      IBUFF(I+1)=ISEQ
      I=I+4
      IF(I.GT.1598) GO TO 103
  101 DO 110 J=I,1600,4
      BUFF(J)=A(L)
      BUFF(J+1)=B(L)
      BUFF(J+2)=0.0
      IBUFF(J+3)=IDLEV
      L=L+1
      IF(L.GT.7) GO TO 200
  110 CONTINUE
  103 WRITE(IOUT) BUFF
      I=1
      GO TO 101
  200 WRITE(IOUT) BUFF
      WRITE(3,901) ISEQ,XMIN,XMAX,YMIN,YMAX
      STOP
   40 A(1)=XMIN
      A(2)=XMAX
      A(3)=XMAX
      A(4)=XMIN
      A(5)=XMIN
      B(1)=YMIN
      B(2)=YMIN
      B(3)=YMAX
      B(4)=YMAX
      B(5)=YMIN
      GO TO 100
  901 FORMAT(3X,17HNUMBER OF CHAINS=,I4,/,
     1  3X,11HX MINIMUM =,F8.4,/,
     1  3X,11HX MAXIMUM =,F8.4,/,
     1  3X,11HY MINIMUM =,F8.4,/,
     1  3X,11HY MAXIMUM =,F8.4)
  982 FORMAT(26HENTER DEVICE LEVEL FOR BOX)
  987 FORMAT(30HENTER OUTPUT DEVICE NUMBER(NN))
  988 FORMAT(I2)
  989 FORMAT(29HENTER INPUT DEVICE NUMBER(NN))
  999 FORMAT(30H ENTER 2 BOX COORDINATES 2F5.3/)
  997 FORMAT(53H IF COORDINATES ARE EQUAL THEN MIN & MAX WILL BE USED)
  998 FORMAT(2F5.3)
      END
 I  .U       0000
 R  BUFF     070E
 I  IBUFF    070E
 R  A        200E
 R  B        20BA
```

```
E XMIN   20D6
F XMAX   20DE
E YMIN   20EA
E YMAX   20EE
E IHEAD  20F2
F ISEQ   20FA
A 999    0728
I #I     0000
A 998    078F
A 989    0702
A 988    06FA
I IN     2102
A 987    06D2
I IOUT   2106
A 982    06AF
I IDLEV  210A
I !      01FB
I #J     0000
A 10     0324
I I      210E
A 11     0280
A 12     0268
A 20     0354
E X      211A
E Y      211E
A 13     031C
A 40     05AE
A 100    0384
I L      2122
A 102    042E
A 103    052C
A 101    047A
A 110    051A
I J      2132
A 200    0552
A 901    062A
I .S     0000
A 997    0750
I .V     0000


          PROGRAMS:
  713E #I        721F .U        724C .V        725A #I
  8286 .S        82BA .MES      8338


ENTRY-POINTS:
  713E #I        722A .U        7250 .V        725A #I
  8286 .S        82BA .MES


COMMON-BLOCKS:
NONE


UNDEFINED:
NONE
```

SUBSET

Purpose

SUBSET performs three steps in the map generation process; subsetting, projecting and reformatting WDBI data.

Procedure Description

Each data point in WDBI is first subjected to a gross test on its latitude and longitude. The geographic (lat., long.) coordinates for this test are input by the user. Only those points inside the specified range are projected. Once a point has been projected, its Cartesian coordinates are tested to determine whether or not the point falls inside the rectangle. If it does, its coordinates are written on tape. If a rectangular boundary is crossed in drawing a line from the point previously examined to the point currently being processed, the coordinates of the intersection of this line and the boundary are determined and written on tape. Whenever a line segment extends beyond the Cartesian coordinate range of the display the segment is truncated and an end-of-chain is written on tape. The end-of-chain marker is also written after the last point of a line segment inside the Cartesian coordinate range has been processed.

Common Blocks

| Block Name | Contents | Description of Contents |
|---|---|---|
| /BLK1/ | XYPTS (1600) | output buffer |
| /BLK2/ | WLAT, BLAT, WLONG, ELONG | geographic coordinate range |
| /BLK3/ | XMIN, XMAX, YMIN, YMAX | Cartesian coordinate range |
| /BLK4/ | USTDP, RUSTDP, CTRLME, RGLOBE, PI, PI1, CONST. | constants for projection |

/BLK5/ IRITE flag signalling that an end-of-chain has just been written in output buffer

Subroutines

PROJEC: Projection of point from the globe into the plane via a secant cone.

CALL PROJEC (RLAT, RLONG, X, Y)

RLAT: latitude of point in radians

RLONG: longitude of point in radians

X: projected x-coordinate, returned by routine

Y: projected y-coordinate, returned by routine

TEST1: tests the geographic coordinates of a point.

CALL TEST1 (PREVPT, PRESPT, RLAT, RLONG)

PREVPT: flag signalling location of last point processed with respect to geographic coordinate range, returned by routine

PRESPT: flag signalling location of point currently being processed with respect to geographic coordinate range, returned by routine

RLAT: latitude of point in radians

RLONG: longitude of point in radians

TEST2: tests x and y coordinates of projected point.

CALL TEST2 (OLDFL, NEWFL, X, Y, BOUNDX, BOUNDY)

OLDFL: flag signalling location of last point processed with respect to Cartesian coordinate range, returned by routine

NEWFL: flag signalling location of point currently being processed with respect to Cartesian coordinate range, returned by routine

X: x-coordinate

Y: y-coordinate

BOUNDX: x-coordinate of boundary point if one has been generated by the routine

BOUNDY: y-coordinate of boundary point if one has been generated by the routine

SAVELS: writes chain identification header in output buffer.

CALL SAVELS (LSN1, LSN2, SOURCE, NSUM, POINT)

LSN1: first three digits of WDB1 line segment number

LSN2: last four digits of WDB1 line segment number

SOURCE: integer specifying the type of data in the chain[8]

NSUM: new chain number

POINT: pointer in XYPTS, the output buffer

SAVEXY: writes x- and y- coordinates of a point in output buffer.

CALL SAVEXY (X, Y, POINT) or CALL SAVEXY (BOUNDX, BOUNDY, POINT)

X: x-coordinate of a projected WDB1 data point

Y: y-coordinate of a projected WDB1 data point

BOUNDX: x-coordinate of a generated boundary point

BOUNDY: y-coordinate of a generated boundary point

POINT: pointer in XYPTS, the output buffer

SAVEND: writes end-of-chain in output buffer

CALL SAVEND (DUMMYX, POINT)

DUMMYX: end-of-chain mark = 1000.0

POINT: pointer in XYPTS, the output buffer

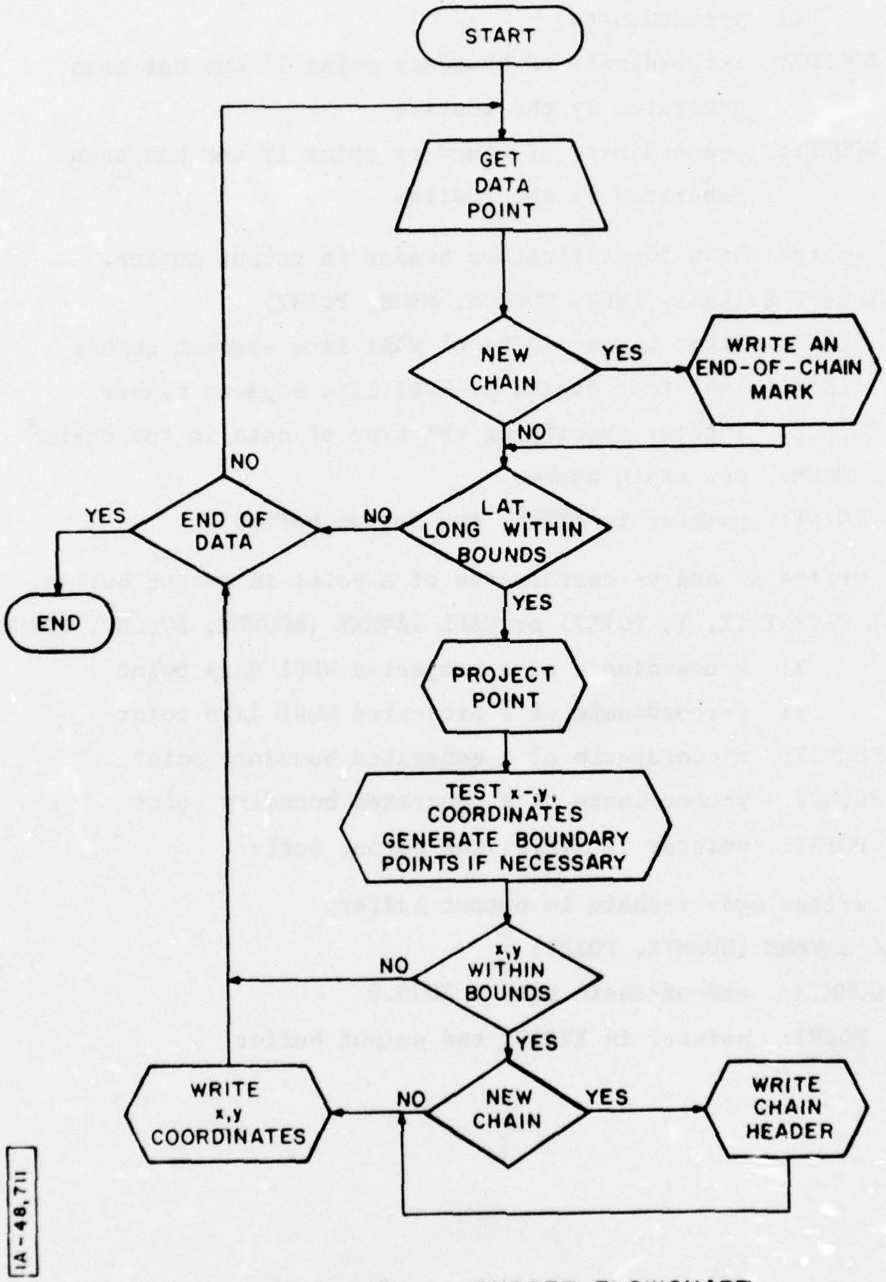



Figure 13 SUBSET FLOWCHART

```
      DIMENSION BLOCK(1600),IBLOCK(1600),IXYPTS(1600)
      COMMON/BLK1/XYPTS(1600) /BLK2/ULAT,BLAT,WLONG,ELONG
      COMMON /BLK3/XMIN,XMAX,YMIN,YMAX
      COMMON /BLK4/USTDP,RUSTDP,CTRLME,RGLOBE,PI,PI1,CONST
      COMMON /BLK5/IRITE
      EQUIVALENCE (BLOCK(1),IBLOCK(1))
      EQUIVALENCE (XYPTS(1),IXYPTS(1))
      INTEGER CHAIN,GRTST,NEWFL,OLDFL,POINT,PRESPT,PREVPT
      INTEGER TOTCHA,TOTSUM,SUMIN
      INTEGER SOURCE
      REAL LARGLA,LARGLO
      IRITE=0
      LSN1=0
      LSN2=0
      BOUNDX=0.
      BOUNDY=0.
      LSIN=0
      LSOUT=0
      TOTSUM=0
      TOTCHA=0
      DUMMYX=1000.
      NSUM=0
      POINT=1
      PI=3.14159
      PI1=PI/180.
      USTDP=57.00
      BSTDP=41.00
      RGLOBE=1.
      CTRLME=7.47
      CTRLME=CTRLME*PI1
      RUSTDP=2.*PI*RGLOBE*((USTDP-BSTDP)/360.)*((COS(USTDP*PI1))/
     C(COS(BSTDP*PI1)-COS(USTDP*PI1)))
      CNTRLP=(USTDP+BSTDP)/2.
      CONST=RUSTDP+2.*PI*RGLOBE*(USTDP-CNTRLP)/360.
C
C READ FROM CARDS - # OF BLOCKS TO BE SKIPPED, # OF BLOCKS TO BE READ,
CLAT-LONG RANGE (DEGREES), X-Y RANGE
C
      READ(1,580) SOURCE
      READ(1,400) N,M
      READ(1,410) ULAT,BLAT,WLONG,ELONG
      READ (1,420) XMIN,XMAX,YMIN,YMAX
      WRITE (3,530)N
      WRITE(3,535) M
      WRITE (3,540) ULAT,BLAT,WLONG,ELONG
      WRITE (3,550) XMIN,XMAX,YMIN,YMAX
      ULAT=ULAT*PI1
      BLAT=BLAT*PI1
      WLONG=WLONG*PI1
      ELONG=ELONG*PI1
```

```
C
C SKIP N BLOCKS
C
      IF(N.EQ.0) GO TO 26
      DO 25 I=1,N
25    READ (6) BLOCK
26    CONTINUE
      DO 250 M1=1,M
      READ(6) BLOCK
      DO 230 I=1,1597,4
      K=I+1
      J=I+2
      L=I+3
      RLAT=BLOCK(J)
      RLONG=BLOCK(L)
      IF(LSN1.NE.IBLOCK(I)) GO TO 75
      IF(LSN2.EQ.IBLOCK(K)) GO TO 125
C
C PROCEDURE FOR FIRST POINT IN LINE SEGMENT
C
75    IF(LSN1.EQ.0) GO TO 100
C
C GIVE SEGMENT SUMMARY FOR PREVIOUS LINE SEGMENT
C
      IF(INBIT.EQ.0) LSOUT=LSOUT+1
      IF(INBIT.EQ.1) LSIN=LSIN+1
      TOTSUM=TOTSUM+SUMIN
      IF(INBITC.EQ.0) CHAIN=CHAIN-1
      TOTCHA=TOTCHA+CHAIN
      WRITE(3,430) GRTST
      WRITE (3,440) CHAIN
      WRITE (3,450) SUMIN
      SMALLA=SMALLA/PI1
      LARGLA=LARGLA/PI1
      SMALLO=SMALLO/PI1
      LARGLO=LARGLO/PI1
      WRITE(3,560) SMALLA,LARGLA
      WRITE (3,570) SMALLO,LARGLO
      IF((IRITE.EQ.0).AND.(NSUM.GT.0)) CALL SAVEND(DUMMYX,POINT)
100   LSN1=IBLOCK(I)
      LSN2=IBLOCK(K)
      SMALLA=BLOCK(J)
      LARGLA=BLOCK(J)
      SMALLO=BLOCK(L)
      LARGLO=BLOCK(L)
      INBITC=0
      INBIT=0
      GRTST=0
      CHAIN=1
      SUMIN=0
      PRESPT=2
      PREVPT=2
      NEWFL=2
      OLDFL=2
```

```
      WRITE(3,460) LSN1,LSN2
      CALL TEST1(PREVPT,PRESPT,RLAT,RLONG)
      CALL PROJEC  (RLAT,RLONG,X,Y)
      CALL TEST2 (OLDFL,NEWFL,X,Y,BOUNDX,BOUNDY)
      IF(NEWFL.EQ.0) GO TO 225
      INBIT=1
      INBITC=1
      NSUM=NSUM+1
      CALL SAVELS(LSN1,LSN2,SOURCE,NSUM,POINT)
      CALL SAVEXY(X,Y,POINT)
      SUMIN=SUMIN+1
      GO TO 225
C
C PROCEDURE FOR REMAINING POINTS OF LINE SEGMENT
C
125   CALL TEST1 (PREVPT,PRESPT,RLAT,RLONG)
      IF (PRESPT.EQ.PREVPT) GO TO 150
      GRTST=GRTST+1
      WRITE(3,485) RLAT,RLONG
150   IF(PRESPT.EQ.0) GO TO 225
      CALL PROJEC (RLAT,RLONG,X,Y)
      CALL TEST2(OLDFL,NEWFL,X,Y,BOUNDX,BOUNDY)
      IF(NEWFL.EQ.0) GO TO 200
      IF (OLDFL.EQ.1) GO TO 175
C
C PRESENT POINT IN, PREVIOUS POINT OUT (ENTERING)
C
      NSUM=NSUM+1
      CALL SAVELS(LSN1,LSN2,SOURCE,NSUM,POINT)
      CALLSAVEXY (BOUNDX,BOUNDY,POINT)
      SUMIN=SUMIN+1
      WRITE(3,470) BOUNDX,BOUNDY,X,Y
C
C* PRESENT POINT IN
C
175   CALL SAVEXY (X,Y,POINT)
      INBIT=1
      INBITC=1
      SUMIN=SUMIN+1
      GO TO 225
C
C PRESENT POINT OUT
C
200   IF(OLDFL.EQ.0) GO TO 225
C
C PRESENT POINT OUT, PREVIOUS POINT IN (LEAVING)
C
      CALL SAVEXY(BOUNDX,BOUNDY,POINT)
      SUMIN=SUMIN+1
      CALL SAVEND(DUMMYX,POINT)
      CHAIN=CHAIN+1
      INBITC=0
      WRITE (3,480) BOUNDX,BOUNDY,X,Y
225   LARGLA=AMAX1(LARGLA,BLOCK(J))
```

```
      SMALLA=AMIN1(SMALLA,BLOCK(J))
      LARGLO=AMAX1(LARGLO,BLOCK(L))
      SMALLO=AMIN1(SMALLO,BLOCK(L))
C
C END OF TESTING FOR THAT POINT
C
240   CONTINUE
250   CONTINUE
C
CEND OF TESTING - ALL POINTS
CGIVE FINAL SEGMENT SUMMARY
C
      CALL SAVEND(DUMMYX,POINT)
      CALL SAVEND(DUMMYX,POINT)
      WRITE(4) XYPTS
      IF (INBIT.EQ.0) LSOUT=LSOUT+1
      IF (INBIT.EQ.1) LSIN=LSIN 1
      TOTSUM=TOTSUM+SUMIN
      IF(INBITC.EQ.0) CHAIN=CHAIN-1
      TOTCHA=TOTCHA+CHAIN
      WRITE(3,430) GRTST
      WRITE (3,440) CHAIN
      WRITE(3,450) SUMIN
C
CWRITE FINAL SUMMARY
C
      WRITE (3,490) LSIN
      WRITE(3,500) LSOUT
      WRITE(3,510) TOTSUM
      WRITE(3,520) TOTCHA
400   FORMAT (2I3)
410   FORMAT(4F6.2)
420   FORMAT(4F8.5)
430   FORMAT(//47H TOTAL NUMBER OF CROSSINGS OF LAT-LONG BOUNDS =,I3)
440   FORMAT(29H NUMBER OF GENERATED CHAINS =,I3)
450   FORMAT(36H NUMBER OF POINTS INSIDE RECTANGLE =,I3)
460   FORMAT(///22H LINE SEGMENT NUMBER =,I3,I4)
470   FORMAT(/22H ENTERING RECTANGLE AT,F8.5,3X,F8.5,6H FROM ,F8.5,3X,
     CF8.5)
480   FORMAT(/22H LEAVING RECTANGLE AT ,F8.5,3X,F8.5,6H FROM ,F8.5,3X,
     CF8.5)
485   FORMAT(/31H CROSSING LAT-LONG BOUNDARY AT ,2E20.8)
490   FORMAT(///14H FINAL SUMMARY,/25H TOTAL LS INSIDE AT ALL = ,I4)
500   FORMAT(/47H TOTAL LS PROCESSED BUT TOTALLY OUTSIDE RECT. =,I4)
510   FORMAT(/29H TOTAL NUMBER OF INSIDE PTS =,I5)
520   FORMAT(/35H TOTAL NUMBER OF GENERATED CHAINS =,I4)
530   FORMAT(27H NUMBER OF BLOCKS SKIPPED =,I3)
535   FORMAT(/30H NUMBER OF BLOCKS TO BE READ =,I3)
540   FORMAT(/11H LAT RANGE ,F6.2,4H TO ,F6.2,13H  LONG RANGE ,F6.2,4H
     CTO ,F6.2)
550   FORMAT(/12H X RANGE IS ,F8.5,4H TO ,F8.5,12H Y RANGE IS ,F8.5,4H
     CTO ,F8.5)
560   FORMAT(13H LAT RANGE IS,F8.3,4H TO ,F8.3)
570   FORMAT(14H LONG RANGE IS,F9.3,4H TO ,F9.3)
```

```
580     FORMAT(I1)
        END
I BLK1    1900
E XYPTS   0000
E IXYPTS  0000
I BLK2    0010
E ULAT    0000
E BLAT    0004
E WLONG   0008
E ELONG   000C
I BLK3    0010
E XMIN    0000
E XMAX    0004
E YMIN    0008
E YMAX    000C
I BLK4    001C
E USTDP   0000
E RUSTDP  0004
E CTRLME  0008
E RGLOBE  000C
E PI      0010
E PI1     0014
E CONST   0018
I BLK5    0004
E IRITE   0000
I .U      0000
E BLOCK   0E04
E IBLOCK  0E04
E CHAIN   2704
E CRTST   2708
E NEWFL   270C
E OLDFL   2710
E POINT   2714
E PRESPT  2718
E PREVPT  271C
E TOTCHA  2720
E TOTSUM  2724
E SUMIN   2728
E SOURCE  272C
E LARGLA  2730
E LARGLO  2734
E LSN1    273C
E LSN2    2740
E BOUNDX  2744
E BOUNDY  274C
E LSIN    2750
E LSOUT   2754
E DUMMYX  2758
E NSUM    2760
E BSTDP   2774
I COS     0000
E CNTRLP  27C4
A 580     0DF8
I *I      0000
```

| | | |
|---|---|---|
| A | 400 | 0A58 |
| E | N | 27C8 |
| E | M | 27CC |
| A | 410 | 0A62 |
| A | 420 | 0A6E |
| A | 530 | 0CC4 |
| A | 535 | 0CEC |
| A | 540 | 0D18 |
| A | 550 | 0D60 |
| A | 26 | 031A |
| A | 25 | 02EE |
| E | I | 2700 |
| L | @J | 0000 |
| A | 250 | 08F2 |
| E | M1 | 2704 |
| A | 230 | 08E0 |
| E | K | 2708 |
| E | J | 270C |
| E | L | 27E4 |
| E | RLAT | 27EC |
| E | RLONG | 27F0 |
| A | 75 | 03CC |
| A | 125 | 06A2 |
| A | 100 | 0536 |
| E | INBIT | 27F4 |
| E | INBITC | 27F8 |
| A | 430 | 0A7A |
| A | 440 | 0AB8 |
| A | 450 | 0AE2 |
| E | SMALLA | 27FC |
| E | SMALLO | 2800 |
| A | 560 | 0DA8 |
| A | 570 | 0DD0 |
| L | SAVEND | 0000 |
| A | 460 | 0B12 |
| L | TEST1 | 0000 |
| L | PROJEC | 0000 |
| E | X | 2804 |
| E | Y | 2808 |
| L | TEST2 | 0000 |
| A | 225 | 0858 |
| L | SAVELS | 0000 |
| L | SAVEXY | 0000 |
| A | 150 | 06F2 |
| A | 485 | 0BC2 |
| A | 200 | 070C |
| A | 175 | 07B0 |
| A | 470 | 0B3A |
| A | 480 | 0B7E |
| L | AMAX1 | 0000 |
| L | AMIN1 | 0000 |
| E | LSIN1 | 280C |
| A | 490 | 0BF2 |
| A | 500 | 0C2E |

```
A 510      0C6A
A 520      0C94
L .V       0000
```

```
PROGRAMS:
   8152 @J         8232 .P         82BA .Q         836A .O
   83B2 .MES       8430 .U         845E .V         846C @I
   9498 COS        94BC SIN        9576 AMIN1      9588 AMAX1
   959A AMOD       95C8 AINT       963A .2         964C $1
   9662 $3         9694 .COMP      96BA .RRARG     96FC $6
   973E .RARG      9770 $8         979A .5         979E .ZERO
   97A2

ENTRY-POINTS:
   7834 TEST1      78FE TEST2      7BD4 SAVEXY     7CE8 SAVELS
   7E96 SAVEND     7FA4 BUMP       8012 PROJEC     8152 @J
   8232 .P         82BA .Q         836A .O         83B2 .MES
   843C .U         8462 .V         846C @I         9498 COS
   94BC SIN        9576 AMIN1      9588 AMAX1      959A AMOD
   95C8 AINT       963A .2         964C $1         9662 $3
   9694 .COMP      96BA .RRARG     96FC $6         973E .RARG
   9770 $8         979A .5         979E .ZERO

