General Disclaimer

One or more of the Following Statements may affect this Document

- This document has been reproduced from the best copy furnished by the organizational source. It is being released in the interest of making available as much information as possible.

- This document may contain data, which exceeds the sheet parameters. It was furnished in this condition by the organizational source and is the best copy available.

- This document may contain tone-on-tone or color graphs, charts and/or pictures, which have been reproduced in black and white.

- This document is paginated as submitted by the original source.

- Portions of this document are not fully legible due to the historical nature of some of the material. However, it is the best reproduction available from the original submission.

Produced by the NASA Center for Aerospace Information (CASI)
The LUMIS program has designed an integrated geographic information system to assist program managers and planning groups in metropolitan regions. Described is the series of computer software programs and procedures involved in data base construction using the census DIME file and point-in-polygon architectures.

The system is described in two parts: (1) instructions to operators with regard to digitizing and editing procedures, and (2) application of the four data base construction algorithms: ROTATE (to achieve map registration), CHAIN (to assure the topological integrity of polygon files), DACS (DIME AREA CENTROID SYSTEM alternative to CHAIN), and PIOS (the polygon intersection overlay system for tabulating land use acreages within administrative districts).
LUMIS

LAND USE MANAGEMENT AND INFORMATION SYSTEMS

COORDINATE ORIENTED PROGRAM DOCUMENTATION

Nevin Bryant
Project Manager
Charles K. Paul
NASA Headquarters
Albert J. Landini
City Planner
R. Wayne Bannister
City Planning Associate
Thomas Logan
Project Data Analyst

Jet Propulsion Laboratory
California Institute of Technology
Pasadena, California 91103

November 1, 1976

Prepared for
Office of Technology Utilization
and
Office of Applications
NATIONAL AERONAUTICS AND SPACE ADMINISTRATION
PREFACE

This work was sponsored by the Office of Technology Utilization of the National Aeronautics and Space Administration through Contract No. NAS 7-100.

This document is one of two principal systems specifications documents to be generated by the LUMIS program. While this document addresses the problem of software and procedures involved in data base construction using the census DIME file and point-in-polygon architectures, the other\(^1\) presents the problem of data base interrogation.

## CONTENTS

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0 INTRODUCTION</td>
<td>1-1</td>
</tr>
<tr>
<td>1.1 The System</td>
<td>1-1</td>
</tr>
<tr>
<td>1.2 The Process</td>
<td>1-1</td>
</tr>
<tr>
<td>2.0 DIGITIZING PROCEDURES</td>
<td>2-1</td>
</tr>
<tr>
<td>2.1 Introduction</td>
<td>2-1</td>
</tr>
<tr>
<td>2.2 Case 1: General Map Case</td>
<td>2-1</td>
</tr>
<tr>
<td>2.2.1 Map tolerance record</td>
<td>2-2</td>
</tr>
<tr>
<td>2.2.2 Control reference points</td>
<td>2-4</td>
</tr>
<tr>
<td>2.2.3 Polygon description record</td>
<td>2-4</td>
</tr>
<tr>
<td>2.2.4 Polygon limit record</td>
<td>2-4</td>
</tr>
<tr>
<td>2.2.5 Polygon digitized boundary records</td>
<td>2-6</td>
</tr>
<tr>
<td>2.3 Case 2: MMS Map Case</td>
<td>2-7</td>
</tr>
<tr>
<td>2.3.1 Map tolerance record</td>
<td>2-7</td>
</tr>
<tr>
<td>2.3.2 Control reference points</td>
<td>2-9</td>
</tr>
<tr>
<td>2.3.3 Digitized reference points</td>
<td>2-9</td>
</tr>
<tr>
<td>2.3.4 Street segment record</td>
<td>2-10</td>
</tr>
<tr>
<td>3.0 CALCULATING THE STATE PLANE COORDINATES</td>
<td>3-1</td>
</tr>
<tr>
<td>3.1 Introduction</td>
<td>3-1</td>
</tr>
<tr>
<td>3.2 Procedure</td>
<td>3-1</td>
</tr>
<tr>
<td>3.3 Calculations</td>
<td>3-2</td>
</tr>
<tr>
<td>4.0 EDITING PROCEDURES</td>
<td>4-1</td>
</tr>
<tr>
<td>4.1 Introduction</td>
<td>4-1</td>
</tr>
<tr>
<td>4.2 General Map</td>
<td>4-1</td>
</tr>
<tr>
<td>4.3 MMS Maps</td>
<td>4-3</td>
</tr>
<tr>
<td>4.4 CHAIN</td>
<td>4-4</td>
</tr>
</tbody>
</table>
## CONTENTS (contd)

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.0 ROTATE</td>
<td>5-1</td>
</tr>
<tr>
<td>5.1 Introduction</td>
<td>5-1</td>
</tr>
<tr>
<td>5.2 Methodology</td>
<td>5-1</td>
</tr>
<tr>
<td>5.2.1 The scaling factor S</td>
<td>5-2</td>
</tr>
<tr>
<td>5.2.2 The rotation angle ( \theta )</td>
<td>5-2</td>
</tr>
<tr>
<td>5.2.3 The surveyed control points</td>
<td>5-2</td>
</tr>
<tr>
<td>5.2.4 The transformation equation</td>
<td>5-3</td>
</tr>
<tr>
<td>5.3 Program</td>
<td>5-3</td>
</tr>
<tr>
<td>5.3.1 Phase I</td>
<td>5-3</td>
</tr>
<tr>
<td>5.3.2 Phase II</td>
<td>5-4</td>
</tr>
<tr>
<td>5.3.2.1 Input</td>
<td>5-4</td>
</tr>
<tr>
<td>5.3.2.2 Output</td>
<td>5-4</td>
</tr>
<tr>
<td>5.4 Overall Specifications</td>
<td>5-4</td>
</tr>
<tr>
<td>5.5 Phase I Specifications</td>
<td>5-4</td>
</tr>
<tr>
<td>5.5.1 Map tolerance record</td>
<td>5-6</td>
</tr>
<tr>
<td>5.5.2 Control reference points</td>
<td>5-6</td>
</tr>
<tr>
<td>5.5.3 Digitized reference points</td>
<td>5-6</td>
</tr>
<tr>
<td>5.6 Phase II Specifications</td>
<td>5-7</td>
</tr>
<tr>
<td>5.6.1 Polygon descriptor record</td>
<td>5-7</td>
</tr>
<tr>
<td>5.6.2 Polygon limit record</td>
<td>5-8</td>
</tr>
<tr>
<td>5.6.3 Polygon digitized boundary records</td>
<td>5-8</td>
</tr>
<tr>
<td>5.7 Program Output</td>
<td>5-9</td>
</tr>
<tr>
<td>5.7.1 The control card</td>
<td>5-9</td>
</tr>
<tr>
<td>5.7.2 The minimum and maximum record</td>
<td>5-9</td>
</tr>
<tr>
<td>5.7.3 The boundary record</td>
<td>5-9</td>
</tr>
</tbody>
</table>
## CONTENTS (contd)

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.0</td>
<td>CHAIN</td>
<td>6-1</td>
</tr>
<tr>
<td>6.1</td>
<td>Introduction</td>
<td>6-1</td>
</tr>
<tr>
<td>6.2</td>
<td>Methodology</td>
<td>6-1</td>
</tr>
<tr>
<td>6.3</td>
<td>Program</td>
<td>6-4</td>
</tr>
<tr>
<td>6.3.1</td>
<td>Input</td>
<td>6-4</td>
</tr>
<tr>
<td>6.3.2</td>
<td>Output</td>
<td>6-4</td>
</tr>
<tr>
<td>6.4</td>
<td>Overall Specifications</td>
<td>6-6</td>
</tr>
<tr>
<td>6.5</td>
<td>Phase I Specifications</td>
<td>6-6</td>
</tr>
<tr>
<td>6.5.1</td>
<td>Map tolerance record</td>
<td>6-6</td>
</tr>
<tr>
<td>6.5.2</td>
<td>Control reference points</td>
<td>6-7</td>
</tr>
<tr>
<td>6.5.3</td>
<td>Digitized reference points</td>
<td>6-7</td>
</tr>
<tr>
<td>6.6</td>
<td>Phase II Specifications</td>
<td>6-7</td>
</tr>
<tr>
<td>7.0</td>
<td>DIME AREA CENTROID SYSTEM (DACS)</td>
<td>7-1</td>
</tr>
<tr>
<td>7.1</td>
<td>Introduction</td>
<td>7-1</td>
</tr>
<tr>
<td>7.2</td>
<td>Methodology</td>
<td>7-1</td>
</tr>
<tr>
<td>7.3</td>
<td>Program</td>
<td>7-2</td>
</tr>
<tr>
<td>7.3.1</td>
<td>Input</td>
<td>7-5</td>
</tr>
<tr>
<td>7.3.2</td>
<td>Output</td>
<td>7-5</td>
</tr>
<tr>
<td>7.4</td>
<td>Phase I Specifications</td>
<td>7-6</td>
</tr>
<tr>
<td>7.5</td>
<td>Phase II Specifications</td>
<td>7-9</td>
</tr>
<tr>
<td>7.6</td>
<td>Phase III Specifications</td>
<td>7-10</td>
</tr>
<tr>
<td>8.0</td>
<td>POLYGON INTERSECTION OVERLAY SYSTEM (PIOS)</td>
<td>8-1</td>
</tr>
<tr>
<td>8.1</td>
<td>Introduction</td>
<td>8-1</td>
</tr>
<tr>
<td>8.2</td>
<td>Methodology</td>
<td>8-1</td>
</tr>
<tr>
<td>8.3</td>
<td>Program</td>
<td>8-6</td>
</tr>
<tr>
<td>8.4</td>
<td>Input Files</td>
<td>8-9</td>
</tr>
</tbody>
</table>
## CONTENTS (contd)

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Output Files</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>8.5</td>
<td>File 11 — major polygon file</td>
<td>8-10</td>
</tr>
<tr>
<td></td>
<td>File 12 — minor polygon file</td>
<td>8-12</td>
</tr>
<tr>
<td></td>
<td>Record 2 major/minor - min/max record</td>
<td>8-13</td>
</tr>
<tr>
<td></td>
<td>Record 3 to 3+ N/4 pairs major/minor record</td>
<td>8-14</td>
</tr>
<tr>
<td>8.6</td>
<td>End of file records</td>
<td>8-14</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Output</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>8.6</td>
<td>Record 1</td>
<td>8-16</td>
</tr>
<tr>
<td></td>
<td>Record 2 thru 2+ N/4</td>
<td>8-16</td>
</tr>
</tbody>
</table>
# FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>LUMIS Land Use Management Information System — Organization of Coordinate Oriented Program Documentation</td>
<td>1-3</td>
</tr>
<tr>
<td>2</td>
<td>Raw Digitized Map File Control Card Layout for Map Cases 1 and 2</td>
<td>2-3</td>
</tr>
<tr>
<td>3</td>
<td>Raw Digitized Map File Layout Resulting from Digitizing Map Case 1</td>
<td>2-5</td>
</tr>
<tr>
<td>4</td>
<td>Raw Digitized Map File Layout Resulting from Digitizing Map Case 2</td>
<td>2-8</td>
</tr>
<tr>
<td>5</td>
<td>Geographic Position Determination of Control Point 1</td>
<td>3-3</td>
</tr>
<tr>
<td>6</td>
<td>SPC Calculation for Land Use Control Point</td>
<td>3-5</td>
</tr>
<tr>
<td>7</td>
<td>State Plane Coordinate CALculation (SPCCAL)</td>
<td>3-6</td>
</tr>
<tr>
<td>8</td>
<td>Deck Set-Up for ROTATE Showing Multiple Map Models</td>
<td>5-5</td>
</tr>
<tr>
<td>9</td>
<td>Block Chaining</td>
<td>6-3</td>
</tr>
<tr>
<td>10</td>
<td>Deck Set-Up for CHAIN for Multiple Block Boundaries</td>
<td>6-5</td>
</tr>
<tr>
<td>11</td>
<td>Sample Block Boundary File Layout</td>
<td>6-8</td>
</tr>
<tr>
<td>12</td>
<td>Boundary Segment Dividing Census Tract</td>
<td>7-3</td>
</tr>
<tr>
<td>13</td>
<td>DACS Edited Polygons</td>
<td>7-3</td>
</tr>
<tr>
<td>14</td>
<td>PIOS Stripping Method</td>
<td>8-3</td>
</tr>
<tr>
<td>15</td>
<td>Example of Alternative Block and Land Use Intersections</td>
<td>8-5</td>
</tr>
<tr>
<td>16</td>
<td>An Example of Identifying Unique Polygons Created by Intersecting Blocks and Land Use</td>
<td>8-5</td>
</tr>
<tr>
<td>17</td>
<td>Reconstruction of Minor Polygons</td>
<td>8-7</td>
</tr>
<tr>
<td>18</td>
<td>Flow Chart PIOS Program</td>
<td>8-8</td>
</tr>
<tr>
<td>19</td>
<td>Major or Minor Polygon File Required by PIOS</td>
<td>8-11</td>
</tr>
</tbody>
</table>
## APPENDICES

<table>
<thead>
<tr>
<th>Appendix</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>SPCCAL Program Listing</td>
<td>A-I-1</td>
</tr>
<tr>
<td>II</td>
<td>Plotter Editing Program Listing General Map Case</td>
<td>A-II-1</td>
</tr>
<tr>
<td>III</td>
<td>Plotter Editing Program Listing MMS Map Case</td>
<td>A-III-1</td>
</tr>
<tr>
<td>IV</td>
<td>ROTATE Program Listing</td>
<td>A-IV-1</td>
</tr>
<tr>
<td>V</td>
<td>CHAIN Program Listing</td>
<td>A-V-1</td>
</tr>
<tr>
<td>VI</td>
<td>DACS Program Listing</td>
<td>A-VI-1</td>
</tr>
<tr>
<td>VII</td>
<td>PIOS Program Listing</td>
<td>A-VII-1</td>
</tr>
<tr>
<td>VIII</td>
<td>Major Polygon Data Report Program Listing</td>
<td>A-VIII-1</td>
</tr>
<tr>
<td>IX</td>
<td>Minor Polygon Data Report Program Listing</td>
<td>A-IX-1</td>
</tr>
</tbody>
</table>
ABSTRACT

The LUMIS program has designed an integrated geographic information system to assist program managers and planning groups in metropolitan regions. Described is the series of computer software programs and procedures involved in data base construction using the census DIME file and point-in-polygon architectures.

The system is described in two parts: (1) instructions to operators with regard to digitizing and editing procedures, and (2) application of the four data base construction algorithms: ROTATE (to achieve map registration), CHAIN (to assure the topological integrity of polygon files), DACS (DIME AREA CENTROID SYSTEM alternative to CHAIN), and PIOS (the polygon intersection overlay system for tabulating land use acreages within administrative districts).
LUMIS

LAND USE MANAGEMENT INFORMATION SYSTEM

Coordinate Oriented Program Documentation

1.0 INTRODUCTION

1.1 The System

The LUMIS project considers two distinct data types identified as nominal and ordinal. Nominal data refers to that information which is geographically located using a political addressing system such as the individual house and street address. Ordinal data is data geographically referenced using a classical X and Y coordinate system such as latitude and longitude. To reconcile nominal and ordinal data the LUMIS provides users with a set of Census Bureau related procedures that aggregate street addressed data to politically defined geographical zones. Those nominally labeled zones are then ordinally identified through the process of digitization with X and Y coordinates describing their perimeter.

1.2 The Process

Once the coordinate files have been produced for both the nominal and ordinal polygons the LUMIS allows for the comparison of associated data bases. Before this comparison takes place LUMIS first requires that the raw digitized coordinates be converted to state plane coordinates and that they be subjected to a graphic edit.