COMMON-BLOCKS:
   E68E BLK1       FF8E BLK2       FF9E BLK3       FFAE BLK4
   FFCA BLK5

UNDEFINED:
NONE
LOADER


XOUT
 LOADER
TA 0400
```

## Subroutine PROJEC

### Purpose

PROJEC projects a point on the globe into the plane

### Procedure Description

At present the projection in use is a conic projection with two standard parallels, also referred to as a secant conic (see Figure 14). The cone intersects the globe along two parallels of latitude which are called standard as they are projected at their proper scale. Between the standard parallels the scale along the parallels is too small, resulting in a shrinking of the projected region. Beyond the standard parallels the parallel scale is too large so that regions may appear stretched in the upper and lower latitudes. The meridians of longitude are always projected at their true scale.

The first step in the projection process is to determine the projected radius of the upper standard parallel (N'Q' in Figure 14). If

r= radius of globe

$\phi^o$ parallel = upper standard parallel

$\theta^o$ parallel - lower standard parallel

Then $N'Q' = 2\pi r \left(\frac{\phi - \theta}{360}\right)\left(\frac{\cos\phi}{\cos\phi - \cos\theta}\right)$

and the distance of an arbitrary point from the apex of the cone is given by

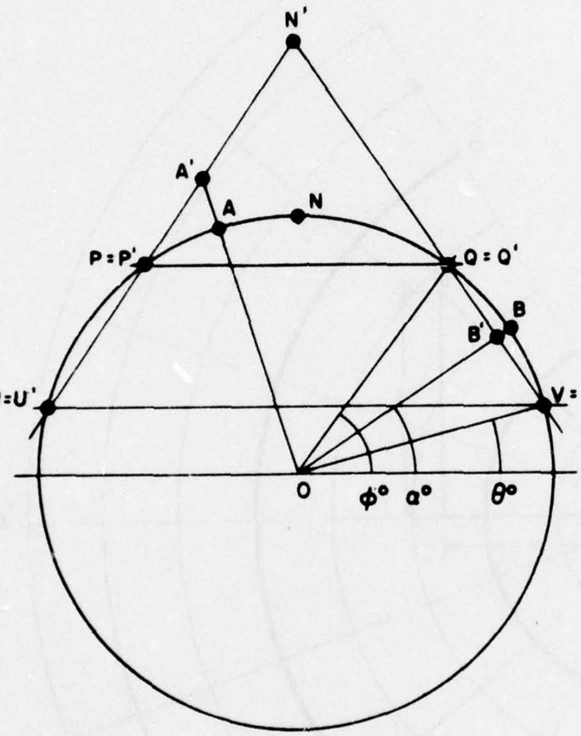
$$N'B' = N'Q' + 2\pi r\left(\frac{\phi - \alpha}{360}\right)$$

if B' is in latitude $\alpha^o$.

As the meridians of longitude are projected at their true angles a point in longitude $\beta^o$ (figure 15, point B') will have coordinates

$x = N'B' \sin\beta'$ where $\beta' = \beta^o - N'M'^o$

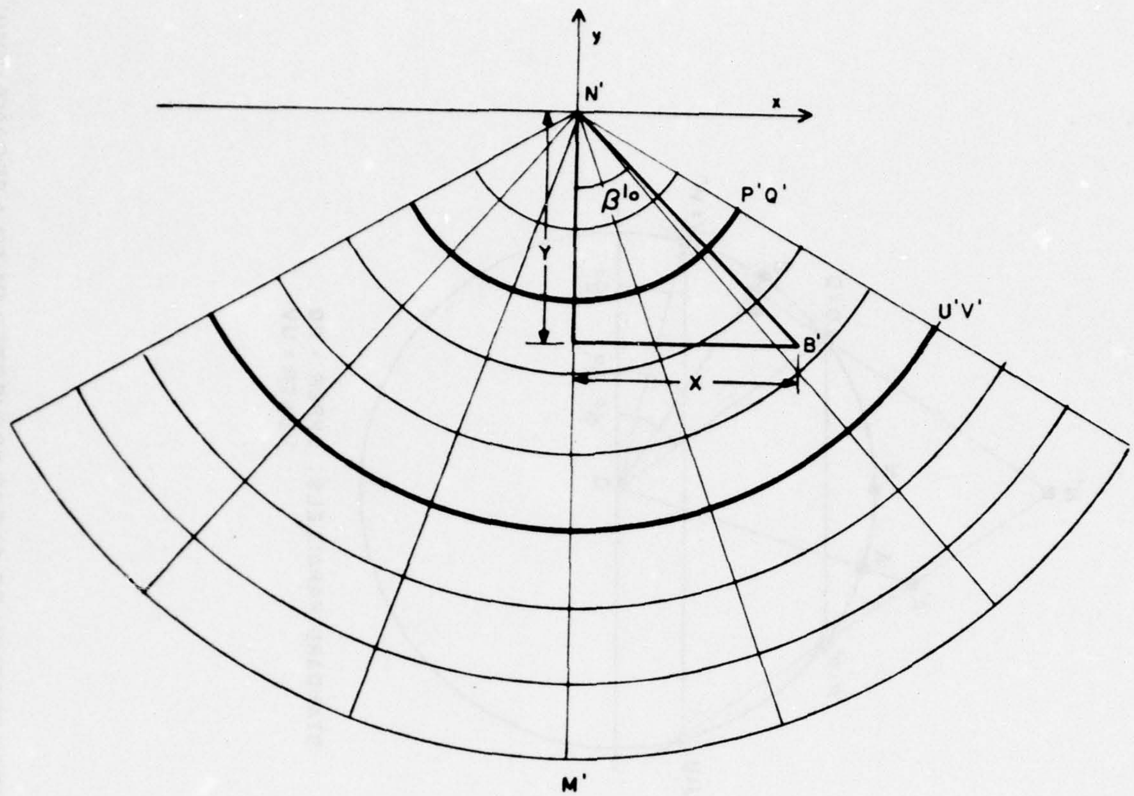
$y = N'B' \cos\beta'$




14 — 48,712

Figure 14 CROSS-SECTION OF GLOBE PROJECTED ONTO A SECANT CONE

IA-48,713




CENTRAL MERIDIAN: N'M'.

Figure 15 SECANT CONE DEVELOPED

The origin of the map can be adjusted by adding constants to the x- and y- coordinates. In PROJEC a constant set equal to the distance of the central parallel[11] from the apex is added to the y-coordinate. Thus, the origin is the intersection of the central meridian with the central parallel.

Common Blocks

| Block Name | Contents of Block | Description of Contents |
|---|---|---|
| /BLK4/ | USTDP | upper standard parallel |
| | RUSTDP | radius of upper standard parallel |
| | CTRLME | central meridian |
| | RGLOBE | radius of globe |
| | PI | 3.14159 |
| | PI1 | 3.14159/180 |
| | CONST | radius of central parallel |

In order to insure a maximum scale error of less than or equal to 1% along the parallels between the parallels $40^{o}N$ and $60^{o}N$ the following constants are set in the main routine:

upper standard parallel = $57^{o}N$

lower standard parallel = $41^{o}N$

in addition

RGLOBE = 1.0

CTRLME = $7.47^{o}E$

Subroutines

None

[11] The central parallel is the mid-parallel from the two standard parallels.

```
      SUBROUTINE PROJEC (RLAT,RLONG,XCRDNT,YCRDNT)
C
C THIS SUBROUTINE PROJECTS A POINT AT LATITUDE RLAT (RADIANS), LONGITUDE RLONG
C(RADIANS) ONTO THE PLANE USING A CONICAL PROJECTION WITH TWO STANDARD PARLLELS
C
CVARIABLE DEFINITIONS
CUSTDP: UPPER STANDARD PARALLEL (DEGREES)
CRGLOBE: RADIUS OF GLOBE
CCTRLME: CENTRAL MERIDIAN (DEGREES)
CRUSTDP: RADIUS OF PROJECTED UPPER STANDARD PARALLEL
CPI1: 3.14159/180.
CCONST: CONSTANT TO BE ADDED TO Y COORDINATE OF PROJECTED POINT.  IT SETS THE
C        ORIGIN OF THE GRAPH AT THE INTERSECTION OF THE CENTRAL MERIDIAN WITH
C        THE CENTRAL PARALLEL.
CRPTPRO: RADIUS OF PROJECTED POINT IN LATITUDE RLAT
CXCRDNT: X COORDINATE OF PROJECTED POINT
CYCRDNT: Y COORDINATE OF PROJECTED POINT
C
      COMMON /BLK4/USTDP,RUSTDP,CTRLME,RGLOBE,PI,PI1,CONST
      RPTPRO=RUSTDP+2.*RGLOBE*PI*((USTDP-RLAT/PI1)/360.)
      IF(RLONG.LT.CTRLME) GO TO 100
C
C LONGITUDE OF POINT IS GREATER THAN OR EQUAL TO CENTRAL MERIDIAN
C
      XCRDNT=RPTPRO*SIN(RLONG-CTRLME)
      YCRDNT=-(RPTPRO*COS(RLONG-CTRLME))+CONST
      GO TO 200
C
C LONGITUDE OF POINT IS LESS THAN CENTRAL MERIDIAN
C
100   XCRDNT=-(RPTPRO*SIN(CTRLME-RLONG))
      YCRDNT=-(RPTPRO*COS(CTRLME-RLONG))+CONST
200   CONTINUE
      RETURN
      END
I BLK4    001C
E USTDP   0000
E RUSTDP  0004
E CTRLME  0008
E RGLOBE  000C
E PI      0010
E PI1     0014
E CONST   0018
K PROJEC  0024
P PROJEC  0140
I .Q      0000
L .P      0000
F RLAT    002A
F RLONG   002C
F XCRDNT  002E
F YCRDNT  0030
F RPTPRO  0148
A 100     00D4
I SIN     0000

 L COS    0000
 A 200    0130
```

Subroutine TEST1

Purpose

TEST1 makes an initial cut on WDBI based on the geographic coordinates of the region to be displayed.

Procedure Description

The latitude and longitude of a data point passed as parameters in the routine are tested against the user-specified min and max geographic coordinates. A flag is set to signal the location of the point with respect to the geographic limits. A flag is also set identifying the location of the point previously processed. By checking the values of these flags upon return to the mainline the current status of the chain is known. The four possible states are (1) chain leaves the region, (2) chain (re-) enters the region, (3) chain remains inside the region, and (4) chain remains outside the region.

Common Block

| Block Name | Contents | Description of Contents |
|---|---|---|
| /BLK2/ | ULAT | Maximum latitude |
| | BLAT | Minimum latitude |
| | WLONG | Western longitude limit |
| | ELONG | Eastern longitude limit |

Subroutines

None

```
      SUBROUTINE TEST1(PREVPT,PRESPT,RLAT,RLONG)
C
C THIS SUBROUTINE CHECKS THE LATITUDE AND LONGITUDE OF A POINT AND SETS A
C(PRESPT)=1 IF THE POINT IS INSIDE THE RANGE AND =0 IF THE POINT IS OUTSID
CRANGE.  IT ALSO SETS PREVPT=1 IF THE PREVIOUS POINT WAS INSIDE THE RANGE
C=0 IF IT WAS OUTSIDE.
C
      COMMON/BLK2/ULAT,BLAT,WLONG,ELONG
      INTEGER PREVPT,PRESPT
      PREVPT=PRESPT
      IF((RLAT.LT.BLAT).OR.(RLAT.GT.ULAT)) GO TO 100
      PRESPT=1
      IF((RLONG.GE.WLONG).AND.(RLONG.LE.ELONG)) GO TO 200
100   PRESPT=0
200   RETURN
      END
I BLK2    0010
E ULAT    0000
E BLAT    0004
E WLONG   0008
E ELONG   000C
K TEST1   0024
P TEST1   008A
L .Q      0000
L .P      0000
F PREVPT  002A
F PRESPT  002C
F RLAT    002E
F RLONG   0030
A 100     009E
A 200     00AA
```

Subroutine TEST2

Purpose

This routine tests the x- and y- coordinates of a data point against the user-specified Cartesian limits of the map. If a chain leaves or (re-) enters the display region and a boundary point needs to be created, the coordinates of the point are determined here.

Procedure Description

As happens in TEST1, the Cartesian coordinates of a data point are tested against user-specified min and max Cartesian coordinates. A flag is set to signal the location of the point with respect to these Cartesian limits. A flag is also set identifying the location of the point previously processed.

These flags are then examined. If their values are not the same the chain has either entered or left the desired region and a boundary point needs to be generated. The algorithm for generating a boundary point can be broken down into three cases (referring to Figure 16):

(1) only one coordinate, either X or Y is outside the range (line segment AB);

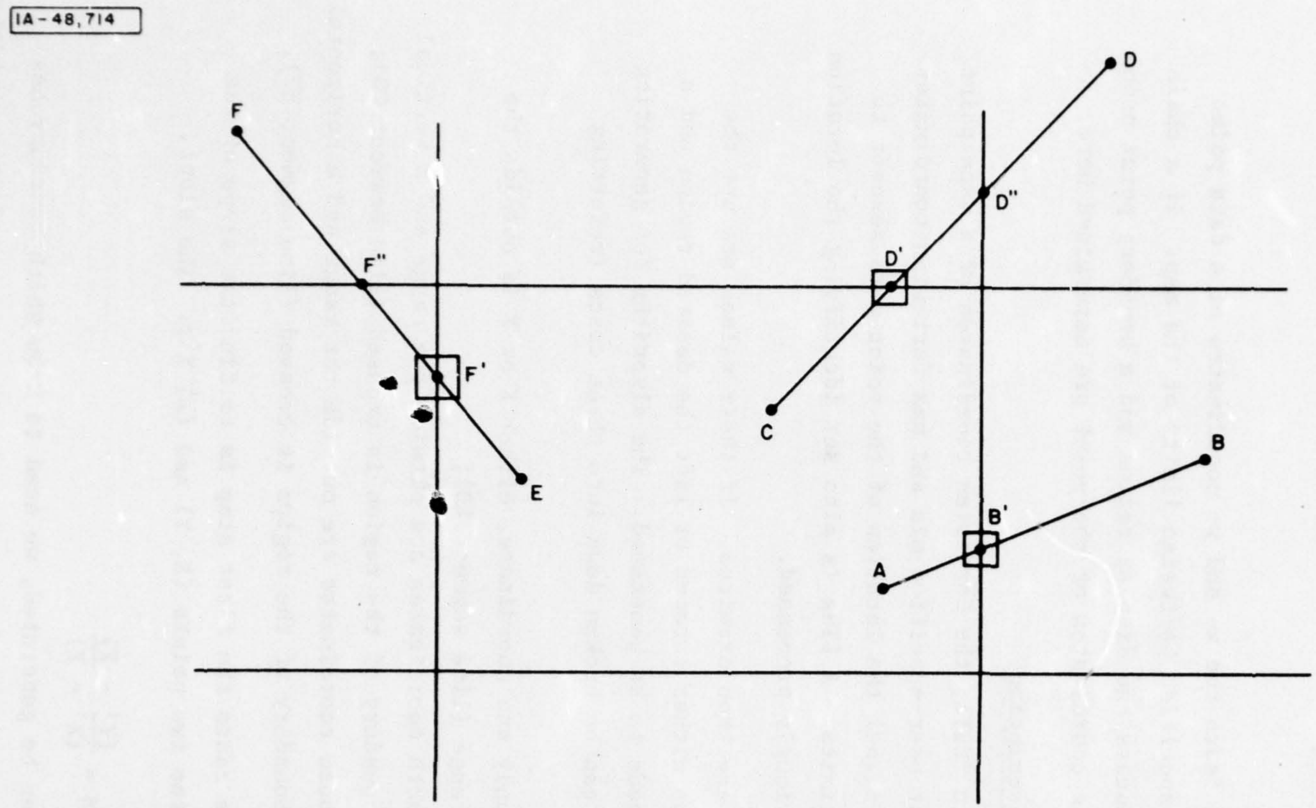
(2) both coordinates are outside the range and a vertical boundary of the region is crossed (line segment CD);

(3) both coordinates are outside the range and a horizontal boundary of the region is crossed (line segment EF).

In all three cases the first step is to find the slope of the line connecting the two points (X, Y) and (X', Y'). The slope, M, is given by:

$$M = \frac{(Y' - X)}{(X' - X)}$$

Before a point can be generated, we need to know which boundary has been crossed. This is done by testing the Cartesian coordinates of the




LINE SEGMENT AB CROSSES A HORIZONTAL BOUNDARY; Y-COORDINATE OF POINT B IS INSIDE Y-RANGE

LINE SEGMENT CD CROSSES A VERTICAL BOUNDARY; BOTH X- AND Y- COORDINATES ARE OUTSIDE THEIR RANGES

LINE SEGMENT EF CROSSES A HORIZONTAL BOUNDARY; BOTH X- AND Y- COORDINATES ARE OUTSIDE THEIR RANGES

Figure 16

point against those of the rectangular boundaries. If only one coordinate is outside its range, (case (1)), we generate the proper value for the other coordinate by:

(1) when X' is outside its range

X" is the X-value of the crossed horizontal boundary

$$Y'' = MX'' - MX + Y$$
$$= M (X'' - X) + Y$$

(2) when Y' is outside its range

Y" is the Y-value of the crossed vertical boundary

$$X'' = (MX + Y'' - Y)/M$$

If both coordinates are outside their respective ranges, Y" is generated by equation (1). If the generated Y" is not inside the Y-range, Y" is set to the Y-value of the crossed vertical boundary and X" is generated by equation (2).

In Figure 16 the Y-coordinate of B' would be determined from the maximum X-value. The Y-coordinate of D" would be determined from the maximum X-value. This Y-value exceeds the Y-range, so Y is set to the maximum Y-value. The X-coordinate of D' is thus determined from the maximum Y. For line segment EF, the Y-coordinate of F' is determined from the minimum X-value and this value is within the Y-range. The coordinates of F" are not calculated.

Common Blocks

| Block Name | Contents of Block | Description of Contents |
|---|---|---|
| /BLK3/ | XMIN | Minimum X value |
| | XMAX | Maximum X value |
| | XMIN | Minimum Y value |
| | XMAX | Maximum Y value |

Subroutines

None

```
      SUBROUTINE TEST2(OLDFL,NEWFL,X,Y,X1,Y1)
      COMMON/BLK3/XMIN,XMAX,YMIN,YMAX
      INTEGER OLDFL,NEWFL
      REAL M
      OLDFL=NEWFL
C
CNEW LS?
C
      IF(NEWFL.GT.1) GO TO 100
      OLDX2=X2
      OLDY2=Y2
C
C TEST X
C
100   NEWFL=1
      X2=X
      IF(X.GT.XMAX) X2=XMAX
      IF(X.LT.XMIN) X2=XMIN
      IF(X2.NE.X) NEWFL=0
C
C TEST Y
C
      Y2=Y
      IF(Y.GT.YMAX) Y2=YMAX
      IF(Y.LT.YMIN) Y2=YMIN
      IF(Y2.NE.Y) NEWFL=0
C
C IF BOTH PREVIOUS AND CURRENT POINTS ARE IN OR OUT

COF IF THE POINT IS FIRST OF LS,  RETURN
C
      IF((OLDFL.EQ.NEWFL).OR.(OLDFL.GT.1)) GO TO 300
C
CDETERMINE BOUNDARY POINT
C
      IF(OLDFL.EQ.1) GO TO 200
C
C SET X2,Y2 TO VALUES FOR PREVIOUS POINT CAUSE YOU ARE ENTERING RECTANGLE.
      X2=OLDX2
      Y2=OLDY2
200   IF(OLDX.NE.X) GO TO 225
      X1=X
      Y1=Y2
      GO TO 300
225   M=(OLDY-Y)/(OLDX-X)
      IF(X2.EQ.X) GO TO 250
      Y1=M*(-OLDX+X2)+OLDY
      X1=X2
      IF((Y1.GE.YMIN).AND.(Y1.LE.YMAX)) GO TO 300
250   X1=(M*OLDX-OLDY+Y2)/M
      Y1=Y2
300   OLDX=X
      OLDY=Y
      RETURN
      END
```

```
I BLK3    0010
E XMIN    0000
E XMAX    0004
E YMIN    0008
E YMAX    000C
K TEST2   0024
P TEST2   029A
I .Q      0000
I .P      0000
F OLDFL   002A
F NEWFL   002C
F X       002E
F Y       0030
F X1      0032
F Y1      0034
F M       02A2
A 100     006C
E OLDX2   02AA
E X2      02AE
E OLDY2   02B2
E Y2      02B6
A 300     0272
A 200     0186
E OLDX    02BE
A 225     018C
E OLDY    02C2
A 250     024A
```

Subroutine SAVELS

Purpose

SAVELS creates a header for chain identification.

Procedure Description

The header for a chain has length of eight words (see Appendix A, Table III). Because positions in the output buffer are always filled in groups of four, the routine can create the first four words of the ID, increment the pointer and then make a single check in the current location of the pointer in the buffer. If the buffer is full, it is dumped to tape and the pointer is reset. The next four words are filled, the pointer is incremented and tested again. The buffer is dumped and the pointer reset if necessary.

Common Blocks

| Block Name | Contents of Block | Description of Contents |
|---|---|---|
| /BLK1/ | XYPTS | output buffer |
| /BLK5/ | IRITE | signals that an end-of-chain has just been written in output buffer |

Subroutines

BUMP: copy output buffer to tape, reset pointer

CALL BUMP (IPOINT)

IPOINT: pointer in output buffer, reset by BUMP

```
      SUBROUTINE SAVELS(LSN1,LSN2,SOURCE,NLSN,IPOINT)
      DIMENSION IXYPTS(1600)
      COMMON /BLK1/ XYPTS(1600)
      COMMON /BLK5/IRITE
      EQUIVALENCE (XYPTS(1),IXYPTS(1))
      INTEGER SOURCE
      IXYPTS(IPOINT)=LSN1
      IXYPTS(IPOINT+1)=LSN2
      IXYPTS(IPOINT+2)=SOURCE
      IPOINT=IPOINT+4
      IF(IPOINT.GT.1600) CALL BUMP(IPOINT)
      IXYPTS(IPOINT)=NLSN
      IXYPTS(IPOINT+1)=NLSN
      IXYPTS(IPOINT+2)=0
      IXYPTS(IPOINT+3)=0
      IPOINT=IPOINT+4
      IF(IPOINT.GT.1600) CALL BUMP(IPOINT)
      IRITE=0
      RETURN
      END
I BLK1    1900
E XYPTS   0000
E IXYPTS  0000
I BLK5    0004
E IRITE   0000
K SAVELS  0024
P SAVELS  019A
L .Q      0000
L .P      0000
F LSN1    002A
F LSN2    002C
F SOURCE  002E
F NLSN    0030
F IPOINT  0032
L BUMP    0000
```

## Subroutine SAVEXY

### Purpose

SAVEXY copies the Cartesian coordinates of a data point into the output buffer.

### Procedure Description

The routine writes the x- and y- coordinate in the buffer and updates the pointer into the buffer four words, thus saving 2 words/ data point deviation and rank to be supplied by DETAIL

### Common Blocks

| Block Name | Contents of Block | Description of Contents |
|---|---|---|
| /BLK1 | XYPTS | Output buffer |
| /BLK5/ | IRITE | Signalling that an end-of-chain has just been written in output buffer. |

### Subroutines

BUMP: copy output buffer to tape, reset pointer

CALL BUMP (IPOINT)

IPOINT: pointer in output buffer, reset by BUMP

```
      SUBROUTINE SAVEXY(FIRST,SECOND,IPOINT)
      DIMENSION IXYPTS(1600)
      COMMON /BLK1/XYPTS(1600)
      COMMON /BLK5/IRITE
      EQUIVALENCE(XYPTS(1),IXYPTS(1))
C
C ON ENTRY WE ASSUME THAT IPOINT=1,5,9,...1597 AND THAT IPOINT POINTS TO T
C NEXT AVAILABLE SLOT.  MUST INCREMENT IPOINT AFTER FILLING AND MUST DUMP
C BUFFER IF FULL.
C
      XYPTS(IPOINT)=FIRST
      XYPTS(IPOINT+1)=SECOND
      XYPTS(IPOINT+2)=0.0
      IXYPTS(IPOINT+3)=0
      IPOINT=IPOINT+4
      IF(IPOINT.GT.1600) CALL BUMP(IPOINT)
      IRITE=0
      RETURN
      END
I BLK1    1900
E XYPTS   0000
E IXYPTS  0000
I BLK5    0004
E IRITE   0000
K SAVEXY  0024
P SAVEXY  00FC
L .Q      0000
L .P      0040
F FIRST   002A
F SECOND  002C
F IPOINT  002E
I BUMP    0000
```

## Subroutine SAVEND

### Purpose

SAVEND places an end-of-chain mark in the output buffer.

### Program Description

SAVEND copies the end-of-chain mark (1000.0) and updates the buffer pointer by four words.

### Common Blocks

| Block Name | Contents of Block | Description of Contents |
| --- | --- | --- |
| /BLK1/ | XYPTS | Output buffer |
| /BLK5/ | IRITE | Signals that an end-of-chain has just been written in output buffer |

### Subroutines

BUMP: copy output buffer to tape, reset pointer

CALL BUMP (IPOINT)

IPOINT: pointer in output buffer, reset by BUMP

```
      SUBROUTINE SAVEND(FIRST,IPOINT)
      DIMENSION IXYPTS(1600)
      COMMON /BLK1/ XYPTS(1600)
      COMMON /BLK5/IRITE
      EQUIVALENCE(XYPTS(1),IXYPTS(1))
      XYPTS(IPOINT)=FIRST
      IXYPTS(IPOINT+1) = 0
      IXYPTS(IPOINT+2)=0
      IXYPTS(IPOINT+3)=0
      IPOINT=IPOINT+4
      IF(IPOINT.GT.1600) CALL BUMP(IPOINT)
      IRITE=1
      RETURN
      END
I BLK1    1900
E XYPTS   0000
E IXYPTS  0000
I BLK5    0004
E IRITE   0000
K SAVEND  0024
P SAVEND  00F6
L .Q      0000
L .P      0000
F FIRST   002A
F IPOINT  002C
L BUMP    0000
```

```
      SUBROUTINE BUMP(IPOINT)
      COMMON /BLK1/XYPTS(1600)
      WRITE(4) XYPTS
      IPOINT=1
      RETURN
      END
I BLK1     1900
E XYPTS    0000
K BUMP     0024
P BUMP     0062
I .Q       0000
L .P       0000
F IPOINT   002A
L *J       0000
```

AD-A037 116 MITRE CORP BEDFORD MASS F/G 8/2
GEOGRAPHIC DATA BASE DEVELOPMENT.(U)
JAN 77 A M MOLLOY F19628-76-C-0001
UNCLASSIFIED MTR-3312 ESD-TR-76-360 NL

2 OF 2
AD
A037116

END
DATE
FILMED
4-77

DETAIL

Purpose

DETAIL is a point-ranking mechanism. By assigning a priority level to each point in the data base the user controls the amount of data to be displayed by specifying a minimum rank which a point must have to be retained in the map.

Procedure Description

Assignment of Rank. The projected and reformatted data produced by SUBSET is organized in chains. A chain is defined as a series of adjacent points forming a coastline or boundary line segment.

Data points are assigned a rank or detail level on a chain-by-chain basis. The first step in the process is to assign to the endpoints of the chain the highest rank. A "trend line" is constructed between these points and the deviation of each data point from the trend line is computed (see Figure 17 (a)). The point having the greatest deviation, point C, is assigned a detail level and a new trend line is constructed between points A and C. The deviations of all points between these endpoints are determined (see Figure 17 (b)). This time point D is assigned a detail level and another trend line is constructed to include D. This process continues until all points between two endpoints have detail levels assigned. When this occurs new endpoints are chosen such that the first endpoint is the last successive data point with a detail level already assigned and the other endpoint is the first ranked data point following the newly assigned endpoint. For example, in Figure 17 (c), if all points between A and D had detail level assigned, and if both of the points between D and C had not been given detail levels, D would remain an endpoint and C would become one. If the maximum deviation of points from the trend line is held by more than one point, all points with

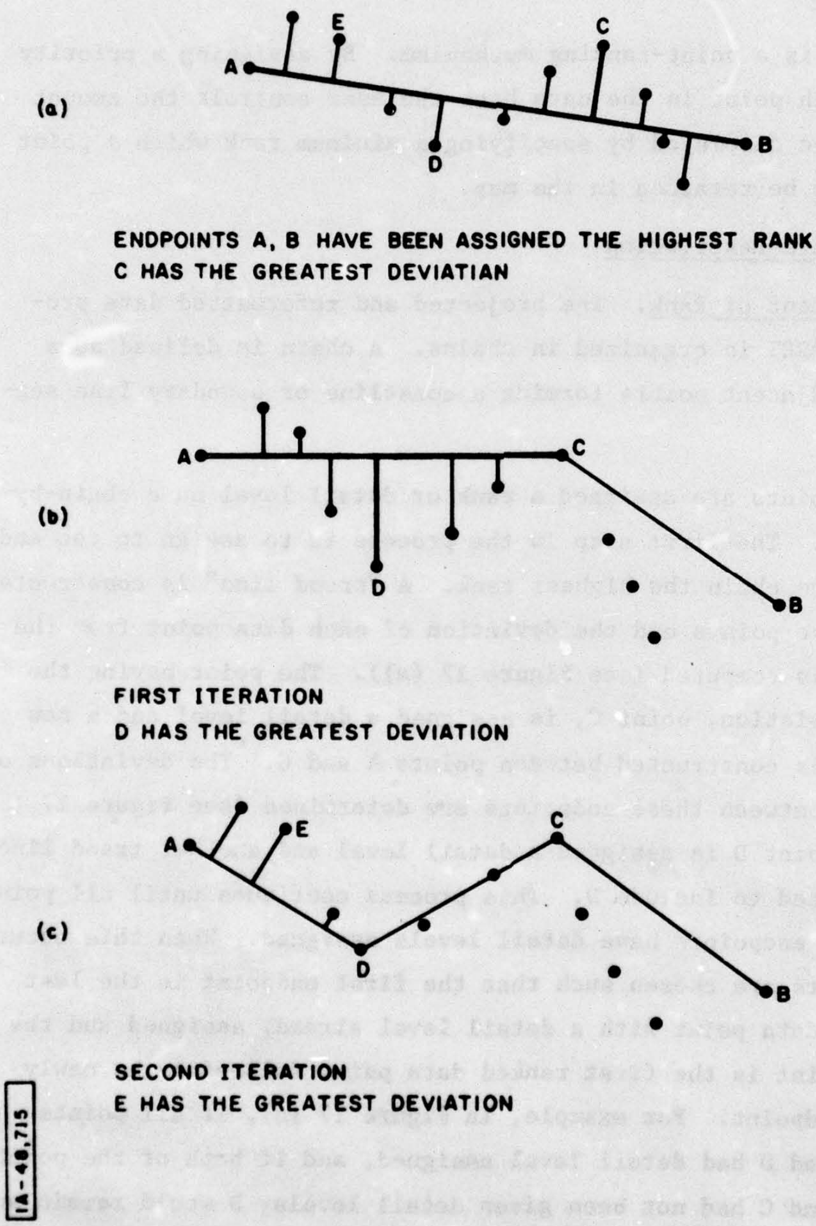



**Figure 17 RANK ASSIGNMENT**

that deviation are ranked the same.

Relationship Between Rank and Deviation. Each rank corresponds to a band whose width has been determined by the user.[12] A point is assigned the rank corresponding to the largest band which is exceeded by either the deviation of the point or the product of its deviation and trend line length.[13]

It is possible that the deviation of a particular point in the $n^{th}$ iteration of the deviation calculations will be greater than that calculated in an earlier iteration. In fact, the deviation of a point in the $n^{th}$ iteration could be greater than or equal to the deviation calculated for the endpoints of that section of trend line. For example, referring again to Figure 17, the deviation of point D in (b) is greater than that calculated for point C or D in (a). Thus a restriction on detail level assignment is necessary. If the deviation of a point is so great so as to assign that point a higher rank than the minimum assigned to the endpoints of its associated line, the point receives the detail level equal to that minimum.

Common Blocks

None

Subroutines

CDLEV: calculates the deviations of points from a trend line and determines the point with the greatest deviation

CALL CDLEV (X, Y, I, J, D, P, MD, TLL)

---

[12] For information on band specification see Section III, User's Guide to DETAIL.

[13] The choice of using the deviation of the point or the product of deviation and trend line length for rank assignment is made by the user. For information on metric choice see Section II, Data Reduction.

X: array containing the x-coordinates of all points in the chain being processed

Y: array containing the y-coordinates of all points in the chain being processed

I: pointer to the first endpoint of the trend line

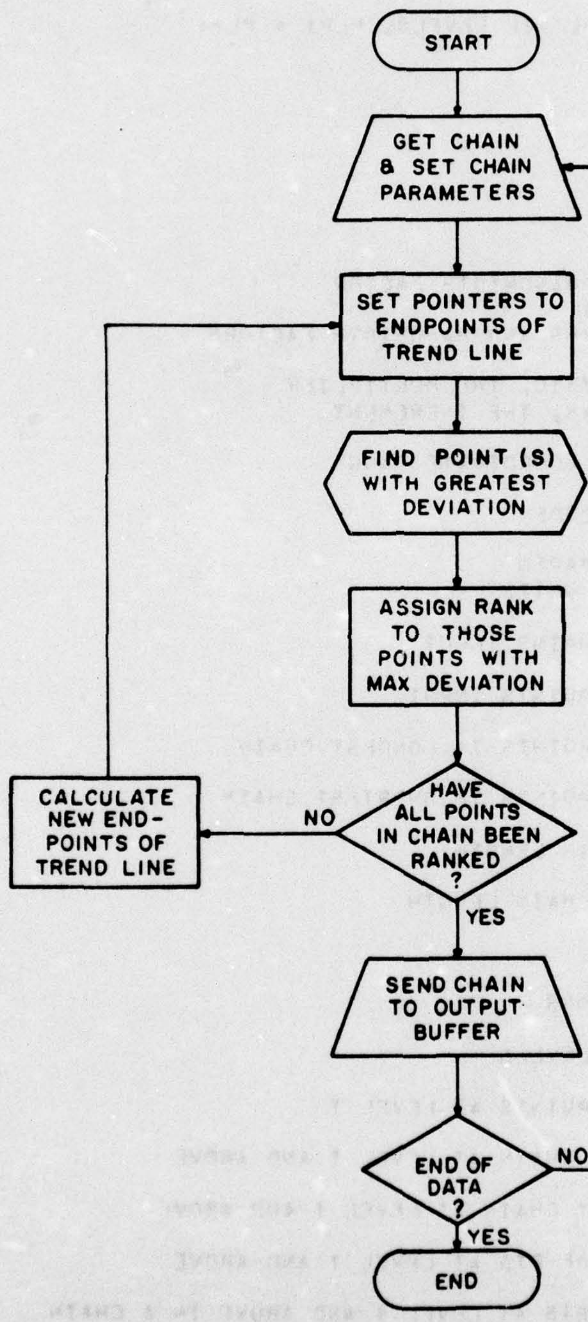
J: pointer to the final endpoint of the trend line

D: array containing -1 for each element yet to be assigned a detail level and the rank of those elements which have been assigned one

P: pointer to the element in chain with current maximum deviation (i.e., to the element to be assigned a detail level upon return to DETAIL)

MD: deviation of point pointed to by P

TLL: current trend line length




IA-48,716

**Figure 18 DETAIL FLOWCHART**

```
      INTEGER ML,MLP1
C              - NO, OF DETAIL LEVELS, MLP1 = ML+1
      INTEGER DEBUG
      INTEGER I,J,P
      INTEGER K
      INTEGER L
C
      INTEGER I1,J1
      INTEGER I2,I3
C
      REAL MINBW
C              - MINIMUM BANDWIDTH FACTOR
      REAL BW(21),BWF(21)
C              - BANDWIDTHS AND BANDWIDTH FACTORS
      REAL FACTOR
C              - INGEOMETRIC, THE MULTIPLIER
C              - IN LINEAR, THE INCREMENT
      REAL MCW
C              - MINIMUM COORDINATE RANGE
      INTEGER MCODE
C              - METRIC CODE
      INTEGER PTR
      INTEGER TTY,TAPEI,TAPEO
C              - LOGICAL UNITS
      INTEGER NCS
C              - NO OF CHAINS INPUT
      INTEGER NPS
C              - NO, OF POINTS INPUT
      INTEGER NPLC
C              - NO, OF POINTS IN LONGEST CHAIN
      INTEGER NPSC
C              - NO, OF POINTS IN SHORTEST CHAIN
      INTEGER M
C              - MAX CHAIN LENGTH
      INTEGER N
C              - ACTUAL CHAIN LENGTH
      REAL X(800),Y(800)
C              - CHAIN
      REAL DEV(800)
C              - DEVIATIONS
      INTEGER D(800)
C              - DETAIL LEVELS
      INTEGER NP(21)
C              - NO, OF POINTS AT LEVEL I
      INTEGER CLC(21)
C              - LONGEST CHAIN AT LEVEL I AND ABOVE
      INTEGER CSC(21)
C              - SHORTEST CHAIN AT LEVEL I AND ABOVE
      INTEGER CNP(21)
C              - NUMBER OF PTS AT LEVEL I AND ABOVE
      INTEGER TCCL(21)
C              - NO, OF PTS AT LEVEL I AND ABOVE IN A CHAIN
      INTEGER TNP(21)
C              - NO, OF PTS AT LEVEL I IN A CHAIN
```

```
      INTEGER MAXMI
C               -MAX DLEV ALLOWED FOR A POINT IN A CHAIN
      REAL IBUFFR(1600),OBUFFR(1600)
      INTEGER IBUFFI(1600),OBUFFI(1600)
      EQUIVALENCE (IBUFFR(1),IBUFFI(1)),(OBUFFI(1),OBUFFR(1))
C               - INPUT AND OUTPUT BUFFERS
      INTEGER IBP,OBP,IBS,OBS
C               - BUFFER POINTERS AND SIZES
      INTEGER BC
C               - INPUT BLOCK COUNT
      INTEGER R,Q
C               - R IS EXCESS CHAIN LENGTH ON INPUT
C               - Q IS FLAG FOR -TOO LONG CHAIN- CONDITION
      REAL EOC
C               - END OF CHAIN CODE
      REAL MD
C               - MAX DEVIATION OF PT FROM TRAND LINE
      REAL TLL
C               - TREND LINE LENGTH
      INTEGER LSN1,LSN2,SOURCE,SSN,NLSN
C               - HEADER DATA
C
C INITIAL IZATION
C
      PTR = 3
      TTY = 5
      TAPEI = 1
      TAPEO = 6
C
C GET INPUT PARAMETERS
C
      WRITE(TTY,900)
900   FORMAT(14HDETAIL DEFINER)
      WRITE(TTY,899)
899   FORMAT(25HENTER 1 FOR DEBUG, ELSE 0)
      READ(TTY,898) DEBUG
898   FORMAT(I1)
C
      WRITE(TTY,931)
931   FORMAT(21HENTER CODE FOR METRIC)
5     WRITE(TTY,932)
932   FORMAT(37HENTER 1 FOR DEVIATION FROM TREND LINE)
      WRITE(TTY,933)
933   FORMAT(54HENTER 2 FOR PRODUCT OF DEVIATION AND TREND LINE LENGTH)
      READ(TTY,934) MCODE
934   FORMAT(I1)
      IF ((MCODE.NE.1).AND.(MCODE.NE.2)) GO TO 5
C
10    WRITE(TTY,901)
901   FORMAT (41HENTER NO. OF DETAIL LEVELS (NN, UP TO 20))
      READ(TTY,902) ML
902   FORMAT(I2)
      IF(ML.GT.20) GO TO 10
      IF (ML.LT.1) GO TO 10
```

```
      MLP1 = ML+1
C
      WRITE(TTY,903)
903   FORMAT(27HENTER BANDWIDTH SPEC METHOD)
20    WRITE(TTY,904)
904   FORMAT(21HENTER 1 FOR GEOMETRIC)
      WRITE(TTY,905)
905   FORMAT(18HENTER 2 FOR LINEAR)
      WRITE(TTY,906)
906   FORMAT(21HENTER 3 FOR ARBITRARY)
      READ(TTY,907) I
907   FORMAT(I1)
      IF(I.EQ.1) GO TO 30
      IF(I.EQ.2) GO TO 40
      IF(I.EQ.3) GO TO 50
      GO TO 20
C
30    WRITE(TTY,908)
908   FORMAT(29HENTER MIN BANDWIDTH (E14.7  ))
      READ(TTY,909)   MINBW
909   FORMAT(E14.7)
      BWF(2) = MINBW
      IF(ML.EQ.1)GO TO 70
60    WRITE(TTY,910)
910   FORMAT(38HENTER BAND MULTIPLIER (N.NNN,OVER 1.0))
      READ(TTY,911)FACTOR
911   FORMAT(F5.3)
      IF(FACTOR.LE.1.0)GO TO 60
      DO 80 I=3,MLP1
80    BWF(I) = FACTOR*BWF(I-1)
      GO TO 70
C
40    WRITE(TTY,908)
      READ(TTY,909)MINBW
      BWF(2) = MINBW
      IF(ML.EQ.1) GO TO 70
100   WRITE(TTY,912)
912   FORMAT(25HENTER INCREMENT (N.NNNNN))
      READ(TTY,909) FACTOR
      IF (FACTOR.LT.0.0) GO TO 100
      DO 90 I=3,MLP1
90    BWF(I) = FACTOR+BWF(I-1)
      GO TO 70
C
50    WRITE(TTY,913) ML
913   FORMAT(5HENTER,I3,21H BANDWIDTHS (N.NNNNN))
      DO 110 I=2,MLP1
      J = I-1
120   WRITE(TTY,914) J
914   FORMAT(15HENTER BANDWIDTH,I3)
      READ(TTY,909)BWF(I)
      IF(I.EQ.2) GO TO 110
      IF(BWF(I).GT.BWF(I-1)) GO TO 110
      GO TO 120
```

```
110     CONTINUE
C
70      WRITE(TTY,915)
915     FORMAT(29HENTER MIN COORD RANGE (F14.8))
        READ(TTY,909) MCW
        DO 130 I=2,MLP1
130     BW(I) = BWF(I)*MCW
        BW(1) = 0.0
        BWF(1) = 0.0
C
C INITIALIZATION
C
        NCS = 0
        NPS = 0
        NPLC = 0
        NPSC = 32767
C*******************
C*********THIS SHOULD BE CHANGED WHEN CHANGING DIMENSIONS
C*********FOR X,Y,D,DEV
C*************
        M=550
C*************
        WRITE(PTR,916) M
916     FORMAT(19HMAX CHAIN LENGTH IS,I5)
        DO 140 I=1,MLP1
        NP(I) = 0
        CLC(I) = 0
140     CSC(I) = 32767
C
        READ(TAPEI) IBUFFR
        IBP = -3
        OBP = -3
        IBS = 1600
        OBS = 1600
        BC = 0
        EOC = 1000.0
C
C GET THE NEXT CHAIN
C
C
C IBP -> LAST 4-WORD BLOCK
C
C GET THE LSN1 4-WORD BLOCK
C
160     IBP = IBP+4
        IF (IBP.LT.IBS) GO TO 161
        READ (TAPEI) IBUFFR
        IBP = 1
161     IF (IBUFFR(IBP).EQ.EOC) GO TO 201
        LSN1 = IBUFFI(IBP)
        LSN2 = IBUFFI(IBP+1)
        SOURCE = IBUFFI(IBP+2)
        T1 = IBUFFI(IBP+3)
C GET THE LSN2 4-WORD BLOCK
```

```
C
      IBP = IBP+4
      IF (IBP.LT.IBS) GO TO 162
      READ (TAPEI) IBUFFR
      IBP = 1
162   SSN = IBUFFI(IBP)
      NLSN = IBUFFI(IBP+1)
      I2 = IBUFFI(IBP+2)
      I3 = IBUFFI(IBP+3)
C
C PASS IT TO THE OUTPUT
C
      OBP = OBP+4
      IF (OBP.LT.1600) GO TO 163
      WRITE (TAPEO) OBUFFR
      OBP = 1
163   OBUFFI(OBP) = LSN1
      OBUFFI(OBP+1) = LSN2
      OBUFFI(OBP+2) = SOURCE
      OBUFFI(OBP+3) = I1
      OBP = OBP+4
      IF (OBP.LT.OBS) GO TO 164
      WRITE (TAPEO) OBUFFR
      OBP = 1
164   OBUFFI(OBP) = SSN
      OBUFFI(OBP+1) = NLSN
      OBUFFI(OBP+2) = I2
      OBUFFI(OBP+3) = I3
      R = 0
      N = 0
      Q = 0
C
170   IBP = IBP+4
      IF(IBP.LT.IBS) GO TO 180
      READ(TAPEI)IBUFFR
      IBP = 1
      BC = BC+1
      IF(MOD(BC,10).NE.0)GO TO 180
      WRITE(PTR,917) BC
917   FORMAT(5HBLOCK,I4)
C
180   IF(IBUFFR(IBP).EQ.EOC) GO TO 200
      IF(N.EQ.M) GO TO 190
      N = N+1
185   X(N) = IBUFFR(IBP)
      Y(N) = IBUFFR(IBP+1)
      GO TO 170
190   Q = 1
      R = R+1
      GO TO 185
C
200   IF(Q.NE.1) GO TO 224
      Q = NCS+1
      WRITE (PTR,918) BC,N,R
```

```
918    FORMAT(5HBLOCK,I4,6H CHAIN,I5,7H EXCESS,I5)
       GO TO 220
C
C
C END OF INPUT FILE
C
201    OBP = OBP+4
       IF(OBP.LT.OBS)GO TO 230
       WRITE(TAPEO) OBUFFR
       OBP = 1
230    OBUFFR(OBP) = EOC
       OBUFFI(OBP+1) = 0
       OBUFFI(OBP+2) = 0
       OBUFFI(OBP+3) = 0
       WRITE(TAPEO) OBUFFR
       CALL EOF(TAPEO)
C
       CNP(MLP1) = NP(MLP1)
       I = ML
240    IF(I.EQ.0)GO TO 250
       CNP(I) = NP(I)+CNP(I+1)
       I = I-1
       GO TO 240
C
250    WRITE(TTY,919)
919    FORMAT(2HDL,1X,3HBWF,5X,2HBW,6X,2HNP,4X,3HCNP,3X,3HCLC,3X,3HCSC)
       DO 260 I=1,MLP1
       J = I-1
260    WRITE(TTY,920)J,BWF(I),BW(I),NP(I),CNP(I),CLC(I),CSC(I)
920    FORMAT(I2,1X,F7.5,1X,F7.5,1X,I5,1X,I5,1X,I5,1X,I5)
       WRITE(TTY,921)NCS
921    FORMAT(I5,1X,16HCHAINS PROCESSED)
       WRITE(TTY,922)NPS
922    FORMAT(I5,1X,16HPOINTS PROCESSED)
       WRITE(TTY,923)NPLC
923    FORMAT(I5,1X,23HPOINTS IN LONGEST CHAIN)
       WRITE(TTY,924)NPSC
924    FORMAT(I5,1X,24HPOINTS IN SHORTEST CHAIN)
       WRITE(TTY,925)
925    FORMAT(4HDONE)
       CALL REW(TAPEI)
       CALL REW(TAPEO)
       STOP
C
C HAVE A NEW CHAIN IN X AND Y
C SET GLOBAL STATISTICS
C
220    NCS = NCS+1
       NPS = NPS+N
       IF(N.GT.NPLC) NPLC = N
       IF(N.LT.NPSC) NPSC=N
C
C INITIALIZE FOR CHAIN PROCESSING
C
```