The graphic edit allows for the correction of a variety of syntax errors possible in the digitizing procedures. After those errors have been corrected the coordinate files are then passed to the CHAIN and ROTATE programs.

ROTATE produces a geographically true X and Y coordinate file from raw digitized or converted coordinate data referenced to any standard map system. That is done to transform coordinates oriented to a digitizing table to coordinates oriented to true ground position. In accomplishing that ROTATE considers both scale and rotation factors.
CHAIN incorporates procedures to link line segments by coordinate locations around unique political or nominally labeled polygons. Those procedures produce multiple records for polygons in which a boundary line has to be used twice, for example, once for each block found on either side of a street.

Where Census Bureau-defined nominal polygons are used such as census tract or census block, it is not necessary to create a unique file of X and Y coordinates through the digitizing procedure to define their perimeters. A set of such coordinates have already been prepared by the Census Bureau through their DIME file creation efforts and those coordinates can be modified for use with LUMIS through the DACS program.

Once the nominal polygon coordinate file and ordinal polygon coordinate file have been completed, the comparison of data associated with each polygon type can take place through the map-like overlay of computer processing. This overlaying of nominal and ordinal polygons takes place in the FIOS program. The output from that effort represents the final LUMIS data product and is used as input to the LUMIS interactive graphics system, proprietary data base management systems, or the installation unique information management systems.

Figure 1 is a flow chart that illustrates the relationships between the various LUMIS coordinate oriented programs and procedures. That same figure also provides a reference to the various sections found in this Report.
Figure 1. LUMIS Land Use Management Information System—Organization of Coordinate Oriented Program Documentation
2.0 DIGITIZING PROCEDURES

2.1 Introduction

Digitizing is the mechanical process by which X and Y coordinates are determined that ordinally describe the outline or perimeter of any mapped polygon. In that process a piece of equipment referred to as a digitizer is used. That machine is similar in appearance and operation to a traditional drafting machine, and consists of a table and cursor.

There are numerous sets of digitizing procedures and techniques available. Their use depends on the equipment employed, the software being used, and the types of uses the coordinates are later subjected to. After reviewing these available procedures and techniques LUMIS evolved its own unique set of digitizing procedures, and they fell into two case types.

The first case type dealt with digitizing ordinal polygons and is always used with mapped and aerial photo data. The second type is for use with nominal polygons and was designed to be used only when Census Bureau DIME coordinate files were not available or nominal polygons other than those defined by the Census Bureau were to be used.

2.2 Case 1: General Map Case

In preparing the data maps from interpreted aerial photos the cartographer has to insure that the numbers identifying unique polygons are consecutive, i.e. \( \text{POLYNO} = 1, 2, 3, 4 \ldots N \). A doughnut polygon or polygon with smaller polygons within it has to be digitized with the inside polygon following the mother polygon. Thus in addition to each polygon being given a sequence number, beginning with one, it is also given a PTYPE or polygon type classification number. All mother polygons are given PTYPE numbers of 10 regardless of their POLYNO or sequence number. All mutually exclusive interior or doughnut polygons are given PTYPE numbers of 11, with inclusive interior polygons being given increasing sequential type numbers. The most interior polygon will always have the highest PTYPE number.

After completing interpretation mapping and polygon identification numbering the map is set up on the digitizer table at any orientation, and
taped down. The digitizer and keypunch are then turned on. The map setup record is established with KSET equal to 1 or other successive numbers depending on the number of map setups for a run. This is recorded on cards or tapes depending on the digitizer being used. Then the map tolerance record is created as indicated in Figure 2 and explained in the text.

2.2.1 Map tolerance record establishes the limits in calibration of the digitized coordinate reference points and the overall perimeter of the polygon to follow.

NPOLY – Total number of polygons for a map setup.

M – Number of control points provided as calculated and digitized reference records.

X0TOL, Y0TOL, and THETOL are user specified tolerance levels in the solution of a digitized origin (X0, Y0 in inches) and the rotation angle (θ - in minutes of arc) between digitized map and accepted reference source.

Sometimes NPOLY is not known until the digitizing is finished for a given setup. Thus the NPOLY value may at times be entered after the digitization is complete. The number of control points (M) is entered for the map, followed by the tolerance levels for the digitizer origin and the rotation angle solutions (X0TOL, Y0TOL, THETOL).

The California State Plane Coordinates (SPC) records follow. Those records contain the SPC's of the M control points, and are computed at a later time according to the procedures outlined for calculating the state plane coordinates.

Control points are selected in the four corner areas as much as possible to avoid problems later in the ROTATE program's least square analysis.

The coordinates of the M control points are digitized in the same order as the SPC coordinates are to be calculated.

Control point cards (digitized reference records) are created by moving the digitizing cursor to each of the map control points, in the proper order, and then pressing the digitizer register button.
### Raw Digitized Map File Control Card Layout for Map Cases 1 and 2

**SPC Digitized Reference Records**

<table>
<thead>
<tr>
<th>COLS</th>
<th>FORMAT</th>
<th>EX.</th>
<th>VAR.</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>12</td>
<td>18</td>
<td>24</td>
</tr>
<tr>
<td>30</td>
<td>36</td>
<td>42</td>
<td>48</td>
</tr>
<tr>
<td>54</td>
<td>60</td>
<td>66</td>
<td>72</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>COLS</th>
<th>FORMAT</th>
<th>EX.</th>
<th>VAR.</th>
</tr>
</thead>
<tbody>
<tr>
<td>16</td>
<td>16</td>
<td></td>
<td></td>
</tr>
<tr>
<td>+24789</td>
<td>+123436</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**California State Plane Coordinate (SPC) Reference Records**

<table>
<thead>
<tr>
<th>COLS</th>
<th>FORMAT</th>
<th>EX.</th>
<th>VAR.</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>12</td>
<td>18</td>
<td>24</td>
</tr>
<tr>
<td>30</td>
<td>36</td>
<td>42</td>
<td>48</td>
</tr>
<tr>
<td>54</td>
<td>60</td>
<td>66</td>
<td>72</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>COLS</th>
<th>FORMAT</th>
<th>EX.</th>
<th>VAR.</th>
</tr>
</thead>
<tbody>
<tr>
<td>19</td>
<td>19</td>
<td>24</td>
<td></td>
</tr>
<tr>
<td>44349278</td>
<td>42471232</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Map Setup Record**

<table>
<thead>
<tr>
<th>COLS</th>
<th>FORMAT</th>
<th>EX.</th>
<th>VAR.</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>10</td>
<td>20</td>
<td>30</td>
</tr>
<tr>
<td>40</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>COLS</th>
<th>FORMAT</th>
<th>EX.</th>
<th>VAR.</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>15</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>30,3</td>
<td>30,3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>500</td>
<td>.500</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Map Tolerance Record**

<table>
<thead>
<tr>
<th>COLS</th>
<th>FORMAT</th>
<th>EX.</th>
<th>VAR.</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>15</td>
<td>7</td>
<td>KSET</td>
</tr>
</tbody>
</table>

---

Figure 2. Raw Digitized Map File Control Card Layout for Map Cases 1 and 2
2. 2. 2 Control reference points are calculated from an existing reference source by the user. They are given in tenths of a foot without a decimal point. In this case the control points are in California Zone 7 State Plane Coordinates, but any legitimate earth coordinate system is acceptable. One through M coordinates are coded with four or less pairs per card.

Digitized reference points are entered in the same order as the control reference points in thousands of an inch, without the decimal point. One through M coordinates are coded with six or less pairs per card.

Digitization of the map polygons is then performed. (See Figure 3.) For each polygon of the map setup which corresponds to the initial setup cards of Figure 2, a polygon designation record is created using the keypunch machine directly. A polygon limit record is then created.

A polygon description record, polygon limit record, and polygon boundary record are required for each polygon digitized (NPOLY) in a map setup (KSET).

2. 2. 3 Polygon description record contains control information for each polygon:

MAPNO: Map number for the digitized polygon.

MAPOL: Total number of polygons on a map. This could, but does not necessarily equal NPOLY (number of polygons in a map setup).

LUCODE: Identifying information code.

POLYNO: Sequence number of polygon, 1 to MAPOL.

PTYPE: Designates doughnut polygons, or polygons within polygons; normal polygons are digitized number 10. Numbers larger than 10 indicate succeeding doughnut polygons.

2. 2. 4 Polygon limit record. One limit record is required for each polygon.

XMIN, YMIN, XMAX, YMAX: The minimum X, minimum Y, maximum X, and maximum Y digitized coordinates, respectively, of the polygon.

NP: Total number of digitized coordinates in a polygon.
Figure 3. Raw Digitized Map File Layout Resulting from Digitizing Map Case 1
To create the polygon limit record, the digitizing cursor is set just outside of the left-most border of the polygon to be digitized. The section "left-most" implies the smallest X coordinate as defined by the digitizer table. Only this minimum X coordinate is digitized and not the Y. It must be insured that this XMIN is less than any X coordinate contained on the polygon boundary.

Next the cursor is placed on the digitizer table just below the lowest polygon point. Just the Y coordinate was digitized as YMIN.

Then the cursor is placed just to the right of the right-most polygon point and the X coordinate digitized as XMAX and finally the cursor is placed above the highest polygon point and the Y coordinate digitized as YMAX. If the digitizing cursor has no provision for entering only one component of a pair, it would be a simple matter to read the coordinate from the digital read out on the console and then enter it by hand on the terminal or keypunch.

It should be noted that the output polygon limit record contains the SPC minima and maxima, not the transformed digitized minima and maxima (if the rotation angle is large enough, the two sets of minima and maxima may be the same).

The number of points (NP) for the polygon is taken from the map (if properly prepared) or by counting the number of polygon digitized boundary records when finished. If this number is N, it is multiplied as 6 by (N-1) and this product added to the number of pairs of coordinates on the list record to obtain NP. The suggested alternative is for the photo interpreter who creates the map to indicate the digitization points at a density sufficient to adequately reflect line curvature. In so doing, he could indicate the point count before digitization commenced.

Finally, the polygon digitized boundary records are created either by running the cursor counterclockwise around the polygon boundary and digitizing at the preselected points, or at the operators' discretion. The polygon is closed by digitizing the first point again at the finish.

2.2.5 Polygon digitized boundary records. One set of digitized coordinates is required for each polygon.

XD(1), YD(1), ---XD(NP), YD(NP): Digitized coordinates of polygon.
2. 3 Case 2: MMS Map Case

In digitizing maps of the Case 2 type users largely follow Case 1 procedures. Generally only four (4) control points are used for the MMS maps (M=4) since these maps were compiled on the State Plane Coordinate base and hence a more accurate transformation from digitizer coordinates to SPC's can be effected than with other map projections.

The control points are located and their State Plane Coordinates determined by finding the SPC tic marks on the base map's margins, and connecting the similar tics with a straight edge and hard lead (4H) pencil. Then the four intersection points are digitized and the SPC's read off the map margins. If there is an insufficient number of SPC tic marks on the map base, users should select a point in each of the four corners that can be easily identified on a USGS quad, and the SPC's identified according to the method described for calculating the State Plane Coordinates.

Digitization of the street segments is then performed, in any arbitrary order and direction. See Figure 4. All street segments are digitized for each tract, and each tract is a unit unto itself, i.e., each tract contains the header records of Figure 2 (except the map setup records, which in this case indicate the number of tracts which are to be processed). Thus each tract has a map tolerance record where NPOLY is the number of street segments in the tract. The SPC reference and SPC digitized reference records follow for each census tract, even though those records may have been precisely the same for several tracts on the same MMS map.

Blocks are digitized as units in a tract, therefore geographic reference control points must be established per each tract. Each map contains one or more tracts. A map tolerance record, control reference record and digitized reference record are also required.

2. 3.1 A map tolerance record establishes the limits in calibration of the digitized coordinate reference points and the overall perimeter of the polygon to follow.

NPOLY — Total number of blocks in a tract.

M — Number of control points provided as calculated and digitized reference records.
Figure 4. Raw Digitized Map File Layout Resulting from Digitizing Map Case 2
XOTOL, YOTOL, and THETOL are user specified tolerance levels in the solution of digitized origin (X0, Y0 in inches) and the rotation angle (α—in minutes of arc) between digitized map and accepted reference source.

2.3.2 Control reference points are calculated from an existing reference source by the user. They are given in tenths of a foot without a decimal point. In the LUMIS test case the control points were in California Zone 7 State Plane Coordinates, but any legitimate earth coordinate system is acceptable. One through M coordinates are coded with four or less pairs per card.

2.3.3 Digitized reference points are entered in the same order as the control reference points in thousands of an inch, without the decimal point. One through M coordinates are coded with six or less pairs per card. Digitizing the MMS maps can be performed by one person as the data maps were if the census areal units are prepunched by a simple FORTRAN program. Then the digitizing operator need only to place the punched records into the keypunch feeder in the same order as the street segments which he proposed to digitize. Since the LUMIS project involved several thousand census blocks in the Santa Monica Mountains, this procedure was followed to automate and expedite the digitizing procedure. However, for a limited amount of digitizing such as that which might have been performed by the City of Los Angeles to update the census areal units each year, a two-person method was suggested. Figure 4 illustrates the card deck setup for that method.

In the "limited amount of digitizing" technique, the digitizing operator places the cursor on the node at the tail end of the arrow along the street segment. He then calls the block designator on the left of the street segment (137411010) to the person on the keypunch, who then punched this designator. Note the ".0" of the tract number (1374.01) was dropped, as redundant information, a "0" was added to the block number (101), and the tract and block numbers were joined. The implied ".0" of the block number was to allow for future block splitting and necessary decimal designators.

The tract and block number to the right of the street segment are then called out to the keypuncher who punches the number as shown. The keypuncher then presses the space bar. The digitizing operator presses the
digitizing button, thus recording on the card the X, Y coordinates of the starting node. The digitizing operator moves the cursor along the arrow to the ending node of the segment and again presses the button thus recording the digitizing table coordinates of this node. The keypuncher releases this card and sets a new card into the keypunch for the next street segment.

The digitizing operator meanwhile marks this segment on the map so that he will not redigitize it later. The procedure then is repeated, segment by segment, in any arbitrary order, until the tract is finished. Note that the tract right field for segments on the census tract boundary (that is, the designator for an area outside of the tract presently being digitized) is simply left blank. In the case where a census block is entirely within another census block (forming what is termed a "Doughnut Hole"), the block right is also left blank.

Link records for each street segment in a census tract are required.

2.3.4 Street Segment Record. Each record contains the left and right census tract/block for a segment and the beginning and ending coordinates. If a segment is missing a tract/block right, the coordinates are coded but the right position is left blank.

ITLEFT = Tract/block left of street segment (1 to MPOLY)
ITRITE = Tract/block right of street segment (1 to NPOLY)
XL, YL = X and Y coordinates of the beginning made for segment.
XR, YR = X and Y coordinate of the ending node for a segment.
3.0 CALCULATING THE STATE PLANE COORDINATES

3.1 Introduction

The following discussion describes the theory and methodology for calculating the State Plane Coordinates (SPCs) of the control points used in the digitizing process. These procedures are only used for the General Map Cases because the SPCs for the special MMS Map Case are determined at the time of digitization. The tools required for calculating the SPCs include:

1. New, unfolded USGS Quad
2. Digitizer
3. SPCCAL computer program (Appendix I)

3.2 Procedure

Place the new USGS quadrangle on the digitizer table, and align it parallel to the bottom and sides. Use the digitizer to ensure that the X coordinate of the left map edge (not paper edge) differs by no more than 0.001 inch between the top and bottom of the map. (If the paper has stretched, this accuracy cannot be obtained; secure a new quad.) Use the same method for the Y coordinate of the bottom left and right map edges. Then tape the quad securely and mark the control points on the map. Locate the four 2-1/2 minute and longitude tic marks surrounding each of the control points.