```
      DO 270 I=1,N
270   D(I) = -1
      D(1) = ML
      D(N) = ML
      DEV(1) = 0.0
      DEV(N) = 0.0
      DO 280 I=1,MLP1
280   TNP(I) = 0
      TNP(MLP1) = 2
      I = 1
      P = N
C
C FIND THE NEXT I AND J
C
290   J = P
      IF(J.GT.(I+1)) GO TO 300
310   I = I+1
      IF(I.EQ.N) GO TO 330
      IF(D(I).NE.-1) GO TO 310
      I = I-1
      J = I+2
320   IF(D(J).NE.-1) GO TO 300
      J = J+1
      GO TO 320
C
C PERFORM TREND LINE CALCULATION
C
300   CONTINUE
      IF(DEBUG.EQ.0) GO TO 301
      WRITE(PTR,896)I,J,P,N
896   FORMAT(2HI=,I5,3H J=,I5,3H P=,I5,3H N=,I5)
      WRITE(PTR,897)(K,X(K),Y(K),D(K),K=1,N)
897   FORMAT(I5,1X,2HX=,F7.5,3H Y=,F7.5,3H D=,I5)
301   CONTINUE
      CALL CDLEV (X,Y,I,J,D,P,MD,TLL)
      IF(DEBUG.EQ.0) GO TO 302
      WRITE(PTR,895)I,J,P,N
895   FORMAT(2HI=,I5,3H J=,I5,3H P=,I5,3H N=,I5)
      WRITE(PTR,894)(K,X(K),Y(K),D(K),K=1,N)
894   FORMAT(I5,1X,2HX=,F7.5,3H Y=,F7.5,3H D=,I5)
302   CONTINUE
C**********
      IF(MCODE.EQ.1) GO TO 339
      IF (TLL.NE.0.0) GO TO 338
      WRITE (PTR,935)
935   FORMAT (17HZERO LENGTH CHAIN)
      GO TO 339
338   MD=MD*TLL
339   CONTINUE
C**********
      I1 = 1
340   IF(I1.GT.MLP1) GO TO 350
      IF(MD.LE.B*(I1)) GO TO 360
      I1 = I1+1
```

```
      GO TO 340
350   I1 = MLP1
360   I1 = I1-1
C
      J1 = 0
370   L = D(P)
      IF(DEBUG.EQ.0)GOTO371
      WRITE(PTR,893)L,P
893   FORMAT(2HL=,I5,3H P=,I5)
371   CONTINUE
      MAXML=D(I)
      IF(D(I).NE.D(J))  MAXML=MIN0(D(I),D(J))
      IF(I1.GT.MAXML) I1=MAXML
      D(P) = I1
      DEV(P) = MD
      J1 = J1+1
      IF(L.EQ.-1)GO TO 380
      P = L
      GO TO 370
C
380   TNP(I1+1)=TNP(I1+1)+J1
      IF(J1.EQ.1) GO TO 290
      WRITE(PTR,926) NCS,J1
926   FORMAT(5HCHAIN,I5,6HEQ PTS,I4)
      GO TO 290
C
C END OF CHAIN
C
330   WRITE(PTR,930)NCS,LSN1,LSN2,SOURCE,SSN,NLSN,N,
     1     (TNP(I),I=1,MLP1)
930   FORMAT(4HCHN=,I3,1X,I3,I4,I2,I6,I6,3H N=,I3,21I5)
      DO 390 I=1,N
      OBP = OBP+4
      IF(OBP.LT.OBS) GO TO 400
      WRITE(TAPEO) OBUFFR
      OBP = 1
400   OBUFFR(OBP) = X(I)
      OBUFFR(OBP+1) = Y(I)
      OBUFFR(OBP+2) = DEV(I)
      OBUFFI(OBP+3) = D(I)
390   CONTINUE
      OBP = OBP+4
      IF(OBP.LT.OBS) GO TO 410
      WRITE(TAPEO) OBUFFR
      OBP = 1
410   OBUFFR(OBP) = EOC
      OBUFFI(OBP+1) = 0
      OBUFFI(OBP+2) = 0
      OBUFFI(OBP+3) = 0
C
      DO 420 I=1,MLP1
420   NP(I) = NP(I) + TNP(I)
      TCCL(MLP1) = TNP(MLP1)
      I = M1
```

```
430     IF(I.EQ.-1) GO TO 440
        TCCL(I) = TNP(I)+TCCL(I+1)
        I = I-1
        GO TO 430
440     DO 450 I=1,MLP1
        IF(TCCL(I).GT.CLC(I)) CLC(I)=TCCL(I)
        IF(TCCL(I).LT.CSC(I)) CSC(I) = TCCL(I)
450     CONTINUE
        GO TO 160
        END
I  .U      0000
E  ML      1B26
E  MLP1    1B2A
E  DEBUG   1B2E
E  I       1B32
E  J       1B36
E  P       1B3A
E  K       1B3E
E  L       1B42
E  I1      1B46
E  J1      1B4A
E  I2      1B4E
E  I3      1B52
E  MINBW   1B56
E  BW      1B5A
E  BWF     1BAE
E  FACTOR  1C02
E  MCW     1C06
E  MCODE   1C0A
E  PTR     1C0E
E  ITY     1C12
E  TAPEI   1C16
E  TAPEO   1C1A
E  NCS     1C1E
E  NPS     1C22
E  NPLC    1C26
E  NPSC    1C2A
E  M       1C2E
E  N       1C32
E  X       1C36
E  Y       2886
E  DEV     3536
E  D       4186
E  NP      4F36
E  CLC     4F8A
E  CSC     4FDE
E  CNP     4F32
E  TCCL    4F86
E  TNP     4FDA
E  MAXML   502E
E  IBUFFR  5032
E  IBUFFI  5032
E  OBUFFR  6932
E  OBUFFI  6932
```

| | | |
|---|---|---|
| I | ØJ | ØØØØ |
| A | 16Ø | Ø8BA |
| A | 161 | Ø8CA |
| A | 2Ø1 | ØCF2 |
| A | 162 | Ø98A |
| A | 163 | ØA2C |
| A | 164 | ØACE |
| A | 17Ø | ØB48 |
| A | 18Ø | ØBDC |
| L | MOD | ØØØØ |
| A | 917 | ØBCC |
| A | 2ØØ | ØC7A |
| A | 19Ø | ØC62 |
| A | 185 | ØC18 |
| A | 22Ø | 1ØD6 |
| A | 918 | ØCC4 |
| A | 23Ø | ØD32 |
| I | EOF | ØØØØ |
| A | 24Ø | ØDDE |
| A | 25Ø | ØE36 |
| A | 919 | ØE4A |
| A | 26Ø | ØE9C |
| A | 92Ø | ØF72 |
| A | 921 | ØFBE |
| A | 922 | ØFFA |
| A | 923 | 1Ø36 |
| A | 924 | 1Ø78 |
| A | 925 | 1ØB4 |
| I | REW | ØØØØ |
| I | .S | ØØØØ |
| A | 27Ø | 1122 |
| A | 28Ø | 1194 |
| A | 29Ø | 11DE |
| A | 3ØØ | 1286 |
| A | 31Ø | 11FC |
| A | 33Ø | 176A |
| A | 32Ø | 1254 |
| A | 3Ø1 | 13A8 |
| A | 896 | 12CC |
| A | 897 | 137E |
| I | CDLEV | ØØØØ |
| A | 3Ø2 | 14EØ |
| A | 395 | 1444 |
| A | 394 | 14B6 |
| A | 339 | 1542 |
| A | 338 | 1536 |
| A | 935 | 1518 |
| A | 34Ø | 154A |
| A | 35Ø | 158A |
| A | 36Ø | 1592 |
| A | 37Ø | 15A6 |
| A | 371 | 16Ø6 |
| A | 893 | 15FØ |
| I | MINØ | ØØØØ |

```
E  IBP      8232
E  OBP      8236
E  IBS      823A
E  OBS      823E
E  BC       8242
E  R        8246
E  Q        824A
E  EOC      824E
E  MD       8252
E  TLL      8256
E  LSN1     825A
E  LSN2     825E
E  SOURCE   8262
E  SSN      8266
E  NLSN     826A
A  900      0038
L  *I       0000
A  899      0064
A  898      00A2
A  931      00BE
A  5        00DC
A  932      00F0
A  933      0132
A  934      018E
A  10       01B6
A  901      01CA
A  902      0218
A  903      0264
A  20       0288
A  904      029C
A  905      02CE
A  906      02FE
A  907      0338
A  30       037A
A  40       044E
A  50       05A8
A  908      038E
A  909      03D0
A  70       0602
A  60       03FA
A  910      042E
A  911      045A
A  80       047E
A  100      053C
A  912      0520
A  90       0578
A  913      05D4
A  110      06C0
A  120      0612
A  914      062E
A  915      06F6
A  130      0740
A  916      07C2
A  140      0812
```

```
F  X        002A
F  Y        002C
F  I        002F
F  J        0030
F  L        0032
F  P        0034
F  MD       0036
F  IL       0038
E  MDEV     03CC
E  XBAR     03D0
E  YBAR     03D4
E  TO       03D8
E  Z        03DC
E  XNJ      03E0
E  YNJ      03E4
E  TEMP     03E8
E  K        03EC
E  K2       03F0
A  200      0344
E  N        040C
A  25       0220
A  30       0274
I  ABS      0000
A  150      031A
A  50       0298
A  100      0302
A  250      03A8
I  .A       0000

   PROGRAMS:
     D6CE @J        D7AE .P        D836 .Q        D8E6 .O
     D92E .MES      D9AC .U        D9DA .V        D9E8 @I
     EA14 MOD       EA4A ABS       EA66 .COMP     EA8C .IIARG
     EAC8 .RARG     EAFA EOF       EB1A REW       EB3A MINO
     EB4C .1        EB5C $1        EB72 $2        EBB0 .S
     EBE4 .A        EC96 ALOG      ED9E EXP       EEA4 AINT
     EF16 $6        EF58 $8        EF82 .5        EF86 .ZERO
     EF8A

   ENTRY-POINTS:
     D2D6 CDLEV     D6CE @J        D7AE .P        D836 .Q
     D8E6 .O        D92E .MES      D9B8 .U        D9DE .V
     D9E3 @I        EA14 MOD       EA4A ABS       EA66 .COMP
     EA8C .IIARG    EAC8 .RARG     EAFA EOF       EB1A REW
     EB3A MINO      EB4C .1        EB5C $1        EB72 $2
     EBB0 .S        EBE4 .A        EC96 ALOG      ED9E EXP
     EEA4 AINT      EF16 $6        EF58 $8        EF82 .5
     EF86 .ZERO

   COMMON-BLOCKS:
   NONE

   UNDEFINED:
   NONE
```

## Subroutine CDLEV

### Purpose

CDLEV calculates the deviation of points from a trend line and returns pointer (s) to the point (s) with maximum deviation.

### Procedure Description

The deviation of a point from its associated trend line is calculated as follows (refer to Figure 19 (a)).

$$\text{let } XBAR = X(J) - X(I)$$

$$YBAR = Y(J) - Y(I)$$

$$\text{slope of line } \ell = m_\ell = \frac{YBAR}{XBAR}$$

$$\text{equation of line } \ell: Y - Y(I) = m_\ell (X - X(I))$$

$$0 = m_\ell X - m_\ell (X(I) - Y + Y(I))$$

$$= YBAR\ X - YBAR\ X(I) - XBAR\ Y + XBAR\ Y(I)$$

$$= YBAR\ X - XBAR\ Y - Y(J)\ X(I) + X(J)\ Y(I)$$

$$\text{let } Z = -X(I)\ Y(J) + X(J)\ Y(I)$$

and $\ell$ is given by

$$YBAR\ X - XBAR\ Y + Z = \emptyset$$

The distance d, of a point (X(N), Y(N)) from line $\ell$ is given by

$$d = \frac{YBAR\ (X(N)) - XBAR\ (Y(N)) + Z}{\sqrt{YBAR^2 + XBAR^2}}$$

The deviations of data points from trend lines are all calculated in this manner with the exception of those cases in which the endpoints of the trend line are coincident (i.e., island chains). In such cases the deviation is measured as the distance of the point from the endpoints of the chain referring to Figure 19 (b).

$$\text{if } \Delta X = X(N) - X(J)$$

$$\Delta Y + Y(N) - Y(J)$$

$$d = \sqrt{\Delta X^2 + \Delta Y^2}$$

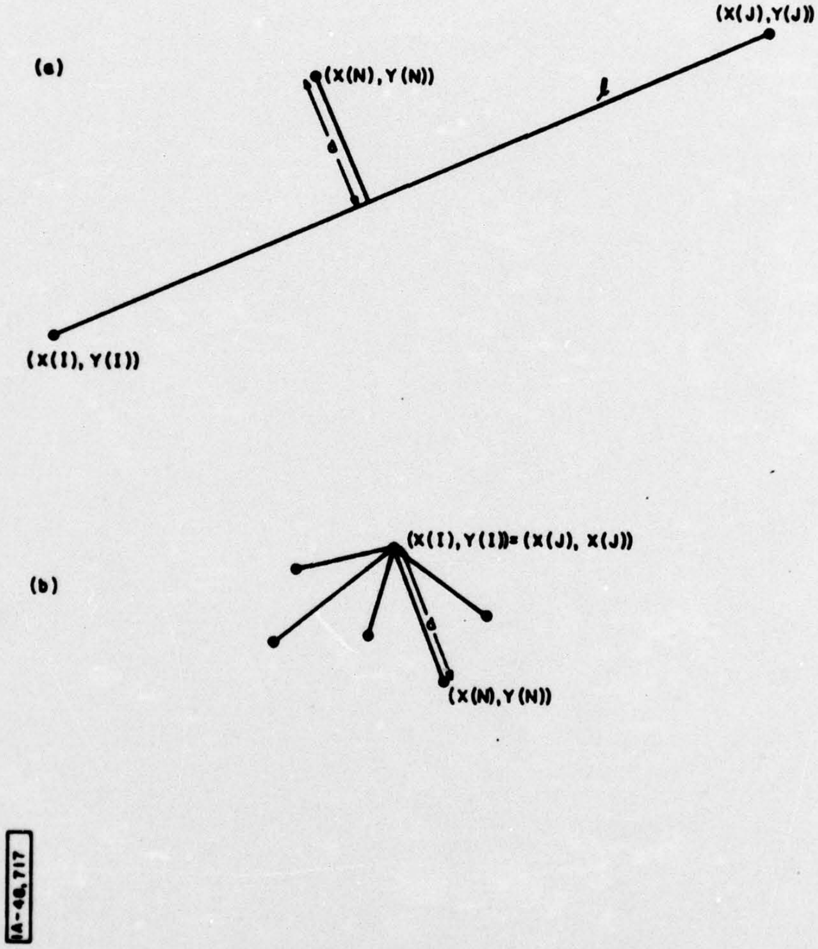



Figure 19 DEVIATION CALCULATION

**Common Blocks**

None

**Subroutines**

None

```
      SUBROUTINE CDLEV(X,Y,I,J,L,P,MD,TL)
      REAL MD,TL,MDEV,X(1),Y(1)
      REAL XBAR,YBAR,TD,Z
      REAL XNJ,YNJ
      INTEGER P,L(1),TEMP,I,J
      INTEGER K,K2
      XBAR=X(I)-X(J)
      YBAR =Y(I)-Y(J)
      TL=XBAR*XBAR+YBAR*YBAR
      Z=X(I)*Y(J)-X(J)*Y(I)
      K=I+1
      K2=J-1
      MDEV=0.0
      P=-1
      DO 200 N=K,K2
C
C CHECK IF CHAIN FORMS AN ISLAND
C
      IF(TL.NE.0.0) GO  TO 25
      XNJ=X(N)-X(J)
      YNJ=Y(N)-Y(J)
      TD=XNJ*XNJ+YNJ*YNJ
      GO TO 30
25    TD=ABS(X(N)*YBAR-Y(N)*XBAR+Z)
30    IF(TD.LT.MDEV) GO TO 200
      IF (TD.EQ.MDEV) GO TO 150
C
C NEW MAX DEVIATION
C
50    IF(P.EQ.-1) GO TO 100
      TEMP=L(P)
      L(P)=-1
      P=TEMP
      GO TO 50
100   MDEV=TD
      P=N
      GO TO 200
```

```
C
C TD=MDEV
C
150     L(N)=P
        P=N
200     CONTINUE
        IF(TL.NE.0.0) GO TO 250
        MD=MDEV**0.5
        RETURN
250     TL=(TL)**0.5
        MD=MDEV/TL
        RETURN
        END
K CDLEV   0024
P CDLEV   03C4
L .Q      0000
L .P      0000

A 380     16E4
A 926     174A
A 930     17EC
A 390     18F6
A 400     1864
A 410     1948
A 420     1982
A 430     1A1C
A 440     1A78
A 450     1B0C
L .V      0000
```

EXCLUD

Purpose

EXCLUD deletes entire chains from the data base. It is used primarily for removing islands which are of insignificant size for the scale of the display.

Procedure Description

After an array of numbers identifying those chains to be deleted in order of their appearance in the data base has been read in, the search for the chains begins. The data is read into core in 6400 byte blocks and is examined in 4-word sections. Each section is identified as a header, a data point entry or end-of-chain. If it is a header the chain number is checked against the first chain number in the array which has not been found in the data base up to that time. If a match is not made the chain is copied to the output buffer. The output buffer is the same size as the input buffer and so the data remains blocked at 1600 words.

Common Blocks

None

Subroutines

None

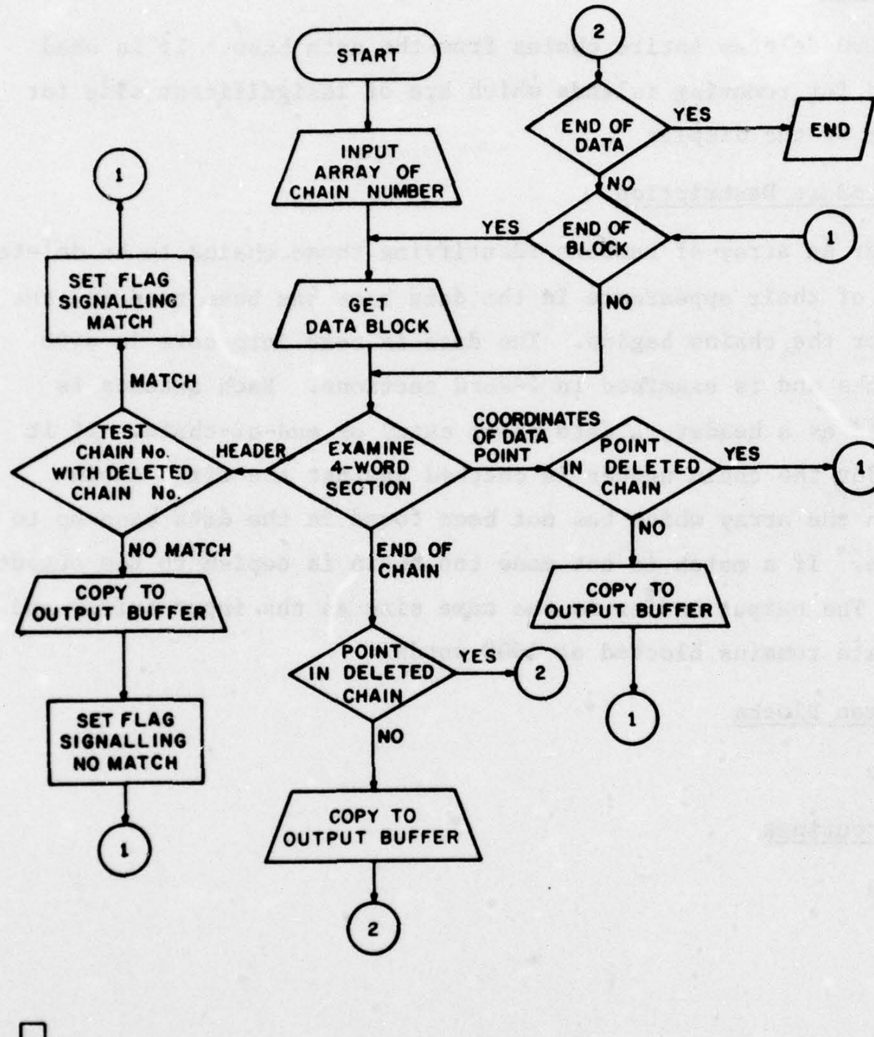



Figure 20 EXCLUD FLOWCHART

```
      DIMENSION B(1600),IB(2),LINK(100),OB(1600),IOB(2),ISAV(4)
      EQUIVALENCE (B(1),IB(1)),(OB(1),IOB(1))
      WRITE(5,998)
      READ(5,902) IN
      WRITE(5,997)
      READ(5,902) IOUT
      WRITE(5,999)
      L=1
    1 READ(5,901) LINK(L)
      IF(LINK(L).EQ.0) GO TO 10
      L=L+1
      GO TO 1
   10 L=1
      NL=LINK(1)
      I=1
      IHEAD=1
   20 READ(IN) B
      DO 100 M=1,1600,4
      GO TO (201,202,203,204),IHEAD
  201 IF(IB(M).GT.1000) GO TO 500
      IHEAD=2
      ISAV(1)=IB(M)
      ISAV(2)=IB(M+1)
      ISAV(3)=IB(M+2)
      ISAV(4)=IB(M+3)
      GO TO 100
  202 IHEAD=4
      IF(NL.EQ.IB(M+1)) GO TO 205
      IHEAD=3
      IOB(I)=ISAV(1)
      IOB(I+1)=ISAV(2)
      IOB(I+2)=ISAV(3)
      IOB(I+3)=ISAV(4)
      I=I+4
      IF(I.LT.1598) GO TO 101
      I=1
      WRITE(IOUT) OB
  101 IOB(I)=IB(M)
      IOB(I+1)=IB(M+1)
  102 IOB(I+2)=IB(M+2)
      IOB(I+3)=IB(M+3)
      I=I+4
      IF(I.LT.1598) GO TO 100
      I=1
      WRITE(IOUT) OB
      GO TO 100
```

```
203 OB(I)=B(M)
    OB(I+1)=B(M+1)
    IF(B(M).EQ.1000.) IHEAD=1
    GO TO 102
205 L=L+1
    NL=LINK(L)
    GO TO 100
204 IF(B(M).EQ.1000.) IHEAD=1

  100 CONTINUE
      GO TO 20
  500 OB(I)=B(M)
      OB(I+1)=B(M+1)
      WRITE(IOUT) OB
      STOP
  901 FORMAT(I3)
  902 FORMAT(I2)
  997 FORMAT(30HENTER OUTPUT DEVICE NUMBER(NN))
  998 FORMAT(29HENTER INPUT DEVICE NUMBER(NN))
  999 FORMAT(35HENTER CHAINS TO BE ELIMINATED(NNN),/,
     1 24HENTER ZERO AS LAST CHAIN)
      END
I  .U      0000
E  B       058C
E  IB      058C
E  LINK    1E8C
E  OB      201C
E  IOB     201C
E  ISAV    391C
A  998     0518
I  *I      0200
A  902     04E8
E  IN      392C
A  997     04F0
E  IOUT    3930
A  999     053E
E  L       3934
A  1       0080
A  901     04E0
A  10      00DF
E  NL      3940
E  I       3944
E  IHEAD   3948
A  20      0142
L  *J      0000
A  100     045E
E  M       394C
```

```
A 201       013C
A 202       010B
A 203       03A6
A 204       043C
A 500       047A
A 205       041A
A 101       028C
A 102       030A
I  .S       0000
I  .V       0000

PROGRAMS:
  B968 @J          BA48 .U          BA76 .V          BA84 @I
  CAB0 .S          CAE4 .MES        CB62

ENTRY-POINTS:
  B968 @J          BA54 .U          BA7A .V          BA84 @I
  CAB0 .S          CAE4 .MES

COMMON-BLOCKS:
NONE

UNDEFINED:
NONE
LOADER
```

ISLAND

Purpose

ISLAND performs two calculations on the chains in a data base which have coincident endpoints. It calculates the horizontal and vertical extent of the chains and determines their separation from nearby ones.

Procedure Description

The calculations are performed in two passes of the data base. The data is handled in the same manner for both passes. Data is read in 6400-byte blocks and 4-word sections are examined and identified as header, data point or end of chain entry. In the first pass an array is created containing the minimum and maximum x and y coordinates of each chain with coincident endpoints (i.e., 4 pairs of coordinates/ chain). From these figures the horizontal and vertical extent of the island are computed. In the second pass the coordinates of each point in the data base are subjected to the following tests to determine chain separation (refer to Figure 21):

(1) Vertical Test

The y-coordinate is tested for its proximity to each island; that is, is $YMIN \geq y \geq YMAX$?

For each island in the y-range the minimum x-coordinate and its y-value are compared to the coordinates of the point. If the x-coordinate is within a certain distance of the x-minimum of the island and similarly for the y-coordinate, a message is sent to the printer.

The minimum separation in the horizontal direction is 4/256 and in the vertical direction, 4/240. These values were empirically chosen with respect to the resolution of the display medium as the minimum distance required for distinct representation of two lines.

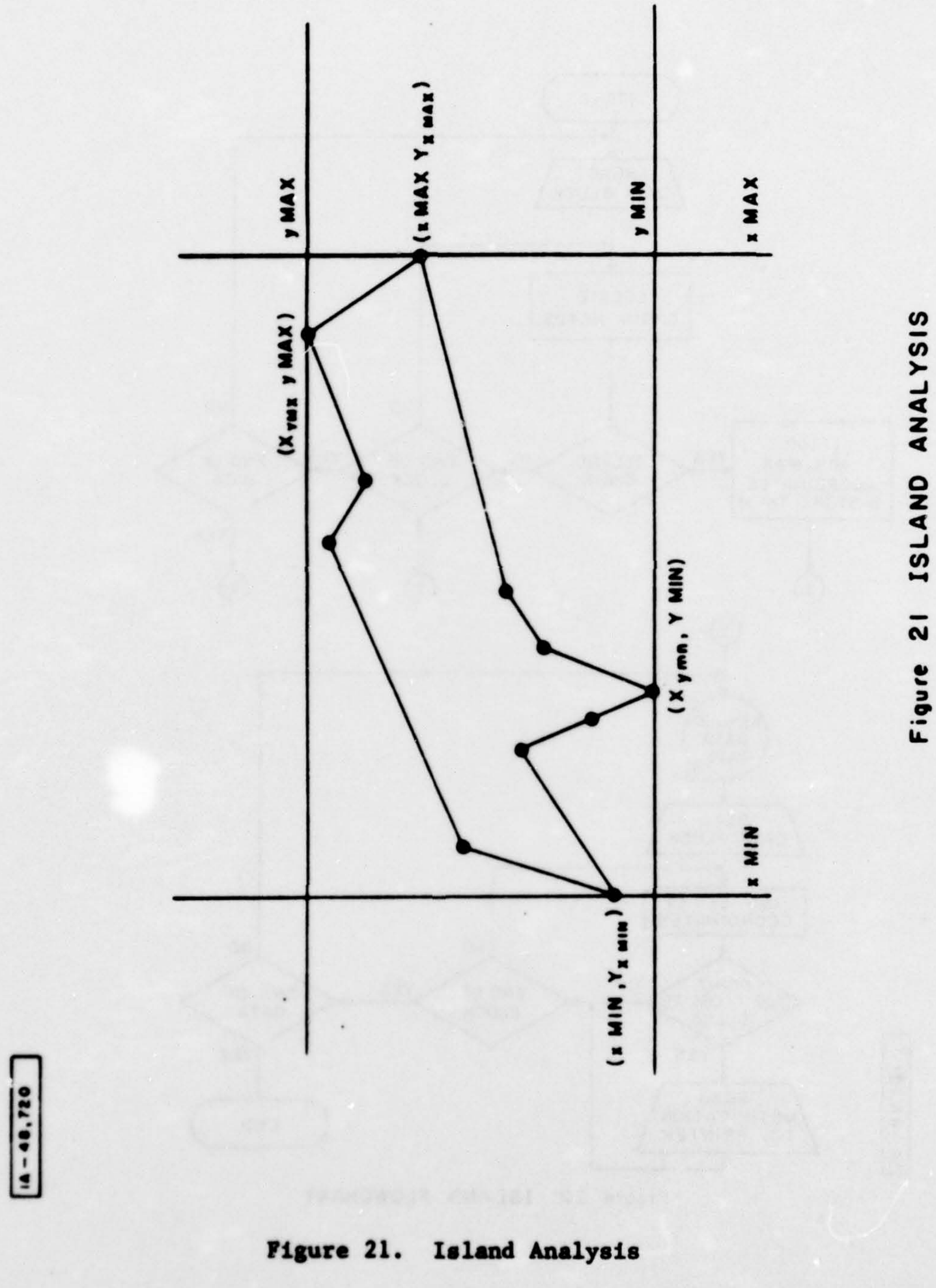



**Figure 21. Island Analysis**

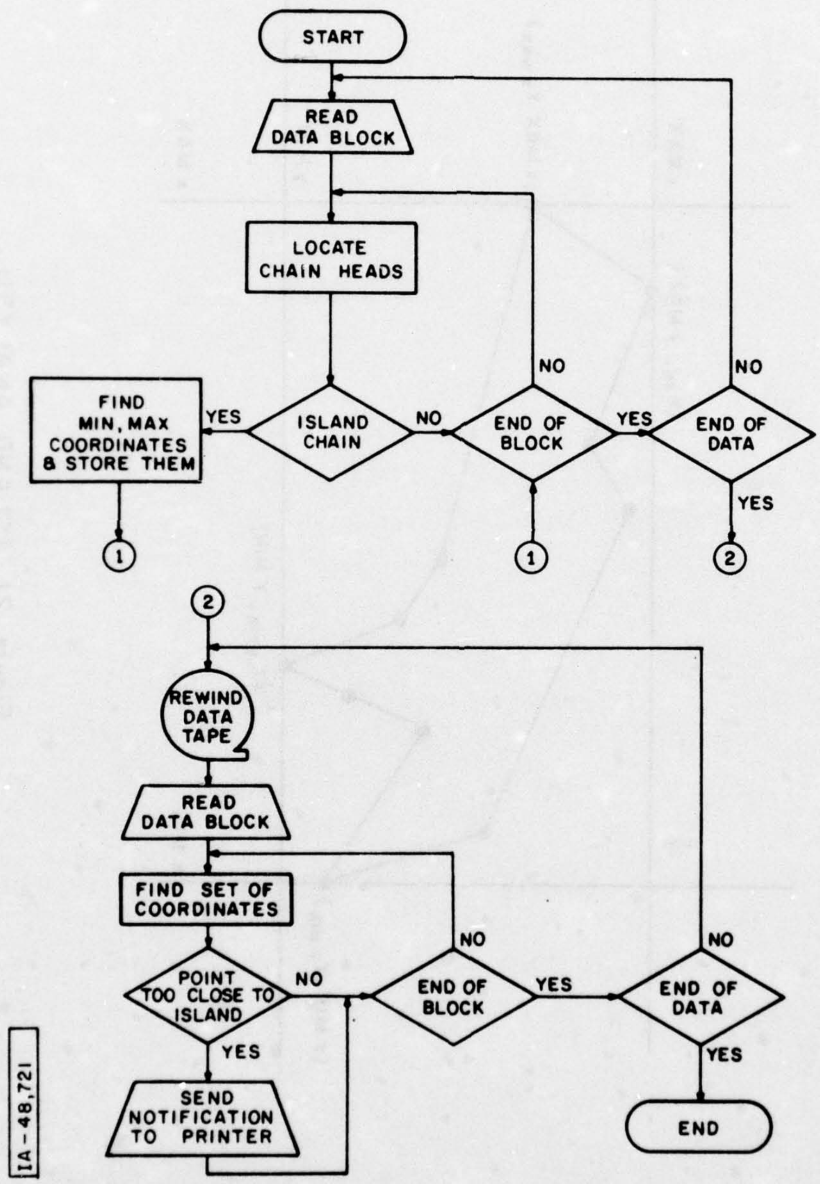



Figure 22: ISLAND FLOWCHART

Thus if

$|XMIN - x| \leq 4/256$ and

$|Y_{XMIN} - y| \leq 4/240$ a message is printed

The same test is performed with the island x-maximum and its y-value.

That is

$|XMAX - x| \leq 4/256$ and

$|Y_{XMAX} - y| \leq 4/240$

(2) Horizontal Test

The x-coordinate is also tested for its proximity to each island, that is, is $XMIN \geq x \geq XMAX$?

If so, the minimum y-coordinate and its corresponding x-value are compared to the coordinates of the point.

If

$|YMIN - y| \leq 4/240$ and

$|X_{YMIN} - x| \leq 4/256$ a message is sent to the printer.

The maximum y and its x-value are also checked;

$|YMAX - y| \leq 4/240$ and

$|X_{YMAX} - x| \leq 4/256$

For each point fulfilling either set of conditions, the chain numbers of the island and the chain to which the point belongs, the coordinates of the island and nearby chain points, and vertical and/ or horizontal separation(s) are listed.

Common Blocks

None

Subroutines

None

```
      DIMENSION B(1600),IB(1600)
      DIMENSION TEST(820),ITEST(820)
      EQUIVALENCE(IB(1),B(1))
      EQUIVALENCE (TEST(1),ITEST(1))
      WRITE(3,998)
      IHEAD=1
      L=8224
      K=1
  100 READ(6) B
      DO 1 I=1,1600,4
      GO TO (10,20,30,40),IHEAD
   10 LN1=IB(I)
      IF(LN1.GT.1000) GO TO 200
      LN2=IB(I+1)
      IORIG=IB(I+2)
      IHEAD=2
      GO TO 1
   20 ISEQ=IB(I+1)
      IOSEQ=IB(I)
      IHEAD=3
      GO TO 1
   30 XSTART=B(I)
      YSTART=B(I+1)
      XMAX=XSTART
      XMIN=XSTART
      XOYMAX=XSTART
      XOYMIN=XSTART
      YMAX=YSTART
      YMIN=YSTART
      YOXMAX=YSTART
      YOXMIN=YSTART
      IHEAD=4
      GO TO 1
   40 IF(B(I).EQ.1000.) GO TO 50
      XEND=B(I)
      YEND=B(I+1)
      XMAX=AMAX1(XMAX,XEND)
      IF(XMAX.EQ.XEND) YOXMAX=YEND
      YMAX=AMAX1(YMAX,YEND)
      IF(YMAX.EQ.YEND) XOYMAX=XEND
      XMIN=AMIN1(XMIN,XEND)
      IF(XMIN.EQ.XEND) YOXMIN=YEND
      YMIN=AMIN1(YMIN,YEND)
      IF(YMIN.EQ.YEND) XOYMIN=XEND
      GO TO 1
   50 IF((XSTART.EQ.XEND).AND.(YSTART.EQ.YEND)) L=10784
      WRITE(3,999) LN1,LN2,IORIG,IOSEQ,ISEQ,XSTART,YSTART,XEND,YEND,L
      IF(L.EQ.8224) GO TO 60
      WRITE(3,997) XMIN,YOXMIN
      WRITE(3,996) XMAX,YOXMAX
      WRITE(3,995) YMIN,XOYMIN
      WRITE(3,994) YMAX,XOYMAX
      DELTAX=XMAX-XMIN
      DELTAY=YMAX-YMIN
```

```
      WRITE(3,993) DELTAX,DELTAY
      L=8224
      ITEST(K)=ISEQ
      TEST(K+1)= XMAX
      TEST(K+2)=YOXMAX
      TEST(K+3)= XMIN
      TEST(K+4)=YOXMIN
      TEST(K+5)= YMAX
      TEST(K+6)=XOYMAX
      TEST(K+7)=YMIN
      TEST(K+8)=XOYMIN
      K=K+9
60    IHEAD=1
    1 CONTINUE
      GO TO 100
C
C AT THIS POINT WE HAVE AN ARRAY, TEST (ITEST), CONTAINING
C THE MIN AND MAX X AND Y COORDINATES OF ALL ISLANDS IN THE DATA BASE
C
200   CALL REW(6)
      IHEAD=1
      CONST=4./240.
      CONSTX=4./256.
      WRITE(3,990)
300   READ(6) B
      DO 2 I=1,1600,4
      GO TO (310,320,330,340), IHEAD
310   IF(IB(I).GT.1000) GO TO 400
      IHEAD =2
      GO TO 2
320   ISEQ=IB(I+1)
      IHEAD=3
      GO TO 2
330   IHEAD=4
      GO TO 342
340   IF(B(I).NE.1000.0) GO TO 342
      IHEAD=1
      GO TO 2
342   X=B(I)
      Y=B(I+1)
C
C TESTX
C
      DO 347 K=6,630,9
      IF(ITEST(K-5).EQ.ISEQ) GO TO 347
      IF((TEST(K).LT.Y).OR.(TEST(K+2).GT.Y)) GO TO 344
      IF(((TEST(K-3)+CONST).LT.Y).OR.((TEST(K-3)-CONST).GT.Y))
     CGO TO 343
      XDIFF=ABS(TEST(K-4)-X)
      IF (XDIFF.GT.0.01) GO TO 343
      WRITE (3,991)ITEST(K-5),TEST(K-4),TEST(K-3),ISEQ,X,Y,XDIFF
343   IF(((TEST(K-1)+CONST).LT.Y).OR.((TEST(K-1)-CONST).GT.Y))
     CGO TO 344
      XDIFF=ABS(TEST(K-2)-X)
```

```
      IF (XDIFF.GT.0.01) GO TO 344
      WRITE(3,991)ITEST(K-5),TEST(K-2),TEST(K-1),ISEQ,X,Y,XDIFF
C
C TEST Y
C
344   IF((TEST(K-4).LT.X).OR.(TEST(K-2).GT.X))GO TO 347
      IF(((TEST(K+1)+CONSTX).LT.X).OR.((TEST(K+1)-CONSTX).GT.X))
     1GO TO 345
      YDIFF=ABS(TEST(K)-Y)
      IF(YDIFF.GT.0.01) GO TO 345
      WRITE(3,992)ITEST(K-5),TEST(K+1),TEST(K),ISEQ,X,Y,YDIFF
345   IF(((TEST(K+3)+CONSTX).LT.X).OR.((TEST(K+3)-CONSTX).GT.X))
     1GO TO 347
      YDIFF=ABS(TEST(K+2)-Y)
      IF(YDIFF.GT.0.01) GO TO 347
      WRITE(3,992)ITEST(K-5),TEST(K+3),TEST(K+2),ISEQ,X,Y,YDIFF
347   CONTINUE
2     CONTINUE
      GO TO 300
400   STOP
990   FORMAT(6HIS.LSN,2X,8HIS.(X,Y),8X,7HM.L.LSN,2X,9HM.L.(X,Y),3X,
     C7HDELTA X,3X,7HDELTA Y)
991   FORMAT(2X,I3,2F8.4,3X,I3,3X,3F8.4)
992   FORMAT(2X,I3,2F8.4,3X,I3,3X,2F8.4,9X,F8.4)
993   FORMAT(6X,11H DELTA X = ,F8.4,11H DELTA Y = ,F8.4//)
994   FORMAT(6X,10H MAX Y IS ,F8.4,8H AT X = ,F8.4)
995   FORMAT(6X,10H MIN Y IS ,F8.4,8H AT X = ,F8.4)
996   FORMAT(6X,10H MAX X IS ,F8.4,8H AT Y = ,F8.4)
997   FORMAT(6X,10H MIN X IS ,F8.4,8H AT Y = ,F8.4)
  998 FORMAT(3X,6HOLD ID,5X,6HSOURCE,3X,11HORIGIN SEQ.,3X,
     1 10HTOTAL SEQ.,5X,6HXSTART,5X,6HYSTART,6X,4HXEND,7X,4HYEND)
  999 FORMAT(3X,I3,I4,3X,I5,6X,I5,8X,I5,5X, 4 (3X,F8.4),2X,A1)
      END
L .U        0000
E B         0D92
E IB        0D92
E TEST      2692
E ITEST     2692
A 998       0CEE
L *I        0040
E IHEAD     3362
E L         336A
E K         3372
A 100       0030
I *J        0000
A 1         04CA
E I         3376
A 10        006A
A 20        0000
A 30        010A
A 40        0184
E LN1       337A
A 200       04F0
E LN2       3382
```

```
E IORIG  3386
E ISEQ   338E
E IOSEQ  3392
E XSTART 339A
E YSTART 339E
E XMAX   33A2
E XMIN   33A6
E XOYMAX 33AA
E XOYMIN 33AE
E YMAX   33B2
E YMIN   33B6
E YOXMAX 33BA
E YOXMIN 33BE
A 50     0264
E XEND   33CA
E YEND   33CE
L AMAX1  0000
L AMIN1  0000
A 999    0D5A
A 60     04C2
A 997    0CC2
A 996    0C96
A 995    0C6A
A 994    0C3E
E DELTAX 33D6
E DELTAY 33DA
A 993    0C0C
L REW    0000
E CONST  33E6
E CONSTX 33F2
A 990    0B72
A 300    051C
A 2      0B56
A 310    0556
A 320    0580
A 330    05A6
A 340    05B2
A 400    0B6C
A 342    050C
E X      33FA
E Y      33FE
A 347    0B44
A 344    08C0
A 343    079A
E XDIFF  3402
L ABS    0000
A 991    0BC4
A 345    0A1E
E YDIFF  3412
A 992    0BE4
L .S     0000
L .V     0000
```

```
            PROGRAMS:
  8416 @J        84F6 .U        8524 .V        8532 @I
  955E AMAX1     9570 .2        9582 $1        9598 $3
  95CA ABS       95E6 .COMP     960C .RARG     963F REW
  965E .S        9692 .MES      9710 AMIN1     9722

ENTRY-POINTS:
  8416 @J        8502 .U        8528 .V        8532 @I
  955E AMAX1     9570 .2        9582 $1        9598 $3
  95CA ABS       95E6 .COMP     960C .RARG     963E REW
  965E .S        9692 .MES      9710 AMIN1

COMMON-BLOCKS:
NONE

UNDEFINED:
NONE
```

ENDPT

Purpose

ENDPT lists the chain numbers of the chains in a data base and the coordinates of the first and last point in each. It is the most general tool in the package for checking data integrity.

Procedure Description

Data is read into core in 6400-byte blocks. Four-word segments are identified as header, data point or end of chain entries. If a header is being processed, the WDBI line segment number, chain number and type code are saved. While examining the data points the coordinates of the first and last are saved. When an end of chain is detected the stored information about the chain is printed.

Common Blocks

None

Subroutines

None

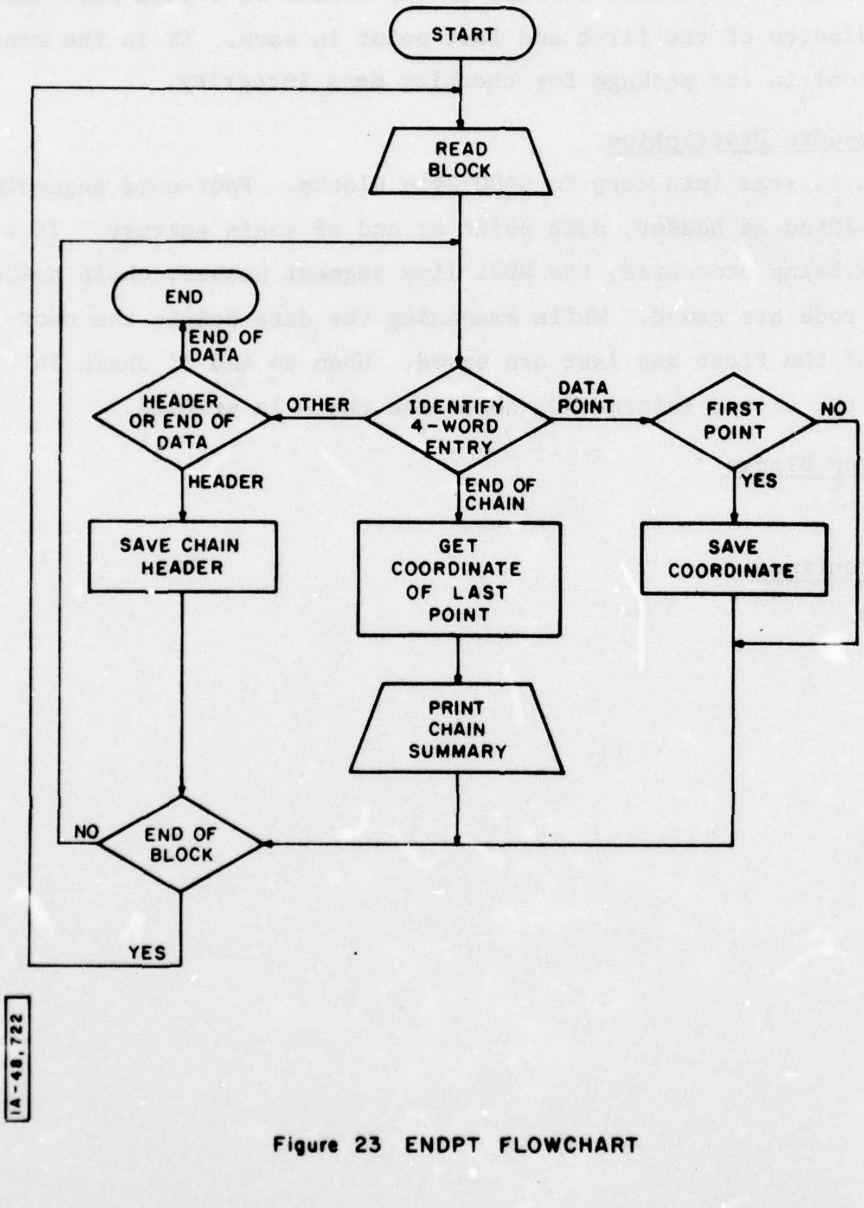



Figure 23 ENDPT FLOWCHART

```
      DIMENSION B(1600),IB(1600)
      EQUIVALENCE(IB(1),B(1))
      WRITE(3,998)
      IHEAD=1
      L=8224
      K=1
  100 READ(6) B
      DO 1 I=1,1600,4
      GO TO (10,20,30,40),IHEAD
   10 LN1=IB(I)
      IF(LN1.GT.1000) GO TO 200
      LN2=IB(I+1)
      IORIG=IB(I+2)
      IHEAD=2
      GO TO 1
   20 ISEQ=IB(I+1)
      IOSEQ=IB(I)
      IHEAD=3
      GO TO 1
   30 XSTART=B(I)
      YSTART=B(I+1)
      IHEAD=4
      GO TO 1
   40 IF(B(I).EQ.1000.) GO TO 50
      XEND=B(I)
      YEND=B(I+1)
      GO TO 1
   50 IF((XSTART.EQ.XEND).AND.(YSTART.EQ.YEND)) L=10784
      WRITE(3,999) LN1,LN2,IORIG,IOSEQ,ISEQ,XSTART,YSTART,XEND,YEND,L
      IF(L.EQ.8224) GO TO 60
      L=8224
60    IHEAD=1
    1 CONTINUE
      GO TO 100
200   WRITE(5,201)
201   FORMAT(5H STOP)
  998 FORMAT(3X,6HOLD ID,5X,6HSOURCE,3X,11HORIGIN SEQ.,3X,
     1 10HTOTAL SEQ.,5X,6HXSTART,5X,6HYSTART,6X,4HXEND,7X,4HYEND)
  999 FORMAT(3X,I3,I4,3X,I5,6X,I5,8X,I5,5X, 4 (3X,F8.4),2X,A1)
      END
L .U      0000
E B       031A
E IB      031A
A 998     0276
L #I      0000
E IHEAD   1C1A
E L       1C22
E K       1C2A
A 100     0030
L #J      0000
A 1       023E
E I       1C2E
A 10      006A
A 20      00D0
```

```
A 30       010A
A 40       0144
E LN1      1C32
A 200      0254
E LN2      1C3A
E IORIG    1C3E
E ISEQ     1C46
E IOSEQ    1C4A
E XSTART   1C52
E YSTART   1C56
A 50       0194
E XEND     1C62
E YEND     1C66
A 999      02E2
A 60       0236
A 201      0268
I .V       0000