It is suggested that these 2-1/2 minute tic marks be used for determining the control points' SPCs instead of the actual SPC tic marks provided on the map margins for two reasons: (1) there is an insufficient number of tic marks on the margin to establish a dependable grid on the topographic map, especially when considering that (2) the SPC grid cannot be linear on a topographic map projection.

The State Plane Coordinates for the 2-1/2 minute intersection tics can then be obtained from the U.S. Coast and Geodetic Survey Plane Coordinate Intersection Tables (2-1/2-minute), SP-327, a copy of which can be obtained from the Los Angeles County Engineer's Office or the Superintendent of Public Documents.
With the SPCs of the four bounding 2-1/2 minute intersections known, they can be designated \((X_1, Y_1), (X_2, Y_2), (X_3, Y_3),\) and \((X_4, Y_4)\) representing the southwest, southeast, northwest, and northeast tic marks respectively. (See Figure 4.) Since the SPCs form a curvilinear coordinate system on the topographic map, a four-way interpretation for the \(X\) and \(Y\) coordinates of each control point can then be made.

Using the digitizer, establish the origin at the first control point and measure (to the nearest .001 inch) the \(X\) and \(Y\) displacements to two diagonal 2-1/2 minute intersections. The four resulting numbers are named \(x_1, x_2, y_1, y_2\), and refer to the segments created by an imaginary perpendicular north-south line drawn through the center of the control point (Figure 5). To obtain the most accurate results, it is best to remeasure the four distances several times (reoriginating on each occasion), then average the results. Repeat this procedure for each control point, proceeding in the same order as the points were digitized on the mylar base map.

### 3.3 Calculations

The four SPC solutions for each of the \(X\) and \(Y\) coordinates of the control points can now be calculated by:

\[
X = X_1 + 2000x_1 + \frac{y_1}{y_1 + y_2} (X_3 - X_1)
\]

\[
X = X_3 + 2000x_1 - \frac{y_2}{y_1 + y_2} (X_3 - X_1)
\]

\[
X = X_2 - 2000x_2 + \frac{y_1}{y_1 + y_2} (X_4 - X_2)
\]

\[
X = X_4 - 2000x_2 - \frac{y_2}{y_1 + y_2} (X_4 - X_2)
\]
Figure 5. Geographic Position Determination of Control Point 1
The SPCCAL program performs the above mathematics. The four X and Y solutions should always agree to within 3 to 30 feet of each other. If the corresponding range of solutions should ever exceed 30 feet, the $x_1$, $y_1$, $x_2$, $y_2$ distances should be remeasured and run through the SPCCAL program until they meet the specified tolerance. The SPCCAL program prints an error message when the tolerance is exceeded. An example of the calculations for the SPCs of a land use control point is exhibited in Figure 6.

The State Plane Coordinates of the control points may then be punched onto cards for inclusion in the ROTATE program (see Figure 7).
LUMIS PROJECT TEST SITE
WOODLAND HILLS

LAND USE MAP 17001

REFERENCE POINT 1: INTERSECTION OF VALLEY CIRCLE BOULEVARD AND NORTHBOUND FEEDER OFF VENTURA FREEWAY

ELEVATION: 900 FEET

GRID TICS

<table>
<thead>
<tr>
<th></th>
<th>SW1</th>
<th>SE2</th>
<th>NW3</th>
<th>NE4</th>
</tr>
</thead>
<tbody>
<tr>
<td>LATITUDE</td>
<td>34° 07' 30&quot;</td>
<td>34° 07' 30&quot;</td>
<td>34° 10'</td>
<td>34° 10'</td>
</tr>
<tr>
<td>LONGITUDE</td>
<td>118° 40'</td>
<td>118° 37' 30&quot;</td>
<td>118° 40'</td>
<td>118° 37' 30&quot;</td>
</tr>
<tr>
<td>X</td>
<td>4085806.8</td>
<td>4098417.5</td>
<td>4085856.4</td>
<td>4098460.8</td>
</tr>
<tr>
<td>Y</td>
<td>4158058.9</td>
<td>4158020.2</td>
<td>4173221.8</td>
<td>4173183.2</td>
</tr>
<tr>
<td>X1 =</td>
<td>4,455 INCHES</td>
<td>X3 - X1 = 49</td>
<td>Y1 - Y2 = 39</td>
<td></td>
</tr>
<tr>
<td>X2 =</td>
<td>1,840 INCHES</td>
<td>X4 - X2 = 43</td>
<td>Y3 - Y4 = 39</td>
<td></td>
</tr>
<tr>
<td>Y1 =</td>
<td>6,610 INCHES</td>
<td>Y1 / \Sigma Y = .8732</td>
<td>X1 / \Sigma X = .7077</td>
<td></td>
</tr>
<tr>
<td>Y2 =</td>
<td>0,960 INCHES</td>
<td>Y2 / \Sigma Y = .1268</td>
<td>X2 / \Sigma X = .2923</td>
<td></td>
</tr>
</tbody>
</table>

\[
\begin{align*}
X & = 4094760 & & Y = 4171251 \\
& = 4094760 & & = 4171251 \\
& = 4094776 & & = 4171274 \\
& = 4094776 & & = 4171274 \\
AVERAGE: & = 4094768 & & 4171262
\end{align*}
\]

Figure 6. SPC Calculation for Land Use Control Point
SPCCAL IS A COMPUTER PROGRAM THAT CALCULATES THE CONTROL POINT SPCs FOR INPUT TO THE ROTATE PROGRAM.

DECK SETUP:

<table>
<thead>
<tr>
<th>COLUMN</th>
<th>VARIABLE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>CARD 1</td>
<td>NUMAP</td>
<td>NUMBER OF MAPS FOR WHICH CONTROL POINT SPCs ARE TO BE CALCULATED IN THIS COMPUTER RUN. (INTEGER)</td>
</tr>
<tr>
<td>CARD 2</td>
<td>MAPNO</td>
<td>MAP NUMBER (EX. 17001) FOR WHICH SPCs ARE TO BE CALCULATED. (INTEGER)</td>
</tr>
<tr>
<td></td>
<td>NCONPT</td>
<td>NUMBER OF CONTROL POINTS. (INTEGER)</td>
</tr>
</tbody>
</table>

CARD 3

SPC COORDINATES OF THE FOUR 2-1/2 MINUTE LATITUDE-LONGITUDE TIC MARKS SURROUNDING THE FIRST CONTROL POINT. (SEE FIGURE (4).) PUNCH THESE COORDINATES (TO ONE DIGIT AFTER THE DECIMAL POINT) IN THE FORM OF X1, X2, X3, X4, Y1, Y2, Y3, Y4, RIGHT JUSTIFIED TO EVERY TENTH COLUMN.

CARD 4

X1, X2, Y3, Y4 (IN THAT ORDER) DIGITIZER MEASURED DISTANCES (IN INCHES) TO THE 2-1/2 MINUTE INTERSECTIONS.

REPEAT ONE SET OF CARDS 3 AND 4 FOR EACH CONTROL POINT. FOR THE SECOND MAP, REPEAT CARD 2 FOLLOWED BY THE NECESSARY SETS OF CARDS 3 AND 4.

Figure 7. State Plane Coordinate CALCulation (SPCCAL)
4.0 EDITING PROCEDURES

4.1 Introduction

Editing the MMS Census Tract and General Map Raw Digitized Map Files is the process through which all syntax errors are removed for input to the CHAIN or ROTATE programs. The procedures for editing the two different map cases are fundamentally the same, but the individual steps are performed with varying emphasis.

The following items are needed in the editing process:

1. Original mylar base map
2. Plotter program and plot of digitized data
3. Printout of raw data
4. Light table

The mylar base is the actual map that was digitized. A listing of the plotter editing program for the general map case is in Appendix II, and for the special MMS map case Appendix III. The editing programs were written for a UNIVAC 1108 Computer using CALCOMP plotter commands.

4.2 General Map

Taking the mylar base map, overlay it on top of each individual plot, and search for obvious errors like missing polygons, truncated or incorrectly digitized boundaries, extraneous lines, off registration, and control point errors. Small gaps and overlaps between polygon boundaries are normal and should not cause concern. Minor boundary errors are best repaired by locating the coordinates of the neighboring polygon's common boundary, and substituting those coordinates in the problem polygon. To use this method, plot the problem border on graph paper marking the location of each coordinate. Note the beginning and end point of the error segment, and locate similar points in the neighboring polygon.

Insert the correct coordinates and rectify the polygon's total number of points (NP) if there was a change.
This method is rather tedious, however, so if the boundary error is serious, it is easier to throw out the polygon and redigitize it as a separate setup within the map.

Often times the raw data cards become shuffled or blank cards manage to mix with the data. These types of errors show up as extraneous lines criss-crossing inside or through polygons. Off-registration is usually caused by the plotter, digitizer, or digitizer person losing origin. In most cases, the error can be corrected by determining the X and Y displacement, pulling the polygon(s) out, and treating them as a separate setup with their control points being a combination of the original control points and the displacement.

If the general map was digitized in several setups rather than all at once, inspect the plot of each setup individually, then combine them (using the plotted control points) over a light table and check for off-registration, bad control points, and missing polygons.

There are a number of errors that the visual inspection of the plotted maps will not find. Among these include incorrect land use codes (LUCODE), number of points (NP), polygon types (PTYPE), and minimum-maximum coordinates. The plotter editing program printout is most useful in spotting these kinds of problems. Its main feature (other than to create the plot) is that it counts the number of points in each polygon, and prints that number out. The most common error by far, is an incorrect NP, and such an error will always abort the ROTATE program. The editing program also prints out the regular polygon information like MAPNO, LUCODE, POLYNO, NP, digitized data, etc., but in an easily read format.

The procedure for editing the NP is simply to scan the printout comparing the computer counted NP against the digitizer's specified NP. This method not only catches wrong NPs, but also out-of-sequence header cards ("polygon designation records") and min-max cards ("polygon limit record"), and also blank cards. As for checking the LUCODE and PTYPE, each polygon in the printout must be individually checked against the mylar base map.

It is very important that no PTYPE errors pass uncorrected, as one error could falsify the polygon overlay area calculations of many neighboring polygons. For the same reason, if a polygon is removed for redigitizing as a separate setup (and all redigitized polygons must become separate setups),
careful note must be taken to determine if it is a polygon PTYPE 11 or greater. If it is, then all the polygons from the previous PTYPE 10 to the last PTYPE 11 or greater in the series (i.e., all polygons within the PTYPE 10) must also be redigitized in the new setup. This is because the polygon overlay program subtracts the area of all PTYPE 11 (or greater) polygons from the polygon they are within. Users should replot and reedit map setups in which there were a lot of errors. Redigitized polygons also should be replotted. Once all this editing is done, the raw digitized map file is ready to be inputted to the ROTATE program.

4.3 MMS Maps

The MMS census tract plotter editing program (Appendix III) produces two plots of each census tract. Along each line segment in the first plot, the sequence number describing the order in which the segment was digitized is printed. In the second plot, the block number of the polygon on either side of the segment is printed. The procedure for editing the MMS raw digitized map file involves the visual comparison of the plotted census tract shapes with the actual base map.

The base map can be overlayed directly on top of the plot, but this method does not work well when the street (line) segments stimulate curved block boundaries as in hilly areas. The kinds of errors encountered include incorrect block numbers, double digitized street segments, missing segments, common nodes greater than tolerance (.045 inch) apart, and incorrectly digitized "doughnut holes". (A doughnut hole census block should be treated like a census tract, i.e., the outside block number left blank.) Between census tracts, checks must be made to insure that the common boundaries are similar. Adding new nodes is accomplished by drawing an X and Y axis through a close existing node, measuring the X and Y displacement to the new node, and adding that displacement to the coordinates of the known node. (An existing node's coordinates are located by referring to the sequence number of its line segment, then finding that line in the editing program printout.) When the entire census tract has been corrected, it should be replotted and reedited again.
4.4 CHAIN

The next step is to run the raw digitized data through the CHAIN program, which will flag all the remaining uncaught errors and print the street segments in the bad census block. The error can be found by closely reexamining the census block's plot, or by manually CHAINing the street segment coordinates in the printout until two nodes are found to be greater than tolerance distance (.045 inch) apart. A common error is to have an incorrectly numbered street segment not only abort the census block it is missing from, but also the census block of the wrongly specified number. When using the CHAIN program for editing purposes, it is a good idea to modify the program so that it does not produce punched cards at each turn-around. When all the blocks finally CHAIN properly, the program can then be run conventionally to convert the digitized street segments to State Plane Coordinates.
5.0 ROTATE

5.1 Introduction

The ROTATE program is written in FORTRAN IV and produces a geographically true X and Y coordinate data base from raw digitized data referenced to any standard map system. These ROTATE corrected coordinates then act as input to PIOS (Polygon Information Overlay System) where polygon intersection statistics are calculated. Raw digitized coordinates such as those corrected by ROTATE are produced when geographically oriented researchers decide to incorporate thematically mapped data, like soils data, geological data or vegetative data into a LUMIS data base. These data maps when digitized are not usually assigned true X and Y coordinates, but rather coordinates that are relative to the digitizer table and the position of the data map to that table. Hence ROTATE's purpose is to adjust those raw digitized coordinates to their relative true ground locations.

5.2 Methodology

To produce geographically true X and Y coordinates the ROTATE program is broken into two major components. The first component operates with the data cards prepared for each unique data map. These initial map setup records describe the geographic coordinates of the digitizer table origin, and the table-to-geographic coordinate system relationship from which a scaling factor and rotation angle are computed. In the first component of ROTATE there is a check that insures control point accuracies meet user specified tolerances.

That check for control point accuracies requires a digitizer table-to-ground coordinate transformation. This transformation is solved by program ROTATE using a least squares solution having four unknown variables. These unknowns are:

1) $X_0$ ................. Control map X axis ground location
2) $Y_0$ ................. Control map Y axis ground location
3) $SCos \theta$ .............. Scaling Factor S times the cosine $\theta$
4) $SSin \theta$ .............. Scaling Factor S times the sine $\theta$
5.2.1 The scaling factor $S$ is expressed in a ground coordinate system of feet per inch on the map.

5.2.2 The rotation angle $\theta$ representing the difference between the map orientation on the digitizing table and true North-South is expressed in degrees, minutes and seconds. The scaling factor and rotation angle are calculated from $SCos \theta$ and $SSin \theta$ in a simple trigonometric equation.

These variables are then used in the least square transformation equation $R$ as follows:

$$
X = X + SNx \quad \text{Transformation Equation}
$$

$$
S = \quad \text{Scaling Factor}
$$

$$
X = (XO) \quad \text{Rotated Coordinates}
$$

$$
(YO)
$$

$$
N = (Cos \theta Sin \theta \quad \text{Rotation Angle}
$$

$$
-Sin \theta Cos \theta)
$$

$$
x = (XDi), i = 1, M \quad \text{Digitized Coordinates}
$$

$$
(YDi)
$$

$$
X = (Xi), i = 1, M \quad \text{Surveyed SPC}
$$

$$
(Yi)
$$

ROTATE uses the least square approach to increase the accuracy of the basic transformation equation. This approach will produce a transformation factor ($SN$) that when applied will change all digitized coordinates to their ground truth equivalents. In the transformation equation, surveyed $X$-$Y$ coordinates ($X$) and digitized $X$-$Y$ coordinates for the control points ($x$) are expressed as arrays, $M$ sets in length.