          PROGRAMS:
   6C6E *J          6D4E .U          6D7C .V          6D8A *I
   7DB6

ENTRY-POINTS:
   6C6E *J          6D5A .U          6D80 .V          6D8A *I


COMMON-BLOCKS:
NONE

UNDEFINED:
NONE
```

PRINTM

Purpose

PRINTM produces a map of a data base at a user-specified level of detail on the printer. It is a crude tool for checking the contents of the data base.

Procedure Description

Through a subroutine call to INITMP a 60 x 100 matrix is initialized with blanks. Data is processed in 6400-byte blocks and 4-word sections are identified as header, data point or end-of-chain entries. If a data point of rank greater than or equal to the user-specified minimum is being examined, the coordinates of the point undergo a linear transformation (see Figure 24) producing a point in the range of the matrix indices. A real to integer conversion is performed and an 'X' is placed in the cell of the matrix corresponding to the transformed and converted point coordinates. After all the data has been processed the matrix is printed.

Common Blocks

None

Subroutines

INITMP - initializes a 60 x 100 matrix

CALL INITMP (MAP, LX, LY)

MAP: 60 x 100

LX: defined as 'X' in the subroutine

LY: defined as '♥' in the subroutine

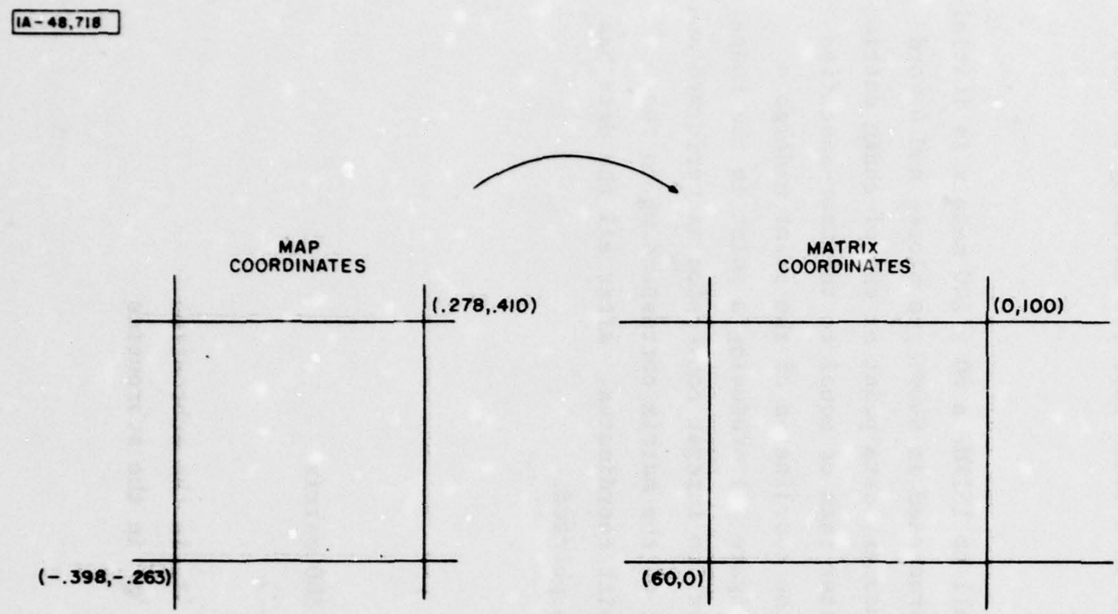



Figure 24 MAP ⟶ MATRIX COORDINATE TRANSFORMATION

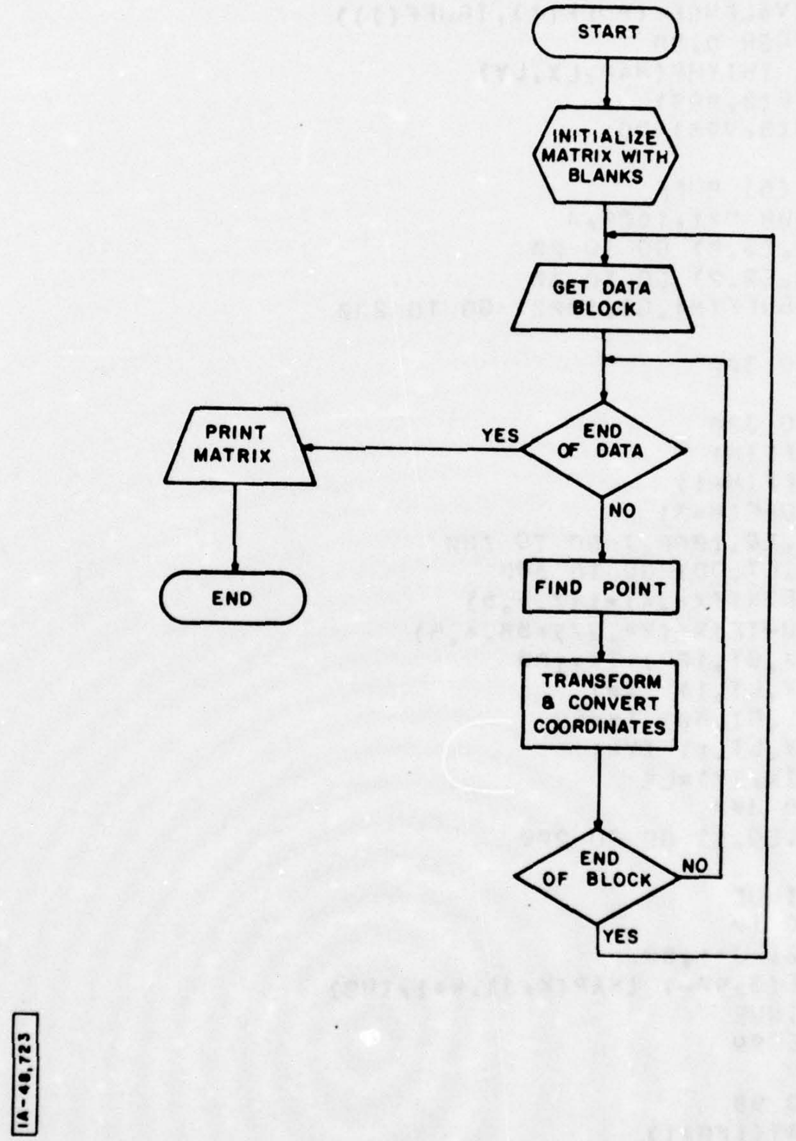



Figure 25 PRINTM FLOW CHART

```
      DIMENSION MAP(100,60),BUFF(1600)
      DIMENSION IBUFF(1600)
      EQUIVALENCE (BUFF(1),IBUFF(1))
      INTEGER D,DD
      CALL INITMP(MAP,LX,LY)
      WRITE(5,999)
      READ(5,998) DD
   98 N=1
   10 READ(6) BUFF
      DO 300 M=1,1600,4
      IF(N.EQ.0) GO TO 20
      IF(N.EQ.2) GO TO 30
      IF(IBUFF(M).GT.1000) GO TO 200
      N=2
      GO TO 300
   30 N=0
      GO TO 300
   20 X=BUFF(M)
      Y=BUFF(M+1)
      D=IBUFF(M+3)
      IF(X.EQ.1000.) GO TO 100
      IF(D.LT.DD) GO TO 300
      IX=IFIX((X+.4)*147.+.5)
      IY=60-IFIX((Y+.27)*88.+.5)
      IF(IX.GT.100) IX=100
      IF(IX.LT.1) IX=1
      IF(IY.GT.60) IY=60
      IF(IY.LT.1) IY=1
      MAP(IX,IY)=LX
      GO TO 300
  100 IF(N.EQ.1) GO TO 200
      N=1
  300 CONTINUE
      GO TO 10
  200 DO 101 J=1,60
      WRITE(3,900) (MAP(K,J),K=1,100)
  101 CONTINUE
      PAUSE 99
      LX=LY
      GO TO 98
  900 FORMAT(100A1)
  950 FORMAT(1H ,2I10)
  998 FORMAT(I2)
  999 FORMAT(22HENTER DETAIL LEVEL(NN))
      END
```

```
L  .U      0000
E  MAP     02F0
E  BUFF    60B0
E  IBUFF   60B0
E  D       79B0
E  DD      79B4
L  INITMP  0000
E  LX      79B8
E  LY      79BC
A  999     02CC
L  @I      0000
A  998     02C4
A  98      0040
E  N       79C0
A  10      0048
L  @J      0000
A  300     0210
E  M       79C8
A  20      00C4
A  30      00B8
A  200     0226
E  X       79D8
E  Y       79DC
A  100     01F6
E  IX      79E4
L  IFIX    0000
E  IY      79FC
A  101     0282
E  J       7A10
A  900     02AA
E  K       7A14
L  .H      0000
A  950     02B6
L  .V      0000
```

```
          PROGRAMS:
   CA62 @J         CB42 .U         CB70 .V         CB7E @I
   DBAA IFIX       DBCA .Y         DC4E .RARG      DC80 .H
   DCC2 .O         DD0A .MES       DD88

ENTRY-POINTS:
   CA20 INITMP     CA62 @J         CB4E .U         CB74 .V
   CB7E @I         DBAA IFIX       DBCA .Y         DC4E .RARG
   DC80 .H         DCC2 .O         DD0A .MES
```

COMMON-BLOCKS:
NONE

UNDEFINED:
NONE

**Subroutine INITMP**

**Purpose**

**INITMP is an assembly routine that initializes the map matrix and sets variables LX to 'X' and LY to '∇'. These variables are used by PRINTM to identify those cells of the matrix corresponding to the coordinates of map data points.**

```
0004R               END
* INITMP  0008R
  LOOP    0018R
  SAVR    0000R

0000R                      ENTRY  INITMP
0000R             SAVR     DS     8
0008R D0C0        INITMP   STM    12,SAVR
      0000R
000CR D1EF                 LM     14,2(15)
      0002
0010R C8D0                 LHI    13,6000
      1770
0014R C8C0                 LHI    12,C'  '
      2020
0018R 40CE        LOOP     STH    12,0(14)
      0000
001CR CAE0                 AHI    14,4
      0004
0020R CBD0                 SHI    13,1
      0001
0024R 4220                 BP     LOOP
      0018R
0028R C8C0                 LHI    12,C'X '
      5820
002CR 40CF                 STH    12,0(15)
      0000
0030R D1C0                 LM     12,SAVR
      0000R
0034R C8C0                 LHI    12,C'0 '
      5F20
0038R 48EF                 LH     14,6(15)
      0006
003CR 40CE                 STH    12,0(14)
      0000
0040R D1C0                 LM     12,SAVR
      0000R
0044R 4AFF                 AH     15,0(15)
      0000
0048R 030F                 BR     15
004AR                      END
```

CMAP

Purpose

CMAP creates an image in a drum file of all the map points with rank equal to or greater than a user-specified level through subroutine calls to PALLET.

Procedure Description

Data is read into core in 6400-byte blocks. The rank of each data point is checked and if it is above the user-specified minimum, the x- and y- coordinates of the point are stored in X and Y arrays. Upon completion of the processing of a chain, the contents of these arrays are written on the drum file as an item (i.e., line) of the map image. If a chain has more than 200 elements of sufficiently high rank to be kept in the image, more than one line is created for that chain.

Common Blocks

| Block Name | Contents | Description of Contents |
| --- | --- | --- |
| RG | BUFF | input buffer |

Subroutines

GET: fills input buffer

no parameters

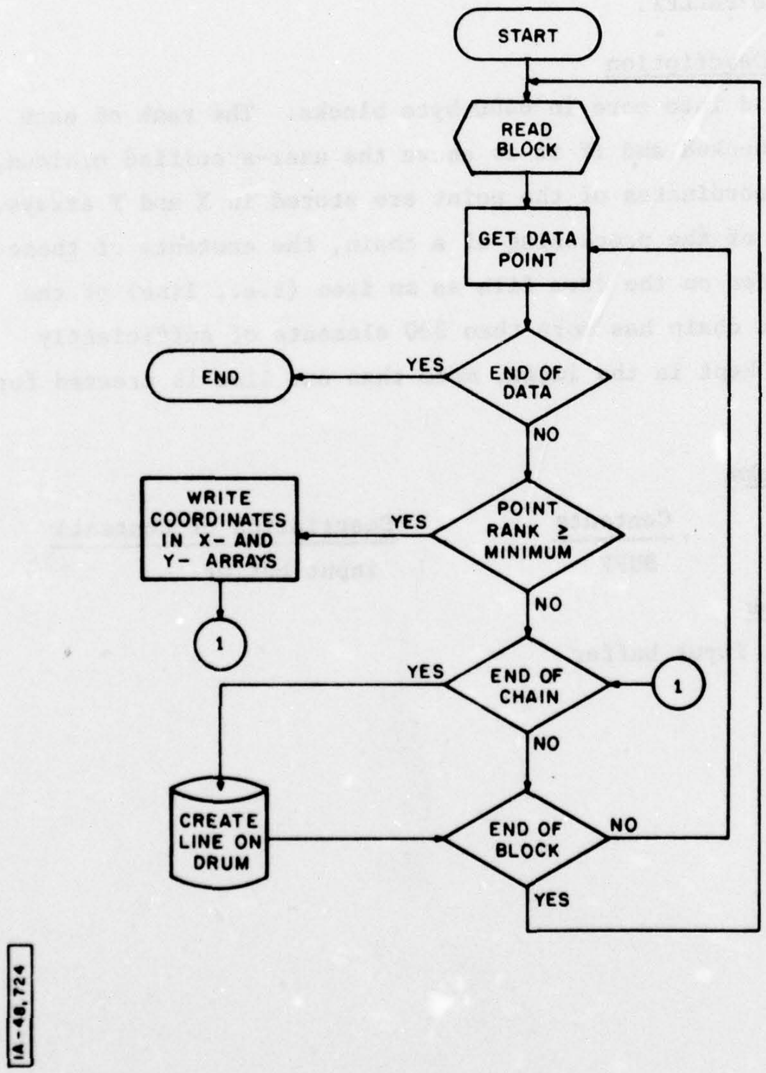



Figure 26: CMAP FLOW CHART

```
      DIMENSION X(800),Y(800),NAME(2)
      DIMENSION IBUFF(1600),BUFF(1600)
      EQUIVALENCE (IBUFF(1),BUFF(1))
      EXTERNAL WRKSP,SAVEA
      INTEGER SAVEA,ERROR
      INTEGER D,DD
C K=BEGINNING OF CURRENT STRING
C J=LAST ENTRY OF CURRENT STRING
C M= CURRENT BUFF ENTRY
C NUM= NUMBER OF SUBIMAGES
      WRITE(5,101)
      READ(5,100) NAME
      WRITE(5,998)
      READ(5,999) D
      M=2000
      NUM=0
      IHEAD=1
      I=1
      CALL SETSAV(WRKSP,SAVEA)
      CALL INTORM(25,128,0,0,0,ERROR)
      IF (ERROR.NE.0) STOP 10
      CALL INTFMP(25,1,SAVEA)
      CALL DEFILE(WRKSP,8HREADFILE,SAVEA,1)
      CALL OPEN(WRKSP,8HREADFILE,SAVEA)
   10 NAME(2)=NAME(2)+1
      NUM=NUM+1
      CALL OPENI(NAME,-.4,-.27,.28,.41,0)
      N=40
   16 M=M+4
      IF(M.GT.1598) CALL GET
      IF(IHEAD.EQ.0) GO TO 17
      IF (IHEAD.EQ.2) GO TO 18
      IF(BUFF(M).EQ.1000.) GO TO 200
18    IHEAD=2
      GO TO 16
      IHEAD=0
      GO TO 16
   17 X(I)=BUFF(M)
      Y(I)=BUFF(M+1)
      Y(I)=.410-Y(I)
      IF(X(I).EQ.1000.) GO TO 30
      DD=IBUFF(M+3)
      IF(D.GT.DD) GO TO 16
      N=N+8
      IF(N.GT.1700) GO TO 21
      I=I+1
      GO TO 16
21    IF(I.GT.1)CALL LINES(8HLINE    ,X(1),Y(1),I,7)
      CALL CLOSEI(NAME)
      X(1)=X(I)
      Y(1)=Y(I)
      I=2
      GO TO 10
   30 I=I-1
```

```
      CALL LINES(8HLINE     ,X(1),Y(1),I,7)
      IHEAD=1
      N=N+12
      I=1
      GO TO 16
  200 CALL CLOSEI(NAME)
      NAME(2)=NAME(2)-NUM+1
      CALL OPENI(8HMAP      ,0.,0.,511.,479.,0)
      DO 201 I=1,NUM
      CALL INCLUD(NAME,-346.,-332.,678.,1108.,0,0.,0.,0.,NAME)
  201 NAME(2)=NAME(2)+1
      CALL CLOSEI(8HMAP      )
      CALL OPENI(8HWORLD    ,0.,0.,511.,479.,0)
      CALL INCLUD(8HMAP      ,0.,0.,511.,479.,0,0.,0.,0.,8HMAP      )
      CALL CLOSEI(8HWORLD   )
      STOP
  100 FORMAT(2A4)
  101 FORMAT(11H ENTER NAME)
  998 FORMAT(23H ENTER DETAIL LEVEL(NN)  )
  999 FORMAT(I2)
      END
L  ,U       0000
E  X        04F2
E  Y        1172
E  NAME     1DF2
E  IBUFF    1DFA
E  BUFF     1DFA
L  WRKSP    0000
L  SAVEA    0000
E  ERROR    36FA
E  D        36FE
E  DD       3702
A  101      04B2
L  *I       0000
A  100      04A8
A  998      04C6
A  999      04E6
E  M        3706
E  NUM      370E
E  IHEAD    3716
E  I        371E
L  SETSAV   0000
L  INTORM   0000
L  ,S       0000
L  INTEMP   0000
L  DEFILE   0000
L  OPEN     0000
A  10       00F8
L  OPENI    0000
E  N        3752
A  16       0144
L  GET      0000
A  17       01C4
A  18       01AC
```

```
A 200     038C
A 30      032C
A 21      02A6
L LINES   0000
L CLOSFI  0000
A 201     0404
L INCLUD  0000
L .V      0000
```

```
      SUBROUTINE GET
      COMMON /RG/BUFF(1600),M
      READ(6) BUFF
      M=1
      RETURN
      END
I RG        1904
E BUFF      0000
E M         1900
K GET       0024
P GET       005C
L ,Q        0000
L ,P        0000
L ,J        0000
```

```
         PROGRAMS:
 65FE N63       6684 IALD4     66C8 FIND4     678C DRIVER
 6830 PICTUR    71E6 AMPCAR    722C DIS       75CC NAMING
 761A OPENI     7C6C RDHIS     7D02 REPLAC    7D68 RGERR
 7DA6 RHEAD     7E80 SETSAV    7EE0 WHEAD     7F92 WRITEB
 8138 POLL      82A8 REC       8454 XMIT      8514 AMP
 8918 TRAKUP    8E58 DISPLY    9816 ERASE     986E NEWAMP
 913E UPHIS     919C ERMSG     91C8 FIXREC    95F2 CONVT
 9688 ADDSON    97DE DEFILE    9874 GETDAT    9922 FNDSON
 9A1C INTDRM    988E INTFMP    9D0C OPEN      9DD2 PUTDAT
 9ECE REWIND    9F22 ADBRO     9F9A ADNOD     A184 FCHAIN
 A368 GTMAIN    A3DE FMPGP3    A418 HASH      A486 NAMGEN
 A50A NODOK     A560 NXTSON    A82A PAGSON    A694 FMPRD
 A716 SAVPAG    A79E SEQSON    A806 FMPTRC    A8A8 IDENT
 A980 RECIVE    AC30 RSWTCH    AC88 IMPINT    AEBA WAIT
 B150 PATCH4    B15A PATCH7    B15E MP        B336 IBUF
 B3B4 IALDU     B51C SHIPIT    B5B6 PENDNG    B6CE TRNSND
 B748 INTHND    B7C0 SEND      BA00 SEND2     BAAE DUPMSG
 BC0A ATTCHQ    BC96 COMQHB    BCC2 SQNPT     BD8A ZRO
 BD9C ,EXIT     BEBE TRACE     C0F4 TOUT1     C198 STQHB
 C1EC .U        C21A .S        C24E .J        C32E ,P
 C386 .Q        C466 .O        C4AE .MES      C52C ,V
 C53A .I        D566 COS       D58A SIN       D644 .A
 D6F6 ALOG      D7FE EXP       D904 MIN0      D916 MAX0
 D928 .1        D938 AMOD      D966 AINT      D9D8 $1
 D9EE $2        DA2C FLOAT     DA6E IFIX      DA8E .Y
 DB12 .W        DB70 .COMP     DB96 .RRARG    DBD8 $6
 DC1A .RARG     DC4C $8        DC76 .5        DC7A .ZERO
 DC7E ATAN2     DD00 ATAN      DDEA

ENTRY-POINTS:
 65BA GET       65FE GETCOR    6602 GETLNG    6604 CORPTR
 6606 GIVCOR    660A GIVLNG    660C GIVPTR    660E FLT511
 6612 FLT479    6616 F5P11     661A F4P79     661E DTOR
 6622 N63       6624 N50       6628 MESS      6678 LENGTH
 667A SUPON     667E SUPOFF    6682 ENMUX     6684 IMPTAB
 66D0 FIND      6780 DRIVER    6854 PICTUR    71EE AMPCAR
 7238 DIS       736C OUTBPR    7374 UPDIS     75D4 NAMING
 7662 IOPEN     768A OPENI     76F2 CLOSEI    7744 INCLUD
 77BC BIND      7820 BLOCK     78A0 CHAR      7948 POINTS
 79BC LINES     7C5A BUFFRP    7C5C ATTRFL    7C7A RDHIS
 7D14 REPLAC    7D88 RGERR     7DB2 RHEAD     7E5A STFACT
 7E88 SETSAV    7EC0 CHKSAV    7EEC WHEAD     7F98 WRITEB
 802A READB     8090 SAVEA     80A6 WRKSP     8138 POLL
 82A8 REC       8454 XMIT      8538 AMP       8934 TRAKUP
 8E78 DISPLY    9020 ERASE     9082 NEWAMP    914A UPHIS
 919C ERMSG     91C8 FIXREC    95FC CONVT     9688 ADDSON
 97DE DEFILE    9874 GETDAT    9922 FNDSON    9A1C INTDRM
 988E INTFMP    9C68 AOK       9C6A ONF       9C6C SAVE1
 9C8C SAVE2     9CAC SAVE3     9CCC SAVE4     9CEC FMPDSC
 9D0C OPEN      9DD2 PUTDAT    9ECE REWIND    9F22 ADBRO
 9F9A ADNOD     A184 FCHAIN    A2E8 B128      A368 GTMAIN
 A3DE FMPGP3    A3E2 FMPGP2    A3E6 FMPGP1    A3EA FMPGP0
 A418 HASH      A486 NAMGEN    A50A NODOK     A52C SAVOK
 A540 FMPZRO    A550 GHIND     A560 NXTSON    A62A PAGSON
 A694 FMPRD     A716 SAVPAG    A782 ERROR     A79E SEQSON
 A806 FMPTRC    A888 FTOFF     A89A FTON      A8A8 IDENT
 A926 IDENT2    A980 RECIVE    A9A6 LISTEN    A9CC UNCIVE
 A9F2 UNLISN    AACC ALDU2     AC30 RSWTCH    AC80 WSWTCH
 AC88 IMPINT    AD3A STMP70    AE82 BUFSIZ    AE86 BUFNUM
 AEBA WAIT      AFBC WAIT2     B150 PATCH4    B158 INTBIT
 B15A PATCH7    B15E MP        B178 MPMAIN    B30E SWITCH
```

```
B31A FRTREG    B32F TQHBNT    B336 IBUF      B384 SETBUF
B390 SHERR     B392 CNTL      B394 BUFFER    B3A4 IALDU
B466 IOUT      B50E ALDBIT    B610 TABIND    B51C SHIPIT
B5B6 PENDNG    B6CF TRNSND    B748 INTHND    B7A0 BUFINT
B7A4 INTRHB    B7C0 SEND      BA00 SEND2     BAAE DUPMSG
BC0A ATTCHQ    BC4C DETCHQ    BC96 COMQHB    BCB0 REWQHB
BCC2 SQNPT     BD5E NXTENT    BD8A ZRO       BD9C .BGIN0
BDAA .BGIN     BE44 TRCRET    BE72 .EXIT0    BE74 .EXIT
BE9E .EXITN    BEB0 .EXT      BFFE TRACE     BFAC TON
BFC0 TOFF      BFD2 RTRAC     C02E DREG      C0F4 TOUT1
C11E TOUT2     C198 STQHB     C1B6 DSECT     C1D6 SIZBUF
C1D8 MAXBUF    C1DA BUFIND    C1DC MSGMAX    C1DE BUSED
C1E0 MACHID    C1E2 TOP       C1E4 BOT       C1E6 LOGLOG
C1E8 TOUTON    C1EA TONOFF    C1F8 .U        C21A .S
C24E .J        C32E .P        C386 .Q        C466 .O
C4AE .MES      C530 .V        C53A .I        D566 COS
D58A SIN       D644 .A        D6F6 ALOG      D7FE EXP
D904 MIN0      D916 MAX0      D928 .1        D938 AMOD
D966 AINT      D9D8 $1        D9EE $2        DA2C FLOAT
DA6E IFIX      DA8E .Y        DB12 .W        DB70 .COMP
DB96 .RRARG    DBD8 $6        DC1A .RARG     DC4C $8
DC76 .5        DC7A .ZERO     DC7E ATAN2     DD00 ATAN

COMMON-BLOCKS:
  DFFE RG        F902 SIM       F94F COLOR     F97E INTR
  FAF2 BUFFER

UNDEFINED:
  ONR          TNUM          BLKSIZ       TNUM         BLKSIZ       TRKFIL
  HNUM         INTSRV
```

JOIN

Purpose

JOIN connects chains which have coincident endpoints. Its primary use is to connect a sequence of short chains so that minimum chain overhead is produced by CMAP when creating the map image on drum.

Procedure Description

A list of chains to be connected is read into an array in the order in which they are to be connected. The data is then processed in 6400-byte blocks. A block is searched for chain headers and when one is found it is compared to the chain number array. If a match is found the entire chain is copied onto a temporary file, usually a drum file. If the chain is not to be connected it is written to the output buffer. When all chains have been examined, the connection process begins. The temporary file is repeatedly rewound and searched for each of the chains in the array. As a chain is located its points are copied to the output buffer. If a chain is to be connected in the opposite order to its current one it is inverted first and then connected.

Common Blocks

None

Subroutines

None

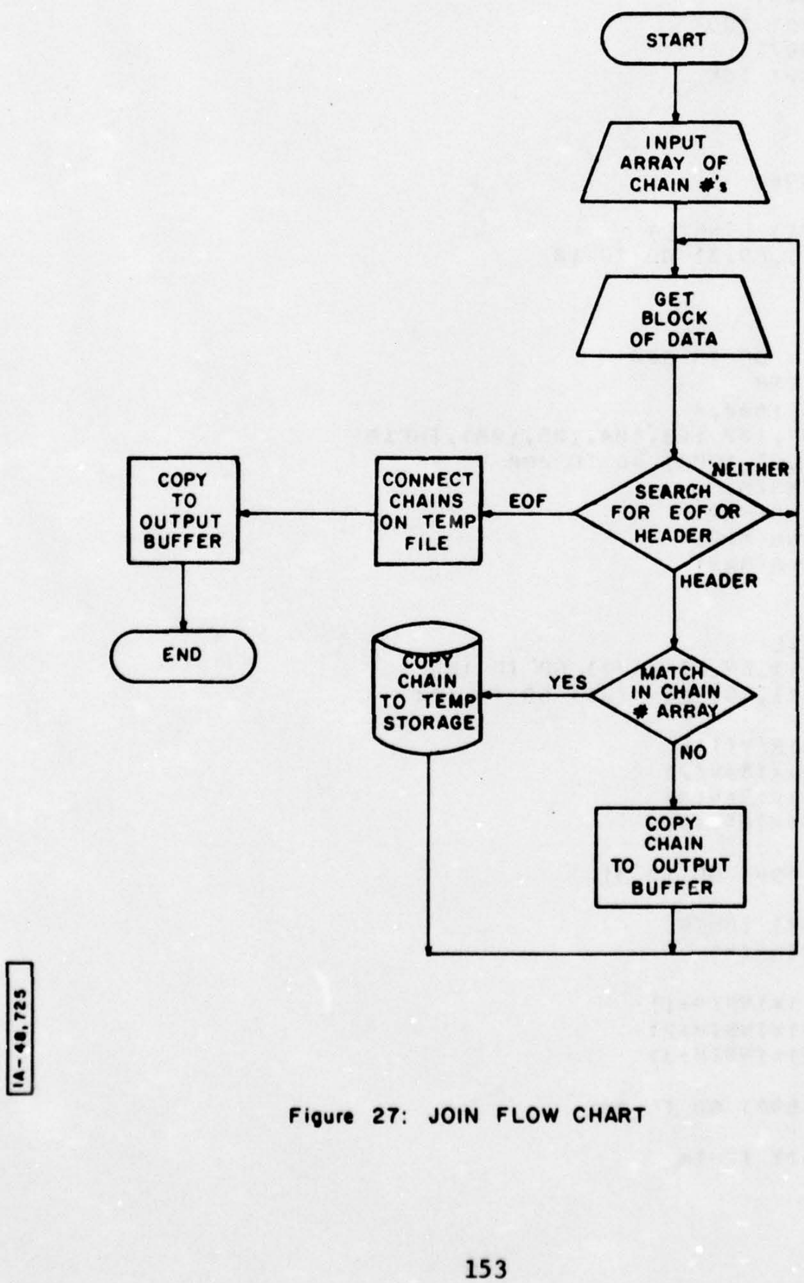



Figure 27: JOIN FLOW CHART

```
      DIMENSION INB(1600),IOUTB(1600),IS(1600),INT(2000)
      DIMENSION ISAV(4)
      DIMENSION XIN(2),XOUT(2),XS(2),LINK(20)
      EQUIVALENCE(INB(1),XIN(1)),(IOUTB(1),XOUT(1)),(IS(1),XS(1))
      EQUIVALENCE (INB(1),INT(1))
    1 WRITE(5,999)
      READ(5,900) IN
      WRITE(5,998)
      READ(5,900) IOUT
      WRITE(5,997)
      READ(5,900) ISK
      I=1
      IHEAD=1
      II=1
      WRITE(5,996)
      L=1
    3 READ(5,901) LINK(L)
      IF(LINK(L).EQ.0) GO TO 10
      L=L+1
      GO TO 3
   10 L=L-1
      IF(L.LE.0) GO TO 500
    2 READ(IN) INB
      DO 20 M=1,1600,4
      GO TO (101,102,103,104,105,106),IHEAD
  101 IF(INB(M).GT.1000) GO TO 400
      ISAV(1)=INB(M)
      ISAV(2)=INB(M+1)
      ISAV(3)=INB(M+2)
      ISAV(4)=INB(M+3)
      IHEAD=2
      GO TO 20
  102 DO 40 J=1,L
      IF(INB(M+1).EQ.LINK(J)) GO TO 104
      IF(INB(M+1).EQ.-LINK(J)) GO TO 104
   40 CONTINUE
      IOUTB(I)=ISAV(1)
      IOUTB(I+1)=ISAV(2)
      IOUTB(I+2)=ISAV(3)
      IOUTB(I+3)=ISAV(4)
      I=I+4
      IF(I.LT.1598) GO TO 31
      I=1
      WRITE(IOUT) IOUTB
   31 IOUTB(I)=INB(M)
      IHEAD=3
      IOUTB(I+1)=INB(M+1)
   21 IOUTB(I+2)=INB(M+2)
      IOUTB(I+3)=INB(M+3)
      I=I+4
      IF(I.LT.1598) GO TO 20
      I=1
      WRITE(IOUT) IOUTB
      GO TO 20
```

```
103 XOUT(I)=XIN(M)
    XOUT(I+1)=XIN(M+1)
    IF(XIN(M).EQ.1000.) IHEAD=1
    GO TO 21
104 IS(II)=ISAV(1)
    IS(II+1)=ISAV(2)
    IS(II+2)=ISAV(3)
    IS(II+3)=ISAV(4)
    II=II+4
    IF(II.LT.1598) GO TO 107
    II=1
    WRITE(ISK) IS
105 CONTINUE
107 IS(II)=INB(M)
    IHEAD=6
    IS(II+1)=INB(M+1)
108 IS(II+2)=INB(M+2)
    IS(II+3)=INB(M+3)
    II=II+4
    IF(II.LT.1598) GO TO 20
    II=1
    WRITE(ISK) IS
    GO TO 20
106 XS(II)=XIN(M)
    XS(II+1)=XIN(M+1)
    IF(XIN(M).EQ.1000.) IHEAD=1
    GO TO 108
 20 CONTINUE
    GO TO 2
400 XS(II)=1000.
    WRITE(ISK) IS
    IOUTB(I)=0
    IOUTB(I+1)=0
    IOUTB(I+2)=4
    IOUTB(I+3)=0
    I=I+4
    IF(I.LT.1598) GO TO 414
    I=1
    WRITE(IOUT) IOUTB
414 IOUTB(I)=1
    J=LINK(1)
    IF(J.LT.0) J=-J
    IOUTB(I+1)=J
    IOUTB(I+2)=0
    IOUTB(I+3)=0
    I=I+4
    IF(I.LT.1598) GO TO 404
    I=1
    WRITE(IOUT) IOUTB
404 DO 401 J=1,L
    CALL REW(ISK)
    I7=0
    NL=LINK(J)
    IF(NL.GT.0) GO TO 403
```

```
      IZ=1
      NL=-NL
  403 IHEAD=1
      II=1
  420 READ(ISK) IS
      DO 402 M=1,1600,4
      GO TO (601,602,603,604,605),IHEAD
  601 IHEAD=2
      IF(IS(M).GT.1000) PAUSE 100
      GO TO 402
  602 IHEAD=3
      IF(IS(M+1).NE.NL) GO TO 402
      IHEAD=4
      IF(IZ.EQ.1) IHEAD=5
      GO TO 402
  603 IF(XS(M).EQ.1000.) IHEAD=1
      GO TO 402
  604 IF(XS(M).EQ.1000.) GO TO 401
      XOUT(I)=XS(M)
      XOUT(I+1)=XS(M+1)
      IOUTB(I+2)=IS(M+2)
      IOUTB(I+3)=IS(M+3)
      I=I+4
      IF(I.LT.1598) GO TO 402
      I=1
      WRITE(IOUT) IOUTB
      GO TO 402
  605 IF(XS(M).EQ.1000.) GO TO 606
      XIN(II)=XS(M)
      XIN(II+1)=XS(M+1)
      INT(II+2)=IS(M+2)
      INT(II+3)=IS(M+3)
      II=II+4
  402 CONTINUE
      GO TO 420
  606 II=II-4
      IF(II.LE.0) GO TO 401
      XOUT(I)=XIN(II)
      XOUT(I+1)=XIN(II+1)
      IOUTB(I+2)=INT(II+2)
      IOUTB(I+3)=INT(II+3)
      I=I+4
      IF(I.LT.1598) GO TO 606
      I=1
      WRITE(IOUT) IOUTB
      GO TO 606
  401 CONTINUE
      XOUT(I)=1000.
      I=I+4
      IF(I.LT.1598) GO TO 501
      I=1
      WRITE(IOUT) IOUTB
  501 XOUT(I)=1000.
      WRITE(IOUT) IOUTB
```

```
      500 STOP
      900 FORMAT(I2)
      901 FORMAT(I4)
      996 FORMAT(31HENTER CHAINS TO BE LINKED(NNNN),/,
         1 34HNEGATIVE NUMBER FOR OPPOSITE ORDER,/,
         1 40HENTER ZERO AS END OF CHAINS TO BE LINKED,/,
         1 18HNULL CHAIN TO EXIT)
      997 FORMAT(31HENTER SCRATCH DEVICE NUMBER(NN))
      998 FORMAT(30HENTER OUTPUT DEVICE NUMBER(NN))
      999 FORMAT(29HENTER INPUT DEVICE NUMBER(NN))
          END
L .U      0000
E INB     0E5E
E INT     0E5E
E XIN     0E5E
E IOUTB   2D9E
E XOUT    2D9E
E IS      469E
E XS      469E
E ISAV    5F9E
E LINK    5FAE
A 1       0004
A 999     0F34
L .I      0000
A 900     0D3E
E IN      5FFE
A 998     0E0C
E IOUT    6002
A 997     0DE4
E ISK     6006
E I       600A
E IHEAD   6012
E II      6016
A 996     0D4E
E L       601A
A 3       00C8
A 901     0D46
A 10      0126
A 500     0D38
A 2       0144
L .J      0000
A 20      06C4
E M       6022
A 101     0182
A 102     021E
A 103     043F
A 104     04B0
A 105     0562
A 106     0652
A 400     06DA
A 40      028A
E J       602E
A 31      034F
A 21      03A2
```

```
A 107     0562
A 108     0586
A 414     07AA
A 404     0872
A 401     0CA4
L REW     0000
E IZ      604E
E NL      6052
A 403     08C4
A 420     08D4
A 402     0888
A 601     0910
A 602     0940
A 603     098E
A 604     09B4
A 605     0ABA
L .H      0000
A 606     0B9E
A 501     0D0A
L .S      0000
L .V      0000


            PROGRAMS:
   E168 .V        F1A2 REW       F13A .U        F1C2 .H
   F238 .MES      E176 @I        E05A @J        F204 .S
   F286


 ENTRY-POINTS:
   E05A @J        F146 .U        E16C .V        E176 @I
   F1A2 REW       F1C2 .H        F204 .S        F238 .MES


 COMMON-BLOCKS:
 NONE


 UNDEFINED:
 NONE
```

EDITDB

Purpose

EDITB is a point editor. The routine allows addition and deletion of points and changes in the coordinates and/or rank of points.

Procedure Description

A list of chain numbers in which the points to be edited are located is read into an array. Data is processed in 6400-byte blocks. Each block is searched for chain headers and when one is found it is compared to the chain number array. If a match is found, the entire chain is copied onto a temporary file. Those chains for which a match is not found are copied to the output buffer.

Upon completion of the chain search the actual editing begins. Editing is performed on a chain-by-chain basis. For a given chain the user specifies the position and coordinates and/or rank, when applicable, of the point. The new data is sent to the output buffer and the next position to be edited in that chain is specified if there is one. If not, the remainder of the chain is copied to the output buffer. The process is repeated for each chain in the temporary file.

Common Blocks

| Block Name | Contents | Description of Contents |
|---|---|---|
| /BLK1/ | IOUT | output device number |
| /BLK4/ | USTDP, RUSTDP, CTRLME RGCOBE, PI, PI1, COPST | constants for conic pro - jection |

Subroutines

MTBUFR: copies output buffer to tape and resets the pointer to the top of the buffer.

CALL MTBUFR (IOUTB, I)

IOUTB: output buffer

I: pointer in the buffer

PROJEC: Projection of point from the globe onto the plane via a secant cone.

CALL PROJEC (RLAT, RLONG, X, Y)

RLAT: latitude of point in radians

RLONG: longitude of point in radians

X: projected x-coordinate, returned by routine

Y: projected Y-coordinate, returned by routine

See SUBSET documentation in this section.

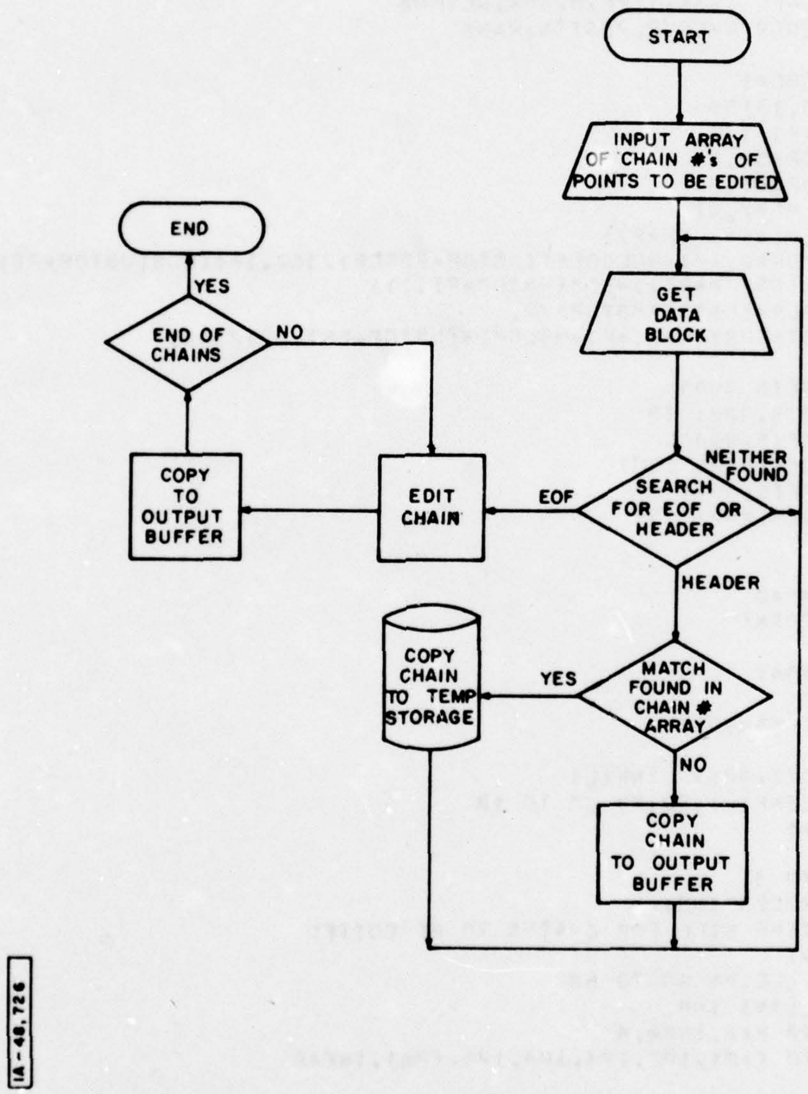



Figure 28: EDITDB FLOW CHART

```
      DIMENSION INB(1600),IOUTB(1600),IS(1600),INT(2000)
      DIMENSION ISAV(4)
      DIMENSION XIN(2),XOUT(2),XS(2),LINK(20)
      COMMON /BLK1/ IOUT
      COMMON/BLK4/USTDP,RUSTDP,CTRLME,RGLOBE,PI,PI1,CONST
      EQUIVALENCE(INB(1),XIN(1)),(IOUTB(1),XOUT(1)),(IS(1),XS(1))
      EQUIVALENCE (INB(1),INT(1))
      INTEGER TASK,TYPE,BLANK,GETPOS
      INTEGER WHERUR,POSITN,RANK
C
      RGLOBE=1
      PI=3.14159
      PI1=PI/180.
      USTDP=57.00
      BSTDP=41.00
      CTRLME=7.47
      CTRLME=CTRLME*PI1
      RUSTDP=2.*PI*RGLOBE*((USTDP-BSTDP)/360.)*((COS(USTDP*PI1))/
     C(COS(BSTDP*PI1)-COS(USTDP*PI1)))
      CNTRLP=(USTDP+BSTDP)/2.
      CONST=RUSTDP+2.*PI*RGLOBE*(USTDP-CNTRLP)/360.
C
    1 WRITE(5,999)
      READ(5,900) IN
      WRITE(5,998)
      READ(5,900) IOUT
      WRITE(5,997)
      READ(5,900) ISK
C
C
      BLANK=0
      GETPOS=1
      I=1
      IHEAD=1
      II=1
      WRITE(5,996)
      L=1
    3 READ(5,901) LINK(L)
      IF(LINK(L).EQ.0) GO TO 10
      L=L+1
C
      GO TO 3
C START PROCESSING:
C CREATE TEMP FILE FOR CHAINS TO BE EDITED
   10 L=L-1
      IF(L.LE.0) GO TO 500
    2 READ(IN) INB
      DO 20 M=1,1600,4
      GO TO (101,102,103,104,105,106),IHEAD
C
  101 IF(INB(M).GT.1000) GO TO 400
      ISAV(1)=INB(M)
      ISAV(2)=INB(M+1)
      ISAV(3)=INB(M+2)
```

```
      ISAV(4)=INB(M+3)
      IHEAD=2
      GO TO 20
C
  102 DO 40 J=1,L
      IF(INB(M+1).EQ.LINK(J)) GO TO 104
   40 CONTINUE
      IOUTB(I)=ISAV(1)
      IOUTB(I+1)=ISAV(2)
      IOUTB(I+2)=ISAV(3)
      IOUTB(I+3)=ISAV(4)
      I=I+4
      IF(I.GE.1598) CALL MTBUFR(IOUTB,I)
      IOUTB(I)=INB(M)
      IHEAD=3
      IOUTB(I+1)=INB(M+1)
   21 IOUTB(I+2)=INB(M+2)
      IOUTB(I+3)=INB(M+3)
      I=I+4
      IF(I.GE.1598) CALL MTBUFR(IOUTB,I)
      GO TO 20
C
  103 XOUT(I)=XIN(M)
      XOUT(I+1)=XIN(M+1)
      IF(XIN(M).EQ.1000.) IHEAD=1
      GO TO 21
C
  104 IS(II)=ISAV(1)
      IS(II+1)=ISAV(2)
      IS(II+2)=ISAV(3)
      IS(II+3)=ISAV(4)
      II=II+4
      IF(II.LT.1598) GO TO 107
      II=1
      WRITE(ISK) IS
C
  105 CONTINUE
  107 IS(II)=INB(M)
      IHEAD=6
      IS(II+1)=INB(M+1)
  108 IS(II+2)=INB(M+2)
      IS(II+3)=INB(M+3)
      II=II+4
      IF(II.LT.1598) GO TO 20
      II=1
      WRITE(ISK) IS
      GO TO 20
C
  106 XS(II)=XIN(M)
      XS(II+1)=XIN(M+1)
      IF(XIN(M).EQ.1000.) IHEAD=1
      GO TO 108
   20 CONTINUE
      GO TO 2
```

```
  400 XS(II)=1000.
      WRITE(ISK) IS
C
C START EDITING ON CHAIN BY CHAIN BASIS
C
      WRITE (5,995)
      WRITE(5,970)
      DO 401 J=1,L
      CALL REW(ISK)
      GETPOS=1
      WHERUR=0
      NL=LINK(J)
      WRITE (5,980)NL
  403 IHEAD=1
C
  420 READ(ISK) IS
      DO 402 M=1,1600,4
      GO TO (601,602,603,604),IHEAD
C
  601 IHEAD=2
      IF(IS(M).GT.1000) PAUSE 100
      LSN1=IS(M)
      LSN2=IS(M+1)
      TYPE=IS(M+2)
      GO TO 402
C
  602 IHEAD=3
      IF(IS(M+1).NE.NL) GO TO 402
      IHEAD=4
      IOUTB(I)=LSN1
      IOUTB(I+1)=LSN2
      IOUTB(I+2)=TYPE
      IOUTB(I+3)=BLANK
      I=I+4
      IF(I.GE.1598) CALL MTBUFR(IOUTB,I)
      IOUTB(I)=IS(M)
      IOUTB(I+1)=IS(M+1)
      IOUTB(I+2)=IS(M+2)
      IOUTB(I+3)=IS(M+3)
      I=I+4
      IF(I.GE.1598) CALL MTBUFR(IOUTB,I)
      GO TO 402
C
  603 IF(XS(M).EQ.1000.) IHEAD=1
      GO TO 402
C
604   WHERUR=WHERUR+1
      IF(GETPOS.EQ.0) GO TO 605
      WRITE(5,992)
      READ(5,991) POSITN
405   WRITE(5,994) POSITN
      READ(5,993) TASK
      IF((0.GT.TASK).OR.(5.LT.TASK)) GO TO 405
      IFOUND=1
```