5.2.3 The surveyed control points $(Xi, Yi, i = 1, M)$ correspond to the digitized control points $(XDi, TDi, i = 1, M)$. In the transformation equation the error between surveyed and digitized control points is averaged using the least squares method.
5.2.4 The transformation equation used in ROTATE creates two equations per control point, one for X and one for Y, so only two control points are required, M = 2. However, the least squares solution can accommodate up to eight points. For greater accuracy the maximum number are suggested. Using the computed transformation factor, each digitized control point X and Y coordinate is entered into the equation and the measured coordinate is solved for. If this computed equivalent of the surveyed X-Y coordinate is not found to be within the used specified tolerances (X0TOL, Y0TOL, or THETOL), a message is printed out by the program indicating the degree of error and processing is stopped.

For example, a tolerance of 0.045 was used in the LUMIS Santa Monica Mountains Study. This was felt appropriate because LUMIS is a block level system and the boundaries should rarely be in error of more than the width of a secondary street, 84 ft. At a scale of approximately 1 inch = 2,000 ft., an error of 0.045 inches on the digitizing table corresponds to a geographical ground error of 90 ft.

Once the equation is formulated and tested the remaining polygons are rotated through angle \( \theta \) and scaled by the factor S. Each map set up must follow this same procedural construction of an equation, testing and processing of the polygons which follow.

5.3 Program

ROTATE written in FORTRAN IV, consists of a main program having two phases, and two subroutines. The subroutines perform matrix inversion and multiplication required by the least square algorithm. The main program itself is divided into two logical phases.

5.3.1 Phase I. The main program first phase calculates the unknown in a least squares equation for each map set-up from the calculated and the digitized control coordinates. The equation is tested by reentering the digitized control points to determine if the calculated control points can be found within user specified tolerances.
5.3.2 Phase II. The second phase reads data points for each polygon, enters them into the equation formed in Phase I. It then makes the adjustment in the digitized coordinates for rotation angle and scaling factor, and writes the adjusted coordinates to an output file.

There are two subroutines used in ROTATE, RATMUL and INVERSE. Subroutine RATMUL multiplies a matrix times its transpose. The regression algorithm performs this task on the original digitized matrix and on its inverse as part of the least squares formula. Subroutine INVERSE defines the inverted matrix of the original digitized data and its transpose.

5.3.2.1 Input. The ROTATE program uses as its input a card file with the following sets of information; the control specifications, the digitized calculated control points, and the digitized polygon min-max and boundary points, Figure 8.

5.3.2.2 Output. The ROTATE program produces as output a listing of transformation solution, rotation angle and a scaling factor, input digitized coordinates for each polygon, a listing and deck of transformed geographic coordinates, and any error condition messages that are invoked.

5.4 Overall Specifications

The polygons are entered as sets (KSET) within an established geographic reference system, or map setup. One map may be coded as one or more setups depending on the number of times a reference is required. The first card in the deck is coded as follows:

```
COL 5

4
KSET
```

5.5 Phase I Specifications

Phase I requires as input the following information: map tolerance record, control reference record, digitized reference record.

5-4
Figure 8. Deck Setup for ROTATE Showing Multiple Map Models
5.5.1 Map tolerance record establishes the limits in calibration of the digitized coordinate reference points and the overall parameter of the polygon to follow.

NPOLY - Total number of polygons for a map setup.

M - Number of control points provided as calculated and digitized reference records.

X0TOL, Y0TOL, THETOL, user specified tolerance levels in the solution of digitized origin (X0, Y0 in inches) and the rotation angle (θ - in minutes of arc) between digitized map and accepted reference source.

<table>
<thead>
<tr>
<th>COL</th>
<th>5</th>
<th>10</th>
<th>20</th>
<th>30</th>
<th>40</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>48</td>
<td>7</td>
<td>.045</td>
<td>.045</td>
<td>.500</td>
</tr>
<tr>
<td>NPOLY</td>
<td>M</td>
<td>X0TOL</td>
<td>Y0TOL</td>
<td>THETOL</td>
<td></td>
</tr>
</tbody>
</table>

5.5.2 Control reference points are calculated from an existing reference source by the user. They are given in tenths of a foot without a decimal point. In this case the control points are in California Zone 7 State Plane Coordinates, but any legitimate earth coordinate system is acceptable. One through M coordinates are coded with four or less pairs per card in the following format:

<table>
<thead>
<tr>
<th>COL</th>
<th>9</th>
<th>18</th>
<th>27</th>
<th>36</th>
<th>ETC.</th>
<th>72</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>44349278</td>
<td>42471232</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>X(1)</td>
<td>Y(1)</td>
<td>X(2)</td>
<td>Y(2)</td>
<td>ETC.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

5.5.3 Digitized reference points are entered in the same order as the control reference points in thousands of an inch, without the decimal point. One through M coordinates are coded with six or less pairs per card in the following format:

<table>
<thead>
<tr>
<th>COL</th>
<th>6</th>
<th>12</th>
<th>18</th>
<th>24</th>
<th>30</th>
<th>36</th>
<th>ETC.</th>
<th>72</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>+24789</td>
<td>+123436</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>XD(1)</td>
<td>YD(1)</td>
<td>XD(2)</td>
<td>YD(2)</td>
<td>ETC.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Upon completion of Phase I the user will receive a message as to the status of the job. The message indicates the calculated scaling factor and rotation angle. If the transform equation fails the tolerance checks, an error message appears.

Problems that arise in rotating data points include syntax errors and/or incorrectly digitized or poorly placed control points. In either case, it is suggested that the program's punched output be deleted until a normal termination is achieved in the first phase.

There are two possible means to correct the error condition without redigitizing the control points: either reduced the number of data points or increase the acceptable tolerances. Since a minimum of two points are required, it is possible to discard all but the last two data points. In the cases where variation is slight, the acceptable tolerance can be increased.

In extreme cases of inaccuracy or when the data points are poorly placed, i.e., too close together or in a linear arrangement, they should be recoded. Recoded control points should make use of the old points, adding only those control points necessary to correct the problems of improper placement.

5.6 Phase II Specifications

If Phase I is completed without errors, Phase II begins with a polygon description record, polygon limit record, and polygon boundary record. One set of these records are required for each polygon digitized (NPOLY) in a map setup (KSET).

5.6.1 Polygon descriptor record contains control information for each polygon:

MAPNO: Map number for the digitized polygon.

MAPOL: Total number of polygons on a map. This could, but does not necessarily equal NPOLY (number of polygons in a map setup).

LUCODE: Identifying information code.
POLYNO: Sequence number for polygon, 1 to MAPOL

PTYPE: Designates doughnut polygons, or polygon without polygons; normal polygons are digitized number 10. Numbers larger than 10 indicate succeeding doughnut polygons.

<table>
<thead>
<tr>
<th>COL</th>
<th>5</th>
<th>10</th>
<th>13</th>
<th>17</th>
<th>19</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>11005</td>
<td>62</td>
<td>A05</td>
<td>104</td>
<td>12</td>
</tr>
<tr>
<td>MAPNO</td>
<td>MAPOL</td>
<td>LUCODE</td>
<td>POLYNO</td>
<td>PTYPE</td>
<td></td>
</tr>
</tbody>
</table>

5.6.2 Polygon limit record. One limit record is required for each polygon. It should be noted that the output polygon limit record contains the SPC minima and maxima, not the transformed digitized minima and maxima (if the rotation angle is large enough, the two sets of minima and maxima may be the same). These limits (in SPC's) are important since they "box in" the map polygons for efficiency in the PIOS overlay techniques.

XMIN, YMIN, XMAX, YMAX: The minimum X, minimum Y, maximum X, and maximum Y digitized coordinates respectively of the polygon.

NP: Total number of digitized coordinates in polygon. Format follows:

<table>
<thead>
<tr>
<th>COL</th>
<th>6</th>
<th>12</th>
<th>18</th>
<th>24</th>
<th>27</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>+02222</td>
<td>+23150</td>
<td>+04444</td>
<td>25179</td>
<td>476</td>
</tr>
<tr>
<td>XMIN</td>
<td>YMIN</td>
<td>XMAX</td>
<td>YMAX</td>
<td>NP</td>
<td></td>
</tr>
</tbody>
</table>

5.6.3 Polygon digitized boundary records. One set of digitized coordinates is required for each polygon.

XD(1), YD(1), ---XD(NP), YD(NP): Digitized coordinates of polygon.

Format follows:

<table>
<thead>
<tr>
<th>COL</th>
<th>6</th>
<th>12</th>
<th>18</th>
<th>24</th>
<th>30</th>
<th>36</th>
<th>ETC....</th>
<th>72</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>+02256</td>
<td>+23190</td>
<td>XD(1)</td>
<td>YD(1)</td>
<td>XD(2)</td>
<td>YD(2)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

5-8
5.7 **Program Output**

The program produces a card file which can later be used directly in PIOS as the minor polygon file. The cards are grouped by polygon with a control card, minimum-maximum card and polygon record description.

5.7.1 The **control card** contains the following data formatted as indicated:

<table>
<thead>
<tr>
<th>Column</th>
<th>Format</th>
<th>Descriptor</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-5</td>
<td>I5</td>
<td>Polygon Descriptor (MAPNO)</td>
</tr>
<tr>
<td>6-10</td>
<td>I5</td>
<td>Number of Polygon on a Map (MAPOL)</td>
</tr>
<tr>
<td>11-13</td>
<td>A3</td>
<td>Land Use Code Identification (LUCODE)</td>
</tr>
<tr>
<td>14-17</td>
<td>I4</td>
<td>Sequence Number of Polygon (POLYNO)</td>
</tr>
<tr>
<td>18-19</td>
<td>I2</td>
<td>Polygon Type, complete or Doughnut Polygon (PTYPE)</td>
</tr>
</tbody>
</table>

5.7.2 The **minimum and maximum record** is formatted as follows:

<table>
<thead>
<tr>
<th>Column</th>
<th>Format</th>
<th>Descriptor</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-9</td>
<td>I9</td>
<td>Minimum X coordinate value of the Polygon (XMIN)</td>
</tr>
<tr>
<td>10-18</td>
<td>I9</td>
<td>Minimum Y coordinate value of the Polygon (YMIN)</td>
</tr>
<tr>
<td>19-27</td>
<td>I9</td>
<td>Maximum X coordinate value of the Polygon (XMAX)</td>
</tr>
<tr>
<td>28-36</td>
<td>I9</td>
<td>Maximum Y coordinate value of the Polygon (YMAX)</td>
</tr>
<tr>
<td>37-41</td>
<td>I5</td>
<td>The number of coordinate pairs describing a Polygon (N)</td>
</tr>
</tbody>
</table>

5.7.3 The **boundary record** is recorded as X and Y values in a 8I9 format, four pairs per card. The number of cards to follow is determined by the data points N divided by four rounded up.
6.0 CHAIN

6.1 Introduction

CHAIN was created to prepare a coordinate data base, referenced to a standard geographic coordinate, as input to PIOS (Polygon Information Overlay System) from a raw digitized street network and political boundary such as a metropolitan map series.

6.2 Methodology

CHAIN also makes use of the basic least squares transformation equations with four unknowns developed for ROTATE. The table-to-ground equations use the same unknowns as those in ROTATE;

1) $X_0$ . . . . . . Control map X axis ground location
2) $Y_0$ . . . . . . Control map Y axis ground location
3) $SCos\theta$ . . . . Scaling Factor $S$ times the cosine $\theta$
4) $SSin\theta$ . . . . Scaling Factor $S$ times the sine $\theta$

The scaling factor $S$ is expressed in a ground coordinate system of feet per inch on the map.

Two to four control points are required by the CHAIN Program. Since the original MMS maps are more accurate than ordinal input maps used by ROTATE, four coordinates are sufficient to maintain the same level of accuracy as eight in ROTATE. The least squares solution is again tested against user specified tolerances.

CHAIN also incorporates an algorithm to link segments by node coordinate around digitized blocks and to produce multiple records for those units which a boundary segment must use twice.

CHAIN must link digitized street segment nodes and check the polygon for completeness before segments can be transformed. Thus CHAIN will correct for minor digitizing errors or point to the node where major errors exist.
The technique used to build the block file starts by digitizing records for each link of the block shown in the top half of Figure 9.

Looking at Blocks 101, 102, 109, and 110, the coordinates are digitized in the following manner:

<table>
<thead>
<tr>
<th>Block Lft</th>
<th>Block Rt</th>
<th>Begin</th>
<th>X-Y</th>
<th>FHD</th>
<th>X-Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>1) 101</td>
<td></td>
<td>+03759</td>
<td>+04222</td>
<td>+03900</td>
<td>+04250</td>
</tr>
<tr>
<td>2) 101</td>
<td></td>
<td>+03910</td>
<td>+04260</td>
<td>+03915</td>
<td>+04500</td>
</tr>
<tr>
<td>3) 109</td>
<td></td>
<td>+03910</td>
<td>+04533</td>
<td>+03902</td>
<td>+04800</td>
</tr>
<tr>
<td>4) 102</td>
<td></td>
<td>+03688</td>
<td>+04292</td>
<td>+03710</td>
<td>+04262</td>
</tr>
<tr>
<td>5) 102</td>
<td>110</td>
<td>+03773</td>
<td>+04503</td>
<td>+08523</td>
<td>+04472</td>
</tr>
<tr>
<td>6) 110</td>
<td>109</td>
<td>+03756</td>
<td>+04522</td>
<td>+03756</td>
<td>+04725</td>
</tr>
<tr>
<td>7) 102</td>
<td>101</td>
<td>+03733</td>
<td>+04251</td>
<td>+03766</td>
<td>+04509</td>
</tr>
<tr>
<td>8) 101</td>
<td>109</td>
<td>+03937</td>
<td>+04522</td>
<td>+03777</td>
<td>+04502</td>
</tr>
</tbody>
</table>

In the Chaining Program each record which has two sides, (5) (6) (7) and (8) are "reversed - duplicated," that is the left block becomes the right block, from and to coordinates are reversed, and the segment is duplicated.

Segments not under consideration are then "erased" from the duplicated file. These records, tract boundary records, will be considered at a later time.

The file is sorted by the left block producing the file shown in the bottom half of Figure 9. The coordinates are then "matched" by ending and beginning nodes for each block according to user specified mapping tolerance (X0TOL, Y0TOL). Cul-de-sacs fall at the end of each block segments. They are retained only for graphic presentation and do not provide any information to PIOS.

The coordinates accepted by the CHAIN edit program are averaged, and transformed to state plane coordinates.
SAMPLE MAP SHOWING SEGMENT IDENTIFICATION FOR BLOCK 101

<table>
<thead>
<tr>
<th>BLOCK LEFT</th>
<th>BLOCK RT</th>
<th>FROM X-Y</th>
<th>TO X-Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1) 101</td>
<td></td>
<td>+03759</td>
<td>+04222</td>
</tr>
<tr>
<td>(1) 101</td>
<td></td>
<td>+03900</td>
<td>+04250</td>
</tr>
<tr>
<td>(2) 101</td>
<td>109</td>
<td>+03910</td>
<td>+04260</td>
</tr>
<tr>
<td>(2) 101</td>
<td></td>
<td>+04260</td>
<td>+03915</td>
</tr>
<tr>
<td>(8) 101</td>
<td>109</td>
<td>+03937</td>
<td>+04522</td>
</tr>
<tr>
<td>(8) 101</td>
<td></td>
<td>+03727</td>
<td>+04502</td>
</tr>
<tr>
<td>(REV) 101</td>
<td>102</td>
<td>+03766</td>
<td>+04509</td>
</tr>
<tr>
<td>(REV) 101</td>
<td></td>
<td>+04509</td>
<td>+043733</td>
</tr>
<tr>
<td>(4) 102</td>
<td></td>
<td>+03688</td>
<td>+04292</td>
</tr>
<tr>
<td>(4) 102</td>
<td></td>
<td>+03710</td>
<td>+04262</td>
</tr>
<tr>
<td>(7) 102</td>
<td>101</td>
<td>+03733</td>
<td>+04251</td>
</tr>
<tr>
<td>(7) 102</td>
<td></td>
<td>+04251</td>
<td>+03766</td>
</tr>
<tr>
<td>(5) 102</td>
<td>110</td>
<td>+03773</td>
<td>+04503</td>
</tr>
<tr>
<td>(5) 102</td>
<td></td>
<td>+04503</td>
<td>+03523</td>
</tr>
</tbody>
</table>

BLOCK CHAINING TECHNIQUE USED BY THE CHAIN PROGRAM FOR SAMPLE MAP SHOWN

Figure 9. Block Chaining
6.3 Program

CHAIN makes use of block boundary segments digitized as single lines separating two units left and right. The program checks for a complete set of boundary elements surrounding each block and splits the boundary segments into left and right corresponding to their respective blocks. It then transforms the block file coordinates from a table coordinate file to a standard geographic coordinate system, maintaining scale and table rotation limit checks.

The CHAIN Program is written in three phases.

Phase I:
Determines, for each census tract, the geographic coordinates of the digitized X-Y origin, map scale, rotating angle to convert digitized coordinates to a standard reference system and tests to determine if control points exceed user specified tolerance.

Phase II:
Reverses and duplicates segments which are boundaries for a block coded with left and right geographic codes and produces a complete boundary file for each block.

Phase III:
Transforms the standard scale and rotation angle of the polygon produced in Phase II, and converts all records for each tract so they will conform to the geographic reference system established in Phase I.

Control coordinates must fall within accepted limits and blocks must chain, i.e., start and end on the same node, to be continued into Phase III, transformation. Failure of the file Phase I or II will generate an error message and stop processing.

6.3.1 Input. The control specifications, digitized and calculated control points, and digitized segment file are shown in Figure 10.

6.3.2 Output. Output includes a listing of the transformation solution, rotation and scaling factors, input digitized coordinates, listing and deck of transformed geographic coordinates, and any error messages that are invoked.
Figure 10. Deck Setup for Chain for Multiple Block Boundaries
Figure 10. Deck Setup for Chain for Multiple Block Boundaries
6.4 **Overall Specifications**

Blocks are digitized as units in a tract, therefore geographic reference control points must be established per each tract. Each map contains one or more tracts.

<table>
<thead>
<tr>
<th>COL</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>KSET</td>
</tr>
</tbody>
</table>

6.5 **Phase I Specifications**

Phase I requires as input the following information: map tolerance record, control reference record, digitized reference record.

6.5.1 **Map tolerance record** establishes the limits in calibration of the digitized coordinate reference points and the overall parameter of the polygon to follow.

- **NNPOLY** - Total number of blocks in a tract.
- **M** - Number of control points provided as calculated and digitized reference records.
- **XOTOL, YOTOL, THETOL** - User specified tolerance levels in the solution of digitized origin (X0, Y0 in inches) and the rotation angle (θ - in minutes of arc) between digitized map and accepted reference source.

<table>
<thead>
<tr>
<th>COL</th>
<th>5</th>
<th>10</th>
<th>20</th>
<th>30</th>
<th>40</th>
</tr>
</thead>
<tbody>
<tr>
<td>48</td>
<td>7</td>
<td>.045</td>
<td>.045</td>
<td>.500</td>
<td></td>
</tr>
<tr>
<td>NNPOLY</td>
<td>M</td>
<td>XOTOL</td>
<td>YOTOL</td>
<td>THETOL</td>
<td></td>
</tr>
</tbody>
</table>

6-6
6.5.2 Control reference points are calculated from an existent reference source by the user. They are given in tenths of a foot without a decimal point. In this case the control points are in California Zone 7 State Plane Coordinates, but any legitimate earth coordinate system is acceptable. One through M coordinates are coded with four or less pairs per card in the following format:

<table>
<thead>
<tr>
<th>COL</th>
<th>9</th>
<th>18</th>
<th>27</th>
<th>36</th>
<th>ETC....</th>
<th>72</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>44349278</td>
<td>424712321</td>
<td>X(1)</td>
<td>Y(1)</td>
<td>X(2)</td>
<td>Y(2)</td>
</tr>
</tbody>
</table>

6.5.3 Digitized reference points are entered in the same order as the control reference points in thousands of an inch, without the decimal point. One through M coordinates are coded with six or less pairs per card in the following format:

<table>
<thead>
<tr>
<th>COL</th>
<th>6</th>
<th>12</th>
<th>18</th>
<th>24</th>
<th>30</th>
<th>36</th>
<th>ETC...</th>
<th>72</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>+24789</td>
<td>+12345</td>
<td>XD(1)</td>
<td>YD(1)</td>
<td>XD(2)</td>
<td>YD(2)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Upon completion of the Phase I, the user will receive a message as to the status of the job. The message indicates the calculated scaling factor and rotation angle.

6.6 Phase II Specifications

If Phase I is completed without errors, Phase II requires link records for each street segment in a census tract.

Street segment record. Each record contains the left and right census tract/block for a segment and the beginning and ending coordinates. If a segment is missing a tract/block right, the coordinates are coded but the right position is left blank as shown in Figure 11.

ITLEFT = Tract/block left of street segment (1 to MPOLY)

ITRITE = Tract/block right of street segment (1 to NPOLY)
Figure 11. Sample Block Boundary File Layout
XL, YL = X and Y coordinates of the beginning node for a segment.

XR, YR = X and Y coordinate of the ending node for a segment.

<table>
<thead>
<tr>
<th>COL</th>
<th>9</th>
<th>18</th>
<th>26</th>
<th>32</th>
<th>38</th>
<th>44</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>137411020</td>
<td>137411100</td>
<td>+03773</td>
<td>+04503</td>
<td>+03523</td>
<td>+04492</td>
</tr>
<tr>
<td>ITLEFT</td>
<td>ITRITE</td>
<td>XL</td>
<td>YL</td>
<td>XR</td>
<td>YR</td>
<td></td>
</tr>
</tbody>
</table>

6-9
7.0 DIME AREA CENTROID SYSTEM (DACS)

7.1 Introduction

The DIME Area Centroid System (DACS) was modified and adapted to the LUMIS project as an alternative to the CHAIN program to create a census tract/block polygon input to PIOS (Polygon Intersect Overlay System) from the DIME file. The DACS program translates a segment record structure used in the DIME file to a single sided polygon outline file which is used by PIOS.

DACS was originally developed by the Census Use Study, a Division of the Bureau of the Census. It was designed to produce edited and reformatted subfiles from the original DIME format. The subfiles can be used in computer mapping, area calculations, and adjacency problems. The editing feature allows the user to correct the file at the same time.

This documentation describes the basic program logic, and data input and output files in the program. This version includes corrections and modifications made by the Jet Propulsion Laboratory to create a PIOS input file. The program and modifications are written in FORTRAN IV for use with an IBM 360/370 computer, but can be adapted to other systems.

7.2 Methodology

DACS is written in three phases. These phases deal with the creating of a single sided segment intermediate subfile, a sort, and editing and reformatting an output file.

An intermediate subfile is created from all boundary segments for a selected region: census tract, census block group, or block which is identified by having two different right and left regions. The data for each boundary is written to the subfile twice, once unaltered as the boundary to the first polygon, and then with the beginning and ending node reversed to create a boundary for the second polygon. The final file contains separate records for each left boundary using counterclockwise orientation, and true records for the right boundary.
In Figure 12 segment A to B has a left side with census tract 1011.00 block 101 and a right tract of 1013.00 block 101. The first segment is written to the subfile as tract 1011.00 block 101 and coordinate for nodes A to B. The second segment is for tract 1013.00 coordinates from B to A, putting that tract on the left side with relation to this X-Y coordinate.

In the sort step, the subfile is sorted from the leftmost region in preparation for the third step.

In the third step, the program edits the file, performs any necessary calculations and reformats the output. All the boundary lines for one region are read in and stored. The array is then passed to the chaining edit program to determine if the segments are for one or more closed loops. If the segments do not close, the user is notified of the condition by a printed message and all calculations are skipped, (see Figure 13A). If the segments form more than one closed loop (Figure 13B), a message indicating the condition is printed, but in this case calculations are not allowed to proceed.

Regions accepted by the chaining test are passed to routines to calculate the centroid coordinates and the area, build lists of adjacent regions, or reformat the segments for use in mapping and overlay programs. The complete documentation for area, adjacency and centroid routines is available through the Census Bureau.

The subroutine to reformat the DIME file for PIOS input is appended to the third step of DACS. The subroutine first modifies the tract and block identifiers by removing the case to form the description record. Minimum and maximum coordinates are determined through a simple logical test, at the same time counting the number of coordinate pairs to form the second polygon record, minimum and maximum coordinates. Finally the X, Y coordinates which describe the polygon are written to a subfile in sets of four as specified for PIOS input.

7.3 Program

The DIME Area Centroid System (DACS) consists of two programs designed to calculate areas and locate centroids of user-specified blocks, tracts, block groups, or other local areas defined in a DIME File. For each region
Map showing boundary segment A-B dividing census tract 1011.00 and 1013.00. Each segment in the DIME file is identified by both the left and the right geocodes making each polygon independent. DACS can then convert DIME files into complete polygon boundary description for PIOS input.

Figure 12. Boundary Segment Dividing Census Tract

Examples of two possible polygons edited by the DACS program.
A. A block which will not chain.
B. A block which chains, but with more than one closed polygon.

Figure 13. DACS Edited Polygons
considered, DACS produces, at user request, a tape listing X and Y coordinates of the boundary segment endpoints forming the polygon vertices and/or type and printed listings of adjacent regions. The centroids and boundary coordinates are useful as input to computer mapping programs such as GRIDS and SYMAP, and as input to polygon overlay programs, such as PIOS.

Errors and anomalies in the area and centroid values, identified by the program, are helpful in locating coding errors in the input file.

DACS is written as a preprocessor, sort and calculation program. These are used in three distinct phases.

**Phase I:**
The first program, DACS presort, uses an input DIME file tape to prepare an intermediate file; the intermediate file passes the control parameters and the selected boundary segments will be used in the calculation program; thus two programs can be thought of as phases of the same program.

**Phase II:**
The second phase sorts the output from the preprocessor so that all the segments from a region will be in one location on the file.

**Phase III:**
The third phase performs the actual computations, edits the file and writes any output files. It reads the sorted intermediate tape and stores segment data for a region internally. When the first record for a different region is encountered, the accumulated data is passed to subroutine CHAIN.

Subroutine CHAIN checks to see if the region's boundary segments form one or more closed loops. If they do not, a message to notify the user of the condition is printed, and calculations are skipped. If they form more than one closed loop, another message is printed; however, in this case, calculation is allowed to proceed.

Data for a valid region is used in subroutine CALC, which computes the area and centroid coordinates. Subroutine POLYPT then determines whether or not the centroid lies within the region boundary. If it lies outside
the boundary, an appropriate message is printed, and the centroid X and Y coordinates are set equal to those of the nearest boundary segment node. Subroutine REFMT reformats the data into a file which can be used by PIOS.

There are a maximum of four output tapes created by DACS. The area and centroid data, with identifying region number, are written onto the first tape, a mapping file onto a second tape, an adjacency file is written onto a third, and PIOS formatted file is written to a fourth tape.

7.3.1 Input. The preprocessor program uses a standard 300 character formatted DIME file or a binary file depending on the user specification and the control parameters. The control parameters are read as cards and passed to the second program as the first input record.

7.3.2 Output. Four output files can be produced by DACS. A file of polygon areas and centroids is created in each run as a standard output. The remaining files are optional, and a mapping file, an adjacency file and a PIOS formatted file.

The mapping file consists of endpoints or vertices of each region's boundary segments. The record to describe a polygon is written in succession and in a format usable by SYMAP.

The adjacency file lists all regions bordering the region analyzed; one adjacent region is listed per record, and all records pertaining to a region occur in succession.

The PIOS compatible file is an 80 character file similar to the mapping file, but is formatted differently. Each polygon has a description record, minimum and maximum X, Y coordinates and the vertices written as four per record.

DACS will also list the polygon's area, and its centroid coordinates — or an adjacency list at the user's request.

The user is continuously notified of error conditions. Possible error conditions include improperly defined regions and anomalies in centroid location, as well as errors in the coding of the original DIME file.
Possible file error conditions can occur because:

a) The polygon has more than 2000 segments.

b) The segments form more than one polygon.

c) The segments do not close.

d) A segment on the file must be reversed before proceeding with processing.

e) The centroid of the polygon falls outside the polygon and must be adjusted.

The polygon only stops processing in case a) or c); in all other cases a warning is printed and processing continues.

7.4 Phase I Specifications:

DACS preprocessor, Phase I, has two input types: a card deck and the tape file. The card deck passes control specifications to the program to determine the input and output selections, and the area of study. The first card, the problem card, selects the user options. This is followed by a series of cards which selects the required census tracts, blocks, etc. contained in the study area.

If SELECT cards are included, the program performs calculations only for the regions identified by the cards; if REJECT cards are included, it considers all regions except those identified. If neither SELECT nor REJECT cards are used, the program calculates areas and centroids for all regions in the file.

Input: Profile Cards
File — Type of input file
1-6 character or binary

KALL — Type of region to be considered

11-14 TRCS for tracks
BLKS for blocks
RGNS for regions identified by area codes
BGPS for block groups; if left blank, the program assumes tracts are desired.
CODE — Type cards used to define the study area,
19-24 selection or rejection
SELECT if selection cards follow
REJECT if rejection cards follow
Blank if neither selection nor rejection cards are included.

ELECT — Type of region specified on selection or rejection cards to follow.
27-30 TRCS for tracts
BLKS for blocks
RGNS for regions identified by area codes
BGRP for block groups
Blank if no cards are included.

AREA — Units in which area is to be calculated
46-49 SQFT for square feet
SQMI for square miles
ACRE for acres
If left blank, the program will calculate areas in square feet

OPTION — Optional output file specified.
59 1 if printout of areas and centroids desired
0 if printout not desired
60 1 if tape with SYMAP boundary coordinates desired
0 if tape not desired
61 1 if "adjacency list" is to be produced (tape and printer)
0 if "adjacency list" not desired.
62 1 if tape with PIOS boundary coordinate desired
0 if tape not desired.

TITLE — Title that will appear at the top of each page of printed output.
69-80 BINARY TRCS SELECT TRCS SQFT 0100 CENSUS---
SELECTION/REJECTION CARDS. ONE CARD

For each tract, block, or region to be selected or rejected.

If regions specified are tracts:

1-3   Blank
4-9   Tract number

If regions specified are blocks:

1-6   Tract number
7-9   Block number

If regions specified are identified by area code:

1-6   Blank
7-9   Area Code

If regions specified are block groups

1-2   Blank
3-8   Tract number
9     Block group number

Restrictions and Assumptions

Number of selection/rejection cards = 2000.
Number of boundary segments for any region specified = 2000

Input: tape

A standard Census Bureau tape is the second input to DACS preprocessor. The program will accept either a standard 300 byte character file or a 324 byte binary file as input.

Output: tape

The output from DACS preprocessor is an intermediate tape file with information about output options and a description of all boundary segments selected from the original ACG-DIME File.
The first record of the tape is the control record.

Format is as follows:

<table>
<thead>
<tr>
<th>Columns</th>
<th>Format</th>
<th>Data Item</th>
</tr>
</thead>
<tbody>
<tr>
<td>19-29</td>
<td>F11.0</td>
<td>Multiplier, to convert calculated area to proper units.</td>
</tr>
<tr>
<td>30-33</td>
<td>A4</td>
<td>Units (SQFT, SQMI, or ACRE)</td>
</tr>
<tr>
<td>34-37</td>
<td>A4</td>
<td>Region type (TRCS, BLKS, BGRP, or RGNS)</td>
</tr>
<tr>
<td>38-41</td>
<td>411</td>
<td>Code to specify selection of output options</td>
</tr>
<tr>
<td>42-53</td>
<td>3A4</td>
<td>Title, to appear on output page</td>
</tr>
</tbody>
</table>

The remainder of the records describe the boundary segments.

Boundary segments are those having different "right" and "left" regions. Data for each such segment in the original file is written twice on the intermediate tape—once with its original orientation, once with orientation reversed (i.e., with "left" and "right," "from" and "to" designations switched). Format for each segment record is as follows:

<table>
<thead>
<tr>
<th>Columns</th>
<th>Format</th>
<th>Data Item</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-9</td>
<td>I9</td>
<td>&quot;Left&quot; region</td>
</tr>
<tr>
<td>10-18</td>
<td>I9</td>
<td>&quot;Right&quot; region</td>
</tr>
<tr>
<td>19-27</td>
<td>I9</td>
<td>Segment serial number</td>
</tr>
<tr>
<td>28-37</td>
<td>I9</td>
<td>&quot;From&quot; node</td>
</tr>
<tr>
<td>38-47</td>
<td>I9</td>
<td>&quot;To&quot; node</td>
</tr>
<tr>
<td>48-57</td>
<td>I9</td>
<td>&quot;From&quot; node x coordinate</td>
</tr>
<tr>
<td>58-67</td>
<td>I9</td>
<td>&quot;From&quot; node y coordinate</td>
</tr>
<tr>
<td>67 bytes</td>
<td></td>
<td>Record length</td>
</tr>
</tbody>
</table>

7.5 Phase II Specifications:

Before it is used as input to the DACS calculation program, the tape must be sorted by "left" region (Columns 1-9).
7.6 Phase III Specifications:

The calculation program requires very little user involvement with the exception of JCL requirements.

Input: The calculation program accepts the output file from the sort step directly without modification.

Output: The program has one output tape that is standard with three additional optional tapes.

Area-centroid tape is a file with the polygon area and its centroid produced on each run. The record format is as follows:

<table>
<thead>
<tr>
<th>Columns</th>
<th>Format</th>
<th>Data Item</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-10</td>
<td>I10</td>
<td>The key used in the selection of block, track, block group, block or right justified block group is preceded by tract number or local area.</td>
</tr>
<tr>
<td>11-30</td>
<td>F20,5</td>
<td>Area (in user-specified units)</td>
</tr>
<tr>
<td>31-40</td>
<td>I10</td>
<td>Centroid &quot;x&quot; coordinate (map miles)</td>
</tr>
<tr>
<td>41-50</td>
<td>I10</td>
<td>Centroid &quot;y&quot; coordinate (map miles)</td>
</tr>
</tbody>
</table>

Record length = 50 bytes.

The "computer mapping" option produces a tape with coordinates of end point modes of each region's boundary segments, all pertaining to a particular region occurring in succession. Output is written to File 11.

Record format as follows:

<table>
<thead>
<tr>
<th>Columns</th>
<th>Format</th>
<th>Data Item</th>
</tr>
</thead>
<tbody>
<tr>
<td>11-20</td>
<td>F10,0</td>
<td>&quot;From&quot; note y coordinate.</td>
</tr>
<tr>
<td>21-30</td>
<td>F10,0</td>
<td>&quot;From&quot; node x coordinate.</td>
</tr>
<tr>
<td>70-76</td>
<td>I7</td>
<td>Number of left-hand tract, block, block group, or local area.</td>
</tr>
<tr>
<td>77-80</td>
<td>I3</td>
<td>Sequence number of segment in the list of boundary segments for the region.</td>
</tr>
</tbody>
</table>

Record length = 80 bytes.
The "adjacency list" tape option produces a tape containing a list of all regions (blocks, tracts, block groups, or special local areas) bordering on each region considered. One adjacent region is listed per record, and all records pertaining to a particular region occur in succeeding records. Output is written to File 12.

Record format is as follows:

<table>
<thead>
<tr>
<th>Columns</th>
<th>Format</th>
<th>Data Item</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-15</td>
<td>I15</td>
<td>Region number</td>
</tr>
<tr>
<td>15-30</td>
<td>I15</td>
<td>Adjacent region number</td>
</tr>
</tbody>
</table>

Record length = 30 bytes.

The "PIOS data file" tape has three record types; description record, minimum maximum coordinates, and polygon boundary record. Output is written to File 13. Record format is as follows:

Record Type 1

<table>
<thead>
<tr>
<th>Columns</th>
<th>Format</th>
<th>Data Item</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-5</td>
<td>I5</td>
<td>Census tract ID deleting first digit of suffix.</td>
</tr>
<tr>
<td>6-10</td>
<td>I5</td>
<td>Block ID deleting first digit of suffix.</td>
</tr>
<tr>
<td>11-13</td>
<td>A3</td>
<td>Block ID — first three numbers</td>
</tr>
<tr>
<td>14-17</td>
<td>I4</td>
<td>Sequence number.</td>
</tr>
<tr>
<td>18-19</td>
<td>I2</td>
<td>Polygon type.</td>
</tr>
</tbody>
</table>

Record Type 2

<table>
<thead>
<tr>
<th>Columns</th>
<th>Format</th>
<th>Data Item</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-9</td>
<td>I9</td>
<td>Minimum x coordinate</td>
</tr>
<tr>
<td>10-18</td>
<td>I9</td>
<td>Minimum y coordinate</td>
</tr>
<tr>
<td>19-27</td>
<td>I9</td>
<td>Maximum x coordinate</td>
</tr>
<tr>
<td>28-36</td>
<td>I9</td>
<td>Maximum y coordinate</td>
</tr>
<tr>
<td>37-41</td>
<td>I5</td>
<td>Number of coordinate pair describing polygon.</td>
</tr>
</tbody>
</table>
### Record Types

<table>
<thead>
<tr>
<th>Columns</th>
<th>Format</th>
<th>Data Item</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-9</td>
<td>I9</td>
<td>X Value</td>
</tr>
<tr>
<td>10-18</td>
<td>I9</td>
<td>Y Value</td>
</tr>
<tr>
<td>19-27</td>
<td>I9</td>
<td>X Value</td>
</tr>
<tr>
<td>28-36</td>
<td>I9</td>
<td>Y Value</td>
</tr>
<tr>
<td>37-45</td>
<td>I9</td>
<td>X Value</td>
</tr>
<tr>
<td>46-54</td>
<td>I9</td>
<td>Y Value</td>
</tr>
<tr>
<td>55-63</td>
<td>I9</td>
<td>X Value</td>
</tr>
<tr>
<td>64-72</td>
<td>I9</td>
<td>Y Value</td>
</tr>
</tbody>
</table>

### Printed Output

For the properly defined regions considered, the user may request a printout of area and centroid coordinates and/or a polygon adjacency list.
8.0 POLYGON INTERSECTION OVERLAY SYSTEM (PIOS)

8.1 Introduction:

The polygon overlay program is part of the PIOS (Polygon Intersection Overlay System); it uses output data from the ROTATE and CHAIN programs to compute statistics for major and minor polygon intersections. PIOS was originally developed for the San Diego Comprehensive Planning Organization (CPO by Environmental Systems Research Institute (ESRI) of Redlands, California). The system was designed and implemented in 1971 under a contract from CPO to ESRI to digitize soils polygons for San Diego County and then quantify soils type within Traffic Analysis Zones (TAZ), which are nominal statistical areas used for planning purposes by CPO.

This documentation describes the program logic and the data files of the Polygon Information Overlay System program Version 75, (PIOS-75). This version includes the modifications which the Jet Propulsion Laboratory has made to PIOS II, in order to accommodate a specific land use polygon overlay problem. This modified version retains PIOS II, which was created for the CPO in San Diego. PIOS-75 will run on either the Univac 1108, or the IBM 360/370 computers.

8.2 Methodology:

PIOS numerically overlays any general set of ordinarily defined polygons. These are identified as a major and a minor polygon, and their overlay produces a residual polygon of their intersections. This residual polygon can be overlaid with a third polygon, and so forth, each overlay producing a residual polygon of common intersection. This overlay system is conceptually similar to a "cookie cutter."

The census blocks are the major polygons in the overlay technique employed in LUMIS. In the Los Angeles test eight digitized ordinal maps became the minor polygons. Major polygons, after processing with the polygon overlay software, contained the information overlaid from the minor polygon. These minor polygons while proposed were never created from land use and natural resource maps, slope maps, elevation maps, geology maps, landslide and fault maps, soil maps, and air pollution indices.
PIOS reads all minor (land use) polygons and stores their approximate location, i.e., the minimum and maximum boundaries or window. The window technique allows complex polygons to be treated as boxes which are sorted and manipulated in their general format until a more accurate definition is required. This technique permits more efficient computational analysis of a complex polygon file.

Having windowed the minor polygon file and stored it in a temporary direct access file, the program reads the major polygon file (census blocks). Before any analysis of overlapping polygon takes place the program breaks up the major polygon into strips (see Figure 14). New points are generated at strip boundaries so that each strip is wholly contained.

The logic of this stripping technique is not unique, but it does provide for substantial increase in program efficiency. Briefly the strips reduce the amount of core storage required by the program by only retaining that portion of the polygon that is being analyzed for intersection at one time. The number of strips is determined by the tradeoff between efficiency gained in the processing, and the core storage requirements necessary to run the program. As the complexity of polygons increases, the efficiency gained by increasing the number of strips also increases. The point-in-polygon technique uses directed line segments in the positive X-direction of slope zero; line segments so directed will not cross over onto another strip. For the LUMIS test the major polygons were broken down into 16 strips of equal height.

After the major polygon (census block) has been stripped a minor polygon (land use) is read in to compute the intersection points, if any. Each point in the minor polygon is tested for being in or out of the major polygon.

If the minor polygon is determined to be inside the major polygon, there are three possible options.

Case 1 — The land use polygon is completely and totally contained within the census tract polygon.

Case 2 — The census tract polygon is completely contained within the land use polygon.

Case 3 — Part of the land use polygon resides within the census tract polygon, and a residual polygon is formed.
Figure 14. PIOS Stripping Method
In Case 1, PIOS-75 writes the entire land use polygon to the output file. In Case 2, the entire census tract polygon is written to the output file. In Case 3, PIOS-75 effectively cuts off and saves the part of the land use polygon which resides within the census tract. By creating two residual polygons from the original minor polygon, Figure 15 geographically displays Cases 1-3.

In Case 3, each segment of a minor polygon must be analyzed individually for being either completely inside, completely outside, or intersecting the major polygon. If the segment intersects the major polygon, it is entering or leaving. This is graphically pictured in Figure 16.

Four possible actions taken are tabulated below:

<table>
<thead>
<tr>
<th>Example</th>
<th>Action Taken</th>
</tr>
</thead>
<tbody>
<tr>
<td>A to B</td>
<td>Throw away point A. Do not consider segment as part of polygon.</td>
</tr>
<tr>
<td>B to C</td>
<td>Find intersection B'. Determine direction on major polygon. Follow major polygon until it goes out of minor polygon F'.</td>
</tr>
<tr>
<td>C to D</td>
<td>Add points to the line segment defining the new polygon being calculated.</td>
</tr>
<tr>
<td>D to E</td>
<td></td>
</tr>
<tr>
<td>E to F</td>
<td></td>
</tr>
<tr>
<td>F to G</td>
<td>Find intersection F'. Determine direction on major polygon. Follow major polygon until it goes out of minor polygon B'.</td>
</tr>
</tbody>
</table>

After selecting one of these actions, a new polygon is developed and subsequently computed for area.

PIOS-75 will examine each land use polygon in sequential order while looking for a match. If after checking all land use polygons no match is found, the main program will request the next census tract to be read in for the same review. If it finds no match between any land use polygon and census tract polygon, the program will terminate.
Figure 15. Example of Alternative Block and Land Use Intersections

Figure 16. An example of identifying unique polygons created by intersecting blocks and land use
As part of the JPL improvements to PIOS, the program lists census tracts and blocks, the residual land use area, and percentage each makes of the entire tract per each run. The listing also includes those tracts for which no overlay was possible as well as those for which the entire census tract fell into one minor polygon. Further abstractions or summaries can be derived from the output listing to meet the users needs.

The summary can be used as a preliminary means of checking for errors in the PIOS run. A better means for error checking is to sum all the residual areas and check the total against the original polygon as illustrated in Figure 17. A second supplemental program is provided for error checking. The second program uses as input the original minor polygon file (12) and the residual polygon file (14).

The residual file must first be sorted by land use map and code, polygon number, census tract and block. The program will aggregate and list the residual polygon coded for a land use map and type and number. If the entire land use was overlaid, the listings should show the polygon as 100% overlayed by blocks. In case of partial overlay, the results can be used to estimate the amount of coverage.

8.3 Program:

PIOS is used to determine the common area of two overlapping polygons. The program will overlay one or more "major" polygons onto the "minor" polygon file. Each major polygon is compared with each minor polygon for possible overlap. Both the major and minor polygons have a predefined "window" associated with them in the form of minimum and maximum X and Y coordinates. If the major and minor "windows" overlap, then the program uses a specialized version of the relatively common point-in-polygon technique to determine if any overlap actually exists. If two polygons do overlap, a polygon representing the overlap area is structured using the existing coordinates of the major and minor polygons. Once the structuring process is complete, the area of the new polygon is calculated and sorted in a "residual" polygon file for later use in tabular or graphic report generating programs. This process is illustrated in Figure 18.
Figure 17. Reconstruction of Minor Polygons from Multiple Intersection Polygons ($A_1 + A_2 + A_3 + A_4$)
Figure 18. Flow Chart PIOS Program
The program is written as a series of FORTRAN IV subroutines invoked by the main program DRIVER.

Subroutine POLYRD (polygon read) is used to read each of the polygons in from the external file storage medium. The subroutine makes use of random-access techniques for reading both the major and minor polygon files. It also performs the function of calculating centroids from the coordinate arrays of simple and containment type polygons.

The function LEVEL is used to provide level indicies for the polygon overlay procedure. Major polygons are "stripped" into 16 horizontal sections or levels. When the points in the minor polygon boundary are submitted for the point-in-polygon examination, the levels of the major polygon could potentially contain the minor point. Then, only the line segments which make up that level are used by the PIP subroutine.

Subroutine PIP (point-in-polygon) determines whether a point on the minor polygon boundary is located inside or outside of the major polygon boundary. Establishing this relationship for each of the minor polygon vertices is the foundation of the overlay process.

Subroutine STPSUB (step subroutine) is used in conjunction with the residual structuring process. In cases where the direction of digitizing is not known and an intersection of major and minor boundaries is encountered, this subroutine provides an indication of the direction to "step" next.

Subroutine AREAOF is called by subroutine POLYRD to determine the area of a polygon, using the cross product. It also determines if the polygon has been digitized in the correct clockwise direction by the sign of the resulting area. If the polygon was digitized in the wrong counterclockwise direction, the polygon is reversed.

8.4 Input Files:

The input files to the polygon overlay program are both in the same format. The files are differentiated by the program in that one is considered as the major file, the other as the minor file. The choice of which is the major file is the users'. The major polygons are usually the larger polygons, and the minor polygons would normally be those that are wholly or partially
contained within the major polygon. The major polygon file is identified as File 11 and the minor polygon file is File 12. Polygons from File 12 are matched up with, or overlayed upon, data from File 11. In the LUMIS project the census block was the major polygon file and land use was the minor polygon file. Figure 19 illustrates the deck setup for PIOS.

PIOS will process both simple and complex polygons. "Doughnut holes" within complex polygons will be written to the output file with negative acreages. This facilitates later summation processes and tabular reporting requirements. The only limitation on input files is that if a minor polygon file contains more than 1,000 polygons, only the first 1,000 will be used in the overlay process. Larger files can be processed by enlarging the internal storage arrays in the polygon overlay program, which increases the core storage requirements.

8.5 Output Files:

The three files created by the PIOS are similar in content and somewhat redundant: The residual file, File 14, is a polygon file that contains the areas that were found to be common between the major and minor polygons. Each polygon contains the tract and block number of the major polygon and the land use code, polygon number and type code of the minor polygon, the calculated area of the polygon (in acres) and the coordinates of the polygon vertices.

The second output file, File 13, is a condensed version of the residual file. It contains the same information for each polygon as the residual file but the boundary coordinates are excluded. This file is more efficiently used in tabular summation programs and reporting applications than the residual file.

The third output file, File 15, is a condensed version of the census tract file. It contains the census tract and block identification and total area of census blocks.

8.5.1 File 11 — Major polygon file. It is a direct access file with census tract and block records. The file contains records 80 characters in length. Each census block is made up of three record types: a descriptor record, a MIN/MAX record, and the boundary points.
Figure 19. Major or Minor Polygon File Required by PIOS
The major polygon descriptor record contains basic identification information to describe the minimum and maximum points and the boundary record to follow. The following data names are used by the program:

MAPNO: The census tract identification number is the census tract number with the first number of the suffix deleted. Since this number is always zero, it can be deleted to save space in later processing.

MAPOL 1 (2): The block number identification number is the census block number with the first number of the suffix deleted. As in the case of the census tract, this number can be deleted without losing information.

LUCODE (2): The block number is repeated in this space dropping the entire suffix. This number serves as a filler in processing the census tract file, but is important in processing the minor file.

LUCODE (1): The map sequence number denotes the sequence in relation to all the census tracts coded in a map.

LUCODE (3): The polygon-type code describes the character of each polygon. A normal polygon is coded 10. Doughnut polygons, (polygons within polygons) are coded by numbers larger than 10, starting with 11. Two mutually exclusive doughnut holes are coded 11, while nested polygons follow in order: 11, 12, ... All digitized doughnut holes must follow the parent.

COL 5 10 13 17 19

8.5.2 File 12 — Minor polygon file. It is a direct access with the land use or ordinal polygons. The file contains records 80 characters in length. Each land use polygon is described by three record types similar to the census tract input with minor modifications and different data names.
The record 1 minor polygon descriptor record contains basic identification information for the minimum and maximum points and boundary records to follow.

The following data names are used in the Program:

SPMAP: The identification number of the ordinal/land use map that is being processed.

SPPOLY (1): The total numbers of polygons coded using a map or identification number.

SPCODE:

SPCODE (1): The identification code identifies the land use type as defined in the land use map legend.

SPCODE (3): The sequence number denotes the sequence in relation to the total number of polygons coded in a map.

SPPOLY (2): The polygon-type code describes the character of each polygon. A normal polygon is coded 10. Doughnut polygons (polygons within polygons) are coded by numbers larger than 10, starting with 11. Two mutually exclusive doughnut holes are coded 11, while nested polygons follow in order: 11, 12, --. All digitized doughnut holes must follow the parent.

<table>
<thead>
<tr>
<th>COL</th>
<th>5</th>
<th>10</th>
<th>13</th>
<th>17</th>
<th>19</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>17401</td>
<td>80</td>
<td>X03</td>
<td>60</td>
<td>10</td>
</tr>
</tbody>
</table>

8.5.3 Record 2 major/minor — min/max record. The Min/Max Record describes the extreme points of the polygon or the Minimum X and Y coordinate and the Maximum X and Y coordinate, and the number of coordinate pairs which will follow to describe the polygon's perimeter.
### Column Format Data Item

<table>
<thead>
<tr>
<th>Column</th>
<th>Format</th>
<th>Data Item</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-9</td>
<td>I9</td>
<td>xmin - The minimum x coordinate value of the polygon</td>
</tr>
<tr>
<td>10-18</td>
<td>I9</td>
<td>ymin - The minimum y coordinate value of the polygon</td>
</tr>
<tr>
<td>19-27</td>
<td>I9</td>
<td>xmax - The maximum x coordinate value of the polygon</td>
</tr>
<tr>
<td>28-36</td>
<td>I9</td>
<td>ymax - The maximum y coordinate value of the polygon</td>
</tr>
<tr>
<td>37-41</td>
<td>I5</td>
<td>n - The number of coordinate pairs describing the polygon</td>
</tr>
</tbody>
</table>

#### 8.5.4 Record 3 to 3+N/4 pairs major/minor record

Polygon boundary record (S) - This set of records describes the polygon boundaries in terms of x and y coordinate pairs starting with x, y to x, y

<table>
<thead>
<tr>
<th>Column</th>
<th>Format</th>
<th>Data Item</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-9</td>
<td>I9</td>
<td>x value</td>
</tr>
<tr>
<td>10-18</td>
<td>I9</td>
<td>y value</td>
</tr>
<tr>
<td>19-27</td>
<td>I9</td>
<td>x value</td>
</tr>
<tr>
<td>28-36</td>
<td>I9</td>
<td>y value</td>
</tr>
<tr>
<td>37-45</td>
<td>I9</td>
<td>x value</td>
</tr>
<tr>
<td>46-54</td>
<td>I9</td>
<td>y value</td>
</tr>
<tr>
<td>55-63</td>
<td>I9</td>
<td>x value</td>
</tr>
<tr>
<td>64-72</td>
<td>I9</td>
<td>y value</td>
</tr>
</tbody>
</table>

#### 8.5.5 End of File records

A sequence of four 9's (9999) is recognized by the IBM 360/370 program as an end of file when located in bytes 1-4. A sequence of five 9's (99999) is recognized by the U1108 program as an end of file when located in bytes 1-5. The last two records of each direct access file (File 11, 12, 13 and 14) should contain these codes.

8-14
8.6 Output:

File 13 contains the residual polygon produced by the PIOS Program. Each record is stored on a direct access device as a 37-character record. The record incorporates the residual polygon census tract identification with the land use codes and the polygon area. The record output is in the following format:

<table>
<thead>
<tr>
<th>Column</th>
<th>Format</th>
<th>Data Item</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-5</td>
<td>I5</td>
<td>Census tract numbers of major polygon involved in the overlay.</td>
</tr>
<tr>
<td>6-10</td>
<td>I5</td>
<td>Block number of major polygon involved in the overlay.</td>
</tr>
<tr>
<td>11-15</td>
<td>I5</td>
<td>Land use map or minor polygon in overlay.</td>
</tr>
<tr>
<td>16-18</td>
<td>A3</td>
<td>Land use identification information code.</td>
</tr>
<tr>
<td>19-22</td>
<td>I4</td>
<td>Polygon sequence number or land use involved in the overlay.</td>
</tr>
<tr>
<td>23-37</td>
<td>F15,2</td>
<td>Area of the residual polygon file.</td>
</tr>
</tbody>
</table>

File 14 contains essentially the same descriptive information as File 13, the census tract and land use code in an 80-column format. It adds to this information a series of x and y coordinates that describe the residual polygon that is formed in the overlay process.
### 8.6.1 Record 1

<table>
<thead>
<tr>
<th>Column</th>
<th>Format</th>
<th>Data Item</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-5</td>
<td>I5</td>
<td>Census Tract</td>
</tr>
<tr>
<td>6-10</td>
<td>I5</td>
<td>Census Block number</td>
</tr>
<tr>
<td>11-15</td>
<td>I5</td>
<td>Land use map</td>
</tr>
<tr>
<td>16-18</td>
<td>A3</td>
<td>Land use map identification code</td>
</tr>
<tr>
<td>19-22</td>
<td>I4</td>
<td>Sequence number of the land use polygon</td>
</tr>
<tr>
<td>23-37</td>
<td>F15.2</td>
<td>Area of the residual polygon</td>
</tr>
<tr>
<td>38-80</td>
<td></td>
<td>Blank</td>
</tr>
</tbody>
</table>

**Polygon Boundary Coordinate Pairs**

### 8.6.2 Record 2 thru 2+N/4

<table>
<thead>
<tr>
<th>Column</th>
<th>Format</th>
<th>Data</th>
<th>Item</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-9</td>
<td>F9.1</td>
<td>X₁</td>
<td>Value</td>
</tr>
<tr>
<td>10-18</td>
<td>F9.1</td>
<td>Y₁</td>
<td>Value</td>
</tr>
<tr>
<td>19-27</td>
<td>F9.1</td>
<td>X₂</td>
<td>Value</td>
</tr>
<tr>
<td>28-36</td>
<td>F9.1</td>
<td>Y₂</td>
<td>Value</td>
</tr>
<tr>
<td>37-45</td>
<td>F9.1</td>
<td>X₃</td>
<td>Value</td>
</tr>
<tr>
<td>46-54</td>
<td>F9.1</td>
<td>Y₃</td>
<td>Value</td>
</tr>
<tr>
<td>55-63</td>
<td>F9.1</td>
<td>X₄</td>
<td>Value</td>
</tr>
<tr>
<td>64-72</td>
<td>F9.1</td>
<td>Y₄</td>
<td>Value</td>
</tr>
</tbody>
</table>

File 15 is the second abbreviated file produced by PIOS. This file contains the same identification information found on the census tract descriptor record: tract, block, and sequence number.
Census tract area calculated in the PIOS Program is added forming the final data item on the record. The record is 29 characters in length and written on a direct access device in the following format:

<table>
<thead>
<tr>
<th>Column</th>
<th>Format</th>
<th>Data Item</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-5</td>
<td>I5</td>
<td>Census tract number of major polygon</td>
</tr>
<tr>
<td>6-10</td>
<td>I5</td>
<td>Block number of major polygon</td>
</tr>
<tr>
<td>11-14</td>
<td>I4</td>
<td>Sequence number of block with census tract</td>
</tr>
<tr>
<td>15-29</td>
<td>F15.2</td>
<td>Area of the block</td>
</tr>
</tbody>
</table>
STATE PLANE COORDINATE CALCULATION PROGRAM (SPCCAL)

INPUT NUMBER OF MAPS FOR WHICH THE CONTROL POINT SPC S ARE TO BE CALCULATED.

1 READ (5,5) NUMAP
   DO B J=1,NUMAP
   WRITE (6,70)
   WRITE (6,50)

INPUT THE MAP NUMBER (MAPNO) AND TOTAL NUMBER OF CONTROL POINTS (NCONPT)

READ (5,10) MAPNO, NCONPT
WRITE (6,15) MAPNO
WRITE (6,50)
DO 20 I=1, NCONPT

THE X1, X2, X3, X4, Y1, Y2, Y3, Y4 SPC COORDINATES OF THE FOUR 2 1/2 MINUTE LATITUDE-LONGITUDE INTERSECTIONS ARE READ IN.

READ (5,30) A,B,C,D,E,F,G,H
WRITE (6,32) I
WRITE (6,50)

THE (SMALL LETTER X,Y) X1, X2, Y1, Y2 DIGITIZER MEASURED DISTANCES ARE READ

Q=X1
P=X2
Q=Y1
R=Y2

READ (5,35) O,P,Q,R
O = O*200U.
P = P*200U.
Q = Q*200U.
R = R*200U.
\[ X_1 = A + O + \left(\frac{O}{Q+R}\right) \cdot (C-A) \]
\[ X_2 = C + O - \left(\frac{R}{Q+R}\right) \cdot (C-A) \]
\[ X_3 = B + P + \left(\frac{Q}{Q+R}\right) \cdot (D-B) \]
\[ X_4 = O - P - \left(\frac{R}{Q+R}\right) \cdot (D-B) \]
\[ Y_1 = E + Q - \left(\frac{O}{Q+P}\right) \cdot (E-F) \]
\[ Y_2 = F + U + \left(\frac{P}{Q+P}\right) \cdot (E-F) \]
\[ Y_3 = G - R - \left(\frac{O}{O+P}\right) \cdot (G-H) \]
\[ Y_4 = H - R - \left(\frac{P}{O+P}\right) \cdot (G-H) \]
\[ Z_1 = \frac{(X_1 + X_2)}{2} \]
\[ Z_2 = \frac{(X_3 + X_4)}{2} \]
\[ Z_3 = \frac{(Y_1 + Y_2)}{2} \]
\[ Z_4 = \frac{(Y_3 + Y_4)}{2} \]
\[ Z_5 = |Z_5| \]
\[ Z_6 = |Z_6| \]

IF (Z_5 > 30.0) GO TO 60
IF (Z_6 > 30.0) GO TO 60

\[ T_1 = \frac{(X_1+X_2+X_3+X_4)}{4} \]
\[ T_2 = \frac{(Y_1+Y_2+Y_3+Y_4)}{4} \]

WRITE (6,40) X1, Y1
WRITE (6,40) X2, Y2
WRITE (6,40) X3, Y3
WRITE (6,40) X4, Y4
WRITE (6,40) Z5, Z6
WRITE (6,50)
WRITE (6,40) T1, T2

IF (1 * EQ. NCONPT) GO TO 43

WRITE (6,45)
43 WRITE (6,50)
GO TO 2

WRITE (6,80)
WRITE (6,50)
WRITE (6,40) X1, Y1
WRITE (6,40) X2, Y2
WRITE (6,40) X3, Y3
WRITE (6,40) X4, Y4
WRITE (6,50)
WRITE (6,90) Z5, Z6

IF (1 * EQ. NCONPT) GO TO 44
WRITE (6,45)
44 WRITE (6,50)

CONTINUE

STOP

END
<p>| | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>17002</td>
<td>5</td>
<td>4111065.3</td>
<td>4123669.9</td>
<td>4111102.5</td>
<td>4123700.8</td>
<td>4173149.8</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>4111065.3</td>
<td>4123669.9</td>
<td>4111102.5</td>
<td>4123700.8</td>
<td>4173149.8</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>4123638.9</td>
<td>4136249.6</td>
<td>4123669.9</td>
<td>4136274.4</td>
<td>4157958.5</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>4136249.6</td>
<td>4148860.4</td>
<td>4136274.4</td>
<td>4148878.9</td>
<td>4157935.3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>4136249.6</td>
<td>4148860.4</td>
<td>4136274.4</td>
<td>4148878.9</td>
<td>4157935.3</td>
</tr>
<tr>
<td></td>
<td>17003</td>
<td>5</td>
<td>4148860.4</td>
<td>4136274.4</td>
<td>4148878.9</td>
<td>4157935.3</td>
<td>4157917.3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>4148860.4</td>
<td>4161471.1</td>
<td>4148878.9</td>
<td>4161483.5</td>
<td>4157917.3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>4148860.4</td>
<td>4161471.1</td>
<td>4148878.9</td>
<td>4161483.5</td>
<td>4157917.3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>4148860.4</td>
<td>4161471.1</td>
<td>4148878.9</td>
<td>4161483.5</td>
<td>4157917.3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>4148860.4</td>
<td>4161471.1</td>
<td>4148878.9</td>
<td>4161483.5</td>
<td>4157917.3</td>
</tr>
</tbody>
</table>
APPENDIX II

PLOTTER EDITING PROGRAM LISTING

General Map Case
INTEG TYPE, POLYN
DIMENSION XO(9000), YO(9000), STATEP(16), FPM(2)
READ(5, *00) JDF
IF (JDF .EQ. 999) GO TO 21
C
500 FORMAT(14)
200 FORMAT(1H, 10X, 14, 15)
1000 FORMAT(215, A, '9, 14, 12)
1021 FORMAT(12F6, 15)
1047 FORMAT(12F5, 15)
1054 FORMAT(12F5, 15)
1058 FORMAT(2F5, 15)
1060 FORMAT(2F5, 15)
1080 FORMAT(10X, 11, 15)
2000 FORMAT(1H0, 10X, 'MAPNO MAPOL _UCODE POLYN POLYN PTYPE'/
1H, 9X, 16, 15, 5X, 8X, 14, 4X, 14, 5X, 12/1H, 9X, 15(1-1) )
2001 FORMAT(1H, 1X, 'CONTROL POINTS?', ?(2F7, 3, 2X) )
2002 FORMAT(1H, 10X, 'XMIN YMIN XMAX YMAX NP=', 14, 4, 12, 15, 5X, 7, 15)
2003 FORMAT(1H, 2X, 'STRUCTED BY CARDS/LINE')
2004 FORMAT(1H, 2X, 'CONTROL POINTS?')
2005 FORMAT(1H, 10X, 'XMIN YMIN XMAX YMAX NP=', 14, 4, 12, 15, 5X, 7, 15)
2006 FORMAT(1H0, 60(1*) )
C
C
CALL PLOTS(0, 0, 20)
CALL PLOT(0.0, -10.0, -3)
CALL PLOT(0.0, 1.50, -1)
IM = 3
IF (IFRAM = 0)
IF (JDF .EQ. 999) GO TO 1
C
2 READ(5, 1006, FND = 900) ITITLE, ISKIP2
IF (JDF .EQ. 777) GO TO 3
CALL SYMBO(0.0, 0.0, 0.21 A, 0.1P5, 0.1P5, ITITLE, 0.0, 5)
CALL NEWFPN(2)
ENCODE(709, IADDR), IM
700 FORMAT(1AT-, 12)
CALL SYMBO(0.950, -0.21 A, 0.1P5, IADDR, 0.0, 5)
IM = IM + 1
IF (IFRAM = 0)
IM = 0
XM = 999, 000
YM = 999, 000
XM = -999, 000
YM = -999, 000
WRITE(6, 2007) ITITLE, ITITLE, ITITLF
REWIND 2
C
READ(5, 100R, FNP = 900, FND = 900) ISTART
IF (ISKIP2 .EQ. 999) READ(5, 1004) (XO(N), N = 1, 14)
IRN = ISTART + 2
READ(5, 1004) (STATEP(I), I = 1, IRN)
DO 12 I = 1, IRN + 2
XM = MIN(XM, STATEP(I))
YM = MIN(YM, STATEP(I))
12 XM = MAX(XXM, STATEP(I))
WRITE(6, 2001) (STATEP(I), I = 1, IRN)
WRITE(6,2003)

1 IF( ERR,FO.999) GO TO 9
READ(4,1000,END=9,ERR=900) MAPNO, MAPOL, LUCODE, POLYN0, PTYPE
GO TO 7
4 READ(0,1000,END=9,ERR=900) MAPNO, MAPOL, LUCODE, POLYN0, PTYPE
7 IF( ERR = 900)
READ(5,1004) XMIN, YMIN, XMAX, YMAX, NP
XM = MIN(XM,XMIN)
YM = MIN(YM,YMIN)
XMAX = MAX(XMAX,XM)
IF( JIF.EQ.999) WRITE(6,600) J, NP
IF( FRAME = IFRAME + 1)

J = 0
DO 13 L = 1, 6000, 1
13 J = L + 5
I = L
READ(5,1004, F=14, END=15) ( XD(N), YD(N), N=1, M)
17 J = J + 6
GO TO 16
14 IF (ERR = 1)
16 N = J - 5
DO 15 I = N, J
15 IF( XM(I+1)) 15, 3
J = I
GO TO 4
15 CONTINUE

4 IF( JIF.EQ.999) GO TO 1
3 IF( JIF.EQ.999) GO TO M
WRITE(6,2002) MAPNO, MAPOL, LUCODE, POLYN0, PTYPE
WRITE(6,2004) NP, J, XMIN, YMIN, XMAX, YMAX
WRITE(6,2004) NC = 0
21 DO 21 N = 1, J, 4
21 M = N + 5
NC = NC + 1
IF( NC.GT.J) GO TO 9
WRITE(6,2004) NC, ( XD(NN), YD(NN), NN=N, M)
WRITE(6,2004)
9 IF( IFRAME.EQ.999) GO TO 6
GO TO 1
9 IF( JIF.EQ.999) GO TO 999
WRITE(6,2009)
IF( JIF.EQ.999) GO TO 2
ENCOD(707, FPM) IFRAME
707 FORMAT(13, ' POLYGONS'
CALL SYMBOL( 0.0, -0.669, 0.125, FPM, 0.0, 12)
J = 0
REWIND 25
CALL NEWPEN(2)
DO 11 I = 1, J, 4
11 CALL SYMBOL(STAT(I)-XM, STAT(I+1)-YM, 0.1875, 3, 0.0, -1)
CALL NEWPEN(1)

A-II-3
10 READ(25)K , (XD(I), YD(I), I=1,K)
   J = J + 1
   IPEN = 3
   DO 20 I = 1,K
   IF( XD(I) )20,20,
      CALL PLOT(XD(I)-XM, YD(I)-YM, IPEN)
20 IPEN = 2
   IF( IFRAME .EQ. J) GO TO 99
   GO TO 10
99 CALL PLOT(ABS(XMX-XM+2.00), 0.0, -3)
   GO TO 2
C
900 IF( JOE.EQ.777) GO TO 999
   CALL FACTOR(0.5)
   CALL PLOT(2.5, 0.0, 999)
999 STOP
END

A-II-4
APPENDIX III

PLOTTER EDITING PROGRAM LISTING

MMS Map Case
DIMENSION DATA(14)
COMMON IONDF
1002 FORMAT(A9)
2000 FORMAT(1HO,110(***))
READ(5,1002) IONDF
IF(IONDF .EQ. 'ON') CALL PLOTS
IF(IONDF .EQ. 'OFF') CALL PLOTS(0,0,20)
REWIND 27

CALL TPLT
REWIND 25
REWIND 26
WRITE(6,2000)
CALL ANONY
100 CONTINUE
END FILE 27
REWIND 27
WRITE(6,2001)
2001 FORMAT(1H1)
105 CONTINUE
READ(27,END=200) DATA
WRITE(6,2002) DATA
GO TO 105
200 CONTINUE
2002 FORMAT(1H,14A6)
CALL PLOT(0,0,999)
END

"TFOR, IS TPLT,TPLT"
SURROUTINE TPLT
COMMON IONDF
DIMENSION DATA(14)
C
1000 FORMAT(5XA3,7XA3,2X4F6.3)
1001 FORMAT(13A6,1P)
1002 FORMAT(13A6,1P)
2000 FORMAT(1HO,5X'POINTS',15,3X'XMIN,XMAX,YMIN,YMAX=',4F10.3)
2001 FORMAT(1HO, 'BAD POINT',14,2(3XA3),4F10.3)
2003 FORMAT(1HO,'LINE LEFT RIGHT XL YL XD YD *YR*/)
2004 FORMAT(1H+11X***)
2005 FORMAT(1H,14A6)
2006 FORMAT(1H1,14A6/)
3001 FORMAT(13)
C
PPD=57,2967P
FACT=1.
IF(IONDF .EQ. 'OFF') FACT=2.
HT=.125/FACT
CALL FACTOR ( FACT )
C
9 NOTES=0
REWIND 25
READ(5,1001,END=400) DATA
WRITE(27) DATA
WRITE(6,2006) DATA
XMIN=1000.
XMAX=-1000.
YMIN=1000.
YMAX=-1000.
CONTINUE
READ (5,1000,END=200,ERR=2) NUM1,NUM2,X1,Y1,X2,Y2
IF (NUM1.EQ.1000) GO TO 1
IF(NUM1.EQ.1 AND NUM2.EQ.1) GO TO 1
IF(FLD(5.6,NUM1).LE.80) GO TO 1
NPTS=NPTS+1
XMIN=MINT(XMIN,X1,X2)
XMAX=MINT(XMAX,X1,X2)
YMIN=MINT(YMIN,Y1,Y2)
YMAX=MINT(YMAX,Y1,Y2)
WRITE(25)NUM1,NUM2,X1,Y1,X2,Y2
GO TO 1
2 READ(0,1001)DATA
WRITE(5,2005)DATA
GO TO 1
200 CONTINUE
IF (NPTS.EQ.0) GO TO 400
ENC FILE 25
DO 500I12=1,2
NL=0
IF(I12.EQ.2)GO TO 202
YSIZE=YMAX-YMIN
WRITE(4,2000)YPTS,XMIN,XMAX,YMIN,YMAX
WRITE(6,2001)
202 CONTINUE
CALL SYMBOL(0...25,140,DATA,0...60)
DO WIND 25
201 CONTINUE
READ(25,FN=300)NUM1,NUM2,X1,Y1,X2,Y2
NL=NL+1
IF(I12.EQ.1)WRITE(6,2002)NL,NUM1,NUM2,X1,Y1,X2,Y2
IF(I12.EQ.2)ENCER(3001,NLI)NL
CALL SYMBOL(X1-XMIN,Y1-YMIN+0.5,1.0,-1)
CALL SYMBOL(X2-XMIN,Y2-YMIN+0.5,2.0,-2)
AL=SORT((X2-X1)**2+(Y2-Y1)**2)
IF(AL.EQ.1)GOTO 207
IF(T12.EQ.1)WRITE(6,2004)
GO TO 201
207 XC=(X1+X2)/2
YC=(Y1+Y2)/2,
CALL ANGLe(X2-X1,Y2-Y1,THETA,$205)
THETA=THETA#DDR
CS=COS(THETA)
SN=SIN(THETA)
XOF=1.5*HT*CS
YOF=1.5*HT*SN
XOF1=0.20*SN
YOF1=0.20*CS
IF(I12.EQ.2)NUM1=NL
CALL SYMBOL(XC-XOF-XOF1-XMIN,YC-YOF+YOF1-YMIN,XT,H,U,M,1,TH0,3)
IF(I12.EQ.2)GO TO 201
XOF=1.020+HT)*SN
YOF=1.020+HT)*CS
CALL SYMBOL(XC-XOF-XOF2-XMIN,YC-YOF-YOF2-YMIN,HT,H,M,2,TH0,3)
GO TO 201
205 WRITE(4,2001)NPTS,NUM1,NUM2,X1,Y1,X2,Y2
GO TO 201
300 CONTINUE
IF(YSIZE*FACT.LT.16.)GO TO 305
CALL PLOT(XMAX-XMIN+2.0,.0,.3)
GO TO 500
305 IF(I12.EQ.1)CALL PLOT(0,YSIZE+1,-3)
IF(112 .EQ. 2) CALL PLOT(XMAX-XMIN+2.,YSIZE-1.,-3)

500 CONTINUE
GO TO 4
A00 CONTINUE
RETURN
END

**FORTRAN**

ANGLE, ANGLE

SUBROUTINE ANGLE(X,Y,THETA,*)

C
SUBROUTINE TO DETERMINE ANGLE (0 TO 2*PI) FROM X-Y PAIR

C
PI=3.1415926
C
IF(X)10,20,30
C
SECOND OR THIRD QUADRANT
C
10 IF(Y)11,12,11
   11 THETA=PI+ATAN(Y/X) " THIRD QUADRANT
   RETURN
   12 THETA=PI " 180 DEGREES
   RETURN
C
ON Y AXIS (PI/2 OR 3*PI/2)
C
20 IF(Y)21,22,23
   21 THETA=1.5*PI " 270 DEGREES
   RETURN
   22 RETURN 4 " DEGENERATE CASE
   23 THETA=PI/2. " 90 DEGREES
   RETURN
C
FIRST OR FOURTH QUADRANT
C
30 IF(Y)31,32,31
   31 THETA=ATAN(Y/X)
   IF(THETA .LT. 0.) THETA=2.*PI+THETA
   RETURN
   32 THETA=0. " 00 DEGREES
   RETURN
C
END

**FORTRAN**

BNDRY,BNDRY

SUBROUTINE BNDRY

DIMENSION DATA(2)
C
1000 FORMAT(5X,A3,7X,A3,2X,F4.1)
1001 FORMAT(5X,A6,2X,A/)  
2000 FORMAT(1H,A3,A4,F10.3)
C
DATA HT/*.125/
DATA XMIN,XMAX,YMIN,YMAX/1000.,-1000.,1000.,-1000.*/
NPTS=0
REWIND 25
REWIND 26
CALL FACTOR(1.)
CONTINUE
READ(5,1001) END=4001 DATA
XMIN=1000.
XMAX=-1000.
YMIN=1000.
YMX=-1000.

1 CONTINUE
READ(5,1000,END=200) NUM1,NUM2,X1,Y1,X2,Y2
IF(NUM2 .NE. '*' ) GO TO 1
IF(NUM1 .EQ. '*' ) GO TO 1
NPTS=NPTS+1
XMIN=MIN(XMIN,X1,X2)
XMAX=MAX(XMAX,X1,X2)
YMIN=MIN(YMIN,Y1,Y2)
YMAX=MAX(YMAX,Y1,Y2)
XMN=MIN(XMN,X1,X2)
XM=MAX(XMX,X1,X2)
YM=NMIN(YMN,Y1,Y2)
YM=MAX(YMX,Y1,Y2)
WRITE(25) NUM1,X1,Y1,X2,Y2
GO TO 1

200 CONTINUE
WRITE(26) DATA,XMN,XMX,YMN,YMX
GO TO 5

400 CONTINUE
IF(NPTS .NE. 0) RETURN
END FILE 25
END FILE 25
REWIND 25
REWIND 25
401 CONTINUE
READ(26,END=410) DATA,XMN,XMX,YMN,YMX
CALL SYMBOL((XMN+XMX)/2.-XMIN-3.5*HT,(YMN+YMX)/2.-YMIN-5*HT,HT,DATA,0,0,0,7)
GO TO 401

410 CONTINUE
411 READ(25,END=420) NUM1,X1,Y1,X2,Y2
WRITE(6,2000) NUM1,X1,Y1,X2,Y2
CALL SYMBOL(X1-XMIN,Y1-YMIN,06,1,0,-1)
CALL SYMBOL(X2-XMIN,Y2-YMIN,06,2,0,-2)
GO TO 410

420 CONTINUE
CALL PLOT(XMAX-XMIN+4,0,-3)
RETURN
END
APPENDIX IV

ROTATE PROGRAM LISTING
C YOU MUST DIMENSION THE NUMBER OF ROWS OF A(*,4), THE NUMBER OF COLUMNS OF AT(*,4), AND THE NUMBER OF COMPONENTS OF B(*) AND C(*).

C(*) EQUAL TO TWO TIMES THE NUMBER OF CONTROL POINTS USED ON MAP

DOUBLE PRECISION A(12,4), AT(4,12), ATAT(4,4), B(12), C(12).

DIMENSION X(12), Y(12), XO(12), YO(12), U(4), ATA(4,4).

C C(14), L(14), THETA(2), THECOM(2,3), IX(12), IY(12)

INTEGER XX

FORMAT(15) 00020
FORMAT(8F9.1) 00030
FORMAT(12F6.3) 00040
FORMAT(12F6.3) 00050
FORMAT(2X9I2F7.3) 00060
FORMAT(IHI) 00070
FORMAT(///s9) 00080
FORMAT(//,9) 00090
FORMAT(12F6.3) 00100
FORMAT(///,9) 00110
FORMAT('DIGITIZATION ORIGIN (XO,YO) AND THE TABLE TO MAP ROTATION') 00120
FORMAT('ANGLE THETA') 00130
FORMAT('SCALING FACTOR S = ',F10.3) 00140
FORMAT('FOR THIS POLYGON —MIN X, MIN Y, MAX X, MAX Y, NUMBER OF POINTS IN POLYGON') 00150
FORMAT('CALIFORNIA STATE PLANE COORDINATES FOR THIS POLYGON') 00160
FORMAT('FOLLOW IN GROUPS,(3F TWELVE PAIRS') 00170
FORMAT('FOR THESE CONTROL POINTS, THE MAP POSITION IS OFF BY •') 00180

*** ASCII COPY ***
C NPOLY TOTAL NUMBER OF POLYGONS T0484
C M NUMBER