```
      GETPOS=0
605   IF((WHERUR.EQ.POSITN).AND.(IFOUND.EQ.1)) GO TO 607
      XOUT(I)=XS(M)
      XOUT(I+1)=XS(M+1)
      IOUTB(I+2)=IS(M+2)
      IOUTB(I+3)=IS(M+3)
      I=I+4
      IF(I.GE.1598) CALL MTBUFR(IOUTB,I)
      IF(XS(M).EQ.1000.0) GO TO 406
      GO TO 402
607   IF((TASK.EQ.2).OR.(TASK.EQ.5)) GO TO 609
      WRITE(5,990)
      READ(5,989) RLAT,RLONG
      CALL PROJEC(RLAT,RLONG,X,Y)
      XOUT(I)=X
      XOUT(I+1)=Y
      IF((TASK.EQ.3).OR.(TASK.EQ.4)) GO TO 609
      IOUTB(I+2)=IS(M+2)
      IOUTB(I+3)=IS(M+3)
      I=I+4
      IF(I.GE.1598) CALL MTBUFR(IOUTB,I)
      GO TO 613
C
609   IF(TASK.EQ.5) GO TO 613
      WRITE(5,988)
      READ(5,900) RANK
      IOUTB(I+3)=RANK
      I=I+4
      IF(I.GE.1598) CALL MTBUFR(IOUTB,I)
      IF(TASK.NE.4) GO TO 613
C
      M=M-4
C
613   IF(TASK.EQ.5) WHERUR=WHERUR-1
      IFOUND=0
      WRITE(5,986)
      READ(5,993) IBACK
      IF(IBACK.EQ.1) GETPOS=1
402   CONTINUE
      GO TO 420
C
406   IF(IFOUND.EQ.0) GO TO 401
      WRITE(5,985)
  401 CONTINUE
      XOUT(I)=1000.
      WRITE(IOUT) IOUTB
  500 STOP
  999 FORMAT(29HENTER INPUT DEVICE NUMBER(NN))
  900 FORMAT(I2)
  998 FORMAT(30HENTER OUTPUT DEVICE NUMBER(NN))
997   FORMAT (31HENTER SCRATCH DEVICE NUMBER(NN))
  996 FORMAT(31HENTER CHAINS TO BE EDITED(NNNN),/,
     1 40HENTER ZERO AS END OF CHAINS TO BE EDITED)
  901 FORMAT(I4)
```

```
992    FORMAT(42H ENTER POSITION OF POINT TO BE EDITED(NNN))
991    FORMAT(I3)
980    FORMAT(16H CHAIN NUMBER = ,I3)
994    FORMAT(19H POSITION NUMBER = ,I3,22H ENTER TASK NUMBER (N))
993    FORMAT(I1)
995    FORMAT(13H EDIT CODES :,/,21H 1-CHANGE COORDINATES,/,
      122H 2-CHANGE DETAIL LEVEL,/,7H 3-BOTH,/,12H 4-ADD POINT)
986    FORMAT(31H FURTHER EDITING IN SAME CHAIN?,/,
      118H ENTER 1=YES, 0=NO)
985    FORMAT(19H POSITION NOT FOUND)
988    FORMAT(24H ENTER DETAIL LEVEL (NN))
989    FORMAT(E14.8,2X,F14.8)
990    FORMAT(47H ENTER LAT AND LONG IN RADIANS (E14.8,2X,E14.8))
970    FORMAT(15H 5-DELETE POINT)
       END
I BLK1    0004
E IOUT    0000
I BLK4    001C
E USTOP   0000
E RUSTOP  0004
E CTRLME  0008
E RGLOBE  000C
E PI      0010
E PI1     0014
E CONST   0018
L .U      0000
E INB     11CC
E INT     11CC
E XIN     11CC
E IOUTB   310C
E XOUT    310C
E IS      4A0C
E XS      4A0C
E ISAV    630C
E LINK    631C
E TASK    636C
E TYPE    6370
E BLANK   6374
E GETPOS  6378
E WHERUR  637C
E POSITN  6380
E RANK    6384
L .W      0000
E BSTOP   6398
L COS     0000
E CNTRLP  63E4
A 1       0102
A 999     0E04
L .I      0200
A 900     0F2A
E IN      63E8
A 998     0F32
A 997     0F5A
E ISK     63FC
```

```
E I       63F4
E IHEAD   63F8
F II      63FC
A 996     0F82
E L       6400
A 3       01D6
A 901     0FD8
A 10      0234
A 500     0EFE
A 2       0252
L .J      0000
A 20      0766
E M       6404
A 101     0290
A 102     032C
A 103     04E0
A 104     0552
A 105     0604
A 106     06F4
A 400     077C
A 40      0364
E J       6410
L MTBUFR  0000
A 21      0460
A 107     0604
A 108     0658
A 995     107A
A 970     1180
A 401     0FBE
L REW     0000
E NL      6428
A 980     101C
A 403     0822
A 420     082A
A 402     0F82
A 601     0864
A 602     08DC
A 603     0A62
A 604     0A88
L .H      0000
E LSN1    642C
E LSN2    6430
A 605     0B3E
A 992     0FE0
A 991     1014
A 405     0AD6
A 994     1038
A 993     1072
E IFOUND  6436
A 607     0C48
A 406     0E98
A 609     0D7C
A 990     1170
A 989     1164
```

```
E RLAT    643C
E RLONG   6440
L PROJEC  0000
E X       6444
E Y       6448
A 613     0E1A
A 988     1142
A 986     10E6
E IBACK   644C
A 985     1126
L .S      0000
L .V      0000
```

```
           PROGRAMS:
  B630 @J        B710 ,P        B798 .Q        B848 .O
  B890 ,MES      B90E ,U        B93C ,V        B94A @I
  C976 ,H        C9B8 .S        C9EC REW       CA0C COS
  CA30 SIN       CAEA AMOD      CB18 AINT      CB8A ,W
  CBE8 .COMP     CC0E ,RRARG    CC50 S6        CC92 .RARG
  CCC4 S8        CCEE .5        CCF2 .ZERO     CCF6

ENTRY-POINTS:
  B474 MTBUFR    B4F0 PROJEC    B630 @J        B710 ,P
  B798 ,Q        B848 .O        B890 .MES      B91A ,U
  B940 ,V        B94A @I        C976 .H        C9B8 .S
  C9EC REW       CA0C COS       CA30 SIN       CAEA AMOD
  CB18 AINT      CB8A ,W        CBE8 .COMP     CC0E ,RWARG
  CC50 S6        CC92 .RARG     CCC4 S8        CCEE .5
  CCF2 .ZERO

COMMON-BLOCKS:
  FFBE BLK1      FFC2 BLK4

UNDEFINED:
NONE
```

MERGE

Purpose

MERGE combines data files (e.g., coastline and boundary data) sequentially.

Procedure Description

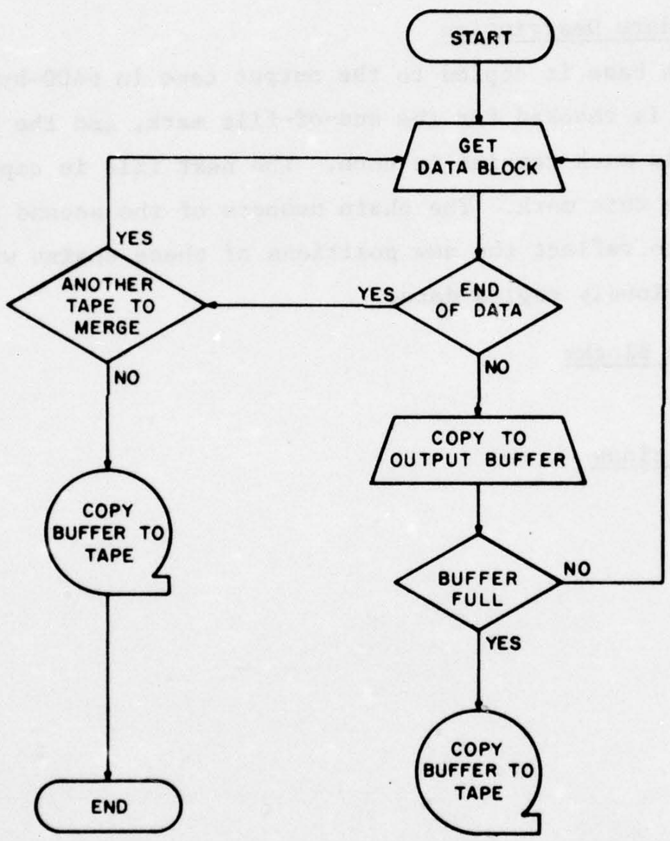
A data base is copied to the output tape in 6400-byte blocks. Each block is checked for the end-of-file mark, and the block containing this mark remains in core. The next file is copied beginning just before this mark. The chain numbers of the second data base are increased to reflect the new positions of these chains with respect to the previously copied data.

Common Blocks

None

Subroutines

None




IA-48,727

**Figure 29 MERGE FLOWCHART**

```
      DIMENSION BUFF(1600),BUFF2(1600),IBUFF(1600),IBUFF2(1600)
      EQUIVALENCE (BUFF(1),IBUFF(1)),(BUFF2(1),IBUFF2(1))
      WRITE(5,989)
      READ(5,988) IN
      WRITE(5,987)
      READ(5,988) IOUT
      IHEAD=1
      ISEQ=1
    1 READ(IN) BUFF
      DO 10 I=1,1600,4
      IF(IHEAD.EQ.0) GO TO 11
      IF(IHEAD.EQ.2) GO TO 12
      IF(BUFF(I).EQ.1000.) GO TO 20
      IHEAD=2
      GO TO 10
   12 IBUFF(I+1)=ISEQ
      ISEQ=ISEQ+1
      IHEAD=0
      GO TO 10
   11 IF(BUFF(I).NE.1000.) GO TO 10
      IHEAD=1
   10 CONTINUE
      WRITE(IOUT) BUFF
      GO TO 1
   20 WRITE(5,986)
      READ(5,988) IN
      IF(IN.EQ.0) GO TO 100
   21 READ(IN) BUFF2
      IHEAD=1
   22 DO 30 M=1,1600,4
      IF(IHEAD.EQ.0) GO TO 31
      IF(IHEAD.EQ.2) GO TO 32
      IF(BUFF2(M).EQ.1000.) GO TO 20
      IHEAD=2
   33 IBUFF(I)=IBUFF2(M)
      IBUFF(I+1)=IBUFF2(M+1)
      IBUFF(I+2)=IBUFF2(M+2)
      IBUFF(I+3)=IBUFF2(M+3)
      I=I+4
      GO TO 29
   32 IBUFF2(M+1)=ISEQ
      ISEQ=ISEQ+1
      IHEAD=0
      GO TO 33
   31 BUFF(I)=BUFF2(M)
      BUFF(I+1)=BUFF2(M+1)
      BUFF(I+2)=BUFF2(M+2)
      IBUFF(I+3)=IBUFF2(M+3)
      I=I+4
      IF(BUFF2(M).NE.1000.) GO TO 29
      IHEAD=1
   29 IF(I.LT.1600) GO TO 30
      I=1
      WRITE(IOUT) BUFF
```

```
   30 CONTINUE
      READ(IN) BUFF2
      GO TO 22
  100 BUFF(I)=1000.
      WRITE(IOUT) BUFF
      ISEQ=ISEQ-1
      WRITE(5,985) ISEQ
      STOP
  987 FORMAT(30HENTER OUTPUT DEVICE NUMBER(NN))
  988 FORMAT(I2)
  989 FORMAT(29HENTER INPUT DEVICE NUMBER(NN))
  986 FORMAT(52HENTER INPUT DEVICE NUMBER(NN) - IF FINISHED ENTER 00)
  985 FORMAT(22HTOTAL NUMBER OF CHAINS,I5)
      END
I  .U       0000
E  BUFF     0558
E  IBUFF    0558
E  BUFF2    1E58
E  IBUFF2   1E58
A  989      04CE
I  .I       0000
A  988      04C6
E  IN       3758
A  987      049E
E  IOUT     375C
E  IHEAD    3760
E  ISEQ     3768
A  1        0074
I  .J       0000
A  10       013C
E  I        376C
A  11       0116
A  12       00F4
A  20       016C
A  986      04F4
A  100      0442
A  21       01AE
A  22       01D3
A  30       0412
E  M        377C
A  31       0308
A  32       0206
A  33       0222
A  29       03DE
A  985      0532
I  .S       0000
I  .V       0000
```

```
              PROGRAMS:
   8788 @J          8868 .U          8896 .           88A4 @I
   98D8 .S          9984 .MFS        9982

ENTRY-POINTS:
   8788 @J          8874 .U          889A .V          88A4 @I
   98D8 .S          9984 .MFS

COMMON-BLOCKS:
NONE

UNDEFINED:
NONE
```

## APPENDIX A

## DATA BASE FORMATS

Table I

World Data Bank I

Block Size: 1600 bytes

Record Size: 80 characters

File 1 and File 2: Coastline and Boundary data files

| Field No. | Field Length (Bytes) | Data Description | Format |
|---|---|---|---|
| 1 | 7 | Line Segment Number | I7 |
| 2 | 20 | Latitude in Radians | E20.8 |
| 3 | 20 | Longitude in Radians | E20.8 |
| 4 | 3 | Blank | 3X |
| 5 | 2 | Latitude - Degree part | I2 |
| 6 | 2 | Latitude - Minute part | I2 |
| 7 | 2 | Latitude - Second part | I2 |
| 8 | 1 | Direction-N(North),S(South) | A1 |
| 9 | 3 | Longitude - Degree part | I3 |
| 10 | 2 | Longitude - Minute part | I2 |
| 11 | 2 | Longitude - Second part | I2 |
| 12 | 1 | Direction-E(East),W(West) | A1 |
| 13 | 3 | Blank | 3X |
| 14 | 2 | Rank | A2 |
| 15 | 1 | Blank | 1X |
| 16 | 9 | Record Sequence Number | I9 |

File 3: The third file of WDBI is actually a merge of two files--the Map Area Code Index and World Data Bank I index. The format of each is given below.

Map Area Code Index

| Field No. | Field Length (Bytes) | Data Description | Format |
|---|---|---|---|
| 1 | 11 | Map Area Code | A11 |
| 2 | 1 | Blank | 1X |
| 3 | 24 | Map Area Description | A24 |
| 4 | 44 | Blank | 44X |

World Data Bank I Index

| Field No. | Field Length (Bytes) | Data Description | Format |
|---|---|---|---|
| 1 | 11 | Map Area Code | A11 |
| 2 | 9 | Blank | 9X |
| 3 | 1 | Map Feature | A1 |
| | | B = Boundary | |
| | | I = Island | |
| | | C = Coastline | |
| | | L = Lake | |
| 4 | 4 | Blank | 4X |
| 5 | 1 | Rank | A1 |
| | | 1 = appears on all maps | |
| | | 2 = single point islands | |
| 6 | 4 | Blank | 4X |
| 7 | 7 | Line Segment Number | I7 |
| 8 | 43 | Blank | 43X |

Table II

Reduced World Data Bank I

Block Size: 1600 bytes

Record Size: 16 byte binary

File 1 and File 2: Coastline and Boundary data files

| Field No. | Field Length(Bytes) | Data Description |
|---|---|---|
| 1 | 4 | First three digits of Line Segment Number |
| 2 | 4 | Last four digits of Line Segment Number |
| 3 | 4 | Latitude in radians |
| 4 | 4 | Longitude in radians |

Table III

Projected Data Base

Block Size: 1600 bytes

For each chain:

| Field No. | Field Length (Bytes) | Data Description |
|---|---|---|
| 1 | 32 | Chain ID (q.v.) |
| 2 | Variable | Data points (q.v.) |
| 3 | 16 | End of chain (q.v.) |

Chain ID

| Field No. | Field Length (Bytes) | Data Description |
|---|---|---|
| 1 | 4 | First 3 digits of original line segment number |
| 2 | 4 | Last 4 digits of original line segment number |
| 3 | 4 | Type of chain<br>1 = coastline<br>2 = boundary<br>3 = box<br>4 = other |
| 4 | 4 | Unused integer field |
| 5 | 4 | Original source chain number |
| 6 | 4 | Cumulative chain number |
| 7 | 8 | Unused integer field |

Data Points

| Field No. | Field Length (Bytes) | Data Description |
|---|---|---|
| 1 | 4 | X - coordinate |
| 2 | 4 | Y - coordinate |
| 3 | 4 | Deviation of point from its trend line* |
| 4 | 4 | Rank of point* |

*this value is determined by DETAIL

End of chain

| Field No. | Field Length (Bytes) | Data Description |
|---|---|---|
| 1 | 4 | 1000.0 |
| 2 | 12 | Unused integer field |

Table IV

WDBI Index by Line Segment Number

| Field No. | Field Length (Bytes) | Data Description | Format |
|---|---|---|---|
| 1 | 3 | First part of Line Segment Number | I3 |
| 2 | 4 | Second part of Line Segment Number | I4 |
| 3 | 2 | Blank | 2X |
| 4 | 1 | Map Feature | A1 |
| 5 | 2 | Blank | 2X |
| 6 | 24 | Map Area Description | 6A4 |
| 7 | 2 | Blank | 2X |
| 8 | 40 | Map Area Code | 10A4 |

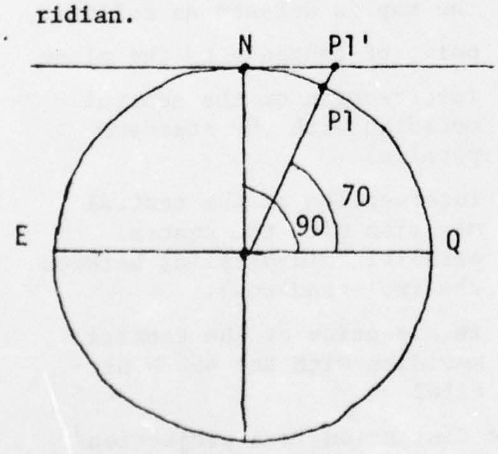
## APPENDIX B

### COMPARISON OF MAP SCALE DISTORTION BY PROJECTION TYPE

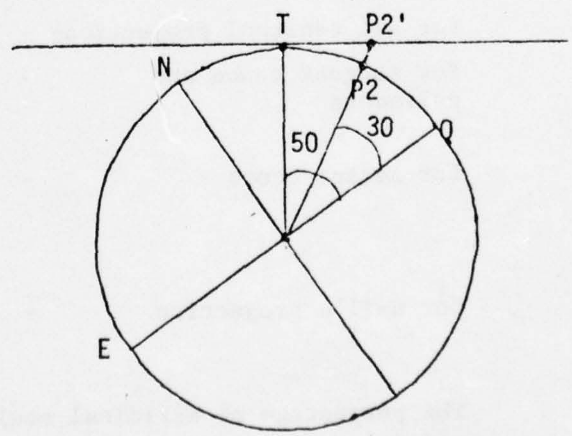
### REGION DESCRIPTION

Assume the region to be mapped has a 40° latitudinal span and a 60° longitudinal span (a somewhat larger east-west extent than the European region we map). The location of the region will vary among projections in order to simplify calculations and to minimize distortion.

For the zenithal projections, the region is assumed to be centered at the North Pole. The scale error of a point whose position is measured relative to the point of tangency to the plane is unefected by the absolute location of either point. Thus, for example, a point 20° south of the North Pole with the projection plane tangent at the pole will undergo the same scale increase/decrease along a radial line from the point of tangency as a point in latitude 30° N if the projection plane were tangent along the parallel 50° N (in the figures below NP1' = TP2'). In the former this radial line is a meridian.




N = North Pole
EQ = Equator
P1 is in latitude 70° N




T is the latitude 50° N
P2 is in latitude 30° N

In the cylindrical group, the region is centered along the equator for both the Mercator and Simple Perspective Projections so that the normal position of the cylinder is used and calculations are minimized. For Gall's projection, the area is centered near the 45° N parallel as the cylinder intersects the globe at 45° N and 45° S parallels and thus the minimum distortion of the region is calculated.

In both cases of the conical group for which scale distortion is given, the region is centered at the 49° N parallel. Thus, the tangent cone is tangent along the 49° N parallel and the secant cone intersects the globe at 41° N and 57° N parallels.

COMPARISON OF SCALE DISTORTION

For each projection two types of scale distortion have been determined; meridianal and parallel. The percentage of parallel scale distortion for projections in the three classes has been determined by calculating the ratio of the length of the parallel N° away from the 'center' of the map to its corresponding length on a globe of unit radius. The 'center' of the map is defined as follows:

| | | |
|---|---|---|
| for all zenithal projections | - | point of tangency to the plane |
| for tangent cones and cylinders | - | intersection of the central meridian with the standard parallel |
| for secant cones | - | intersection of the central meridian with the central parallel (mid-parallel between the two standards). |
| for Gall's projection | - | intersection of the central meridian with the 45° N parallel |

The percentage of meridinal scale distortion is a projection/globe distance ratio and is determined as follows:

| | | |
|---|---|---|
| zenithal projections | - | meridianal distance from the center of projection to a point |

| | | |
|---|---|---|
| | | in parallel N° away from the center |
| cylindrical projections | - | meridianal distance from the equator to a point in parallel N° away from the center |
| conical projections | - | meridianal distance from the equator to a point in parallel N° away from the center |

Tables V a and b list the scale distortion for the stereographic, orthographic and gnomonic cases of the zenithal group; Tables VI a and b lists the same for three cases of the cylindrical group and Tables VII a and b for two conical projections.

Table V

ZENITHAL PROJECTIONS

(a) % MERIDIANAL SCALE DISTORTION

| Projection | Distance from Point of Tangency (Degrees) | |
|---|---|---|
| | 20° | 30° |
| Stereographic | 1 % | 2.3 % |
| Orthographic | 2 % | 4.5 % |
| Gnomonic | 4.2 % | 10.3 % |

(b) % PARALLEL SCALE DISTORTION

| Projection | Distance from Point of Tangency (Degrees) | |
|---|---|---|
| | 10° | 20° |
| Stereographic | 0.8 % | 3.1 % |
| Orthographic | 0 % | 0 % |
| Gnomonic | 1.5 % | 6.0 % |

Table VI

CYLINDRICAL PROJECTIONS

(a) % MERIDIANAL SCALE DISTORTION

| Projection | Distance from Standard Parallel (Degrees) | |
|---|---|---|
| | 20° | 30° |
| Mercator | 2.1 % | 4.9 % |
| Simple Perspective (one Standard Parallel) | 4.2 % | 10.2 % |
| Gall's (two Standard parallels) | 13.3 % | 14.2 % |

(b) % PARALLEL SCALE DISTORTION

| Projection | Distance from Standard Parallel (Degrees) | |
|---|---|---|
| | 10° | 20° |
| Mercator | 1.5 % | 6.4 % |
| Simple Perspective (one Standard Parallel) | 1.5 % | 6.4 % |
| Gall's (two Standard parallels) | 15.8 % | 21.9 % |

Table VII

CONICAL PROJECTIONS

(a) % MERIDIANAL SCALE DISTORTION

| Projection | Distance from Standard/Central Parallel (Degrees) | |
|---|---|---|
| | 20° | 30° |
| One Standard Parallel | 0 % | 0 % |
| Two Standard Parallels | 0 % | 0 % |

(b) % PARALLEL SCALE DISTORTION

| Projection | Distance from Standard/Central Parallel (Degrees)* | |
|---|---|---|
| | 9° | 19° |
| One Standard Parallel | 1.2 % | 5.7 % |
| Two Standard Parallels | 0.2 % | 3.8 % |

*9° and 19° are used instead of 10° and 20° because distortion at this distance had been previously determined for the conic projection with central parallel at 49° for the European map displays.

## BIBLIOGRAPHY

Central Intelligence Agency, CAM, "Cartographic Automatic Mapping Program Documentation - Version 4", OGCR CD 75-1 (Revision), March 1975.

Deetz, Charles H. and Oscar Adams, Elements of Map Projection with Applications to Map and Chart Constitution; Special Publication No. 68, Fifth Edition, Revised, U. S. Department of Commerce Coast and Geodetic Survey, Washington, D. C. 1945.

Kelloway, George P., Map Projections, Second Edition, London, Methuen and Co., Ltd., July 1949.

Laboratory for Computer Graphics and Spatial Analysis, Harvard University, "POLYVRT User's Manual," Version 1.1, Cambridge, Mass. November 1974.

Mainwaring, James, An Introduction to the Study of Map Projections, London, MacMillan and Co., Limited, 1942.

National Technical Information Service, "World Data Bank I Bibliographic Data Sheet," PB 223 178, Springfield, Virginia, 29 August 1973, (U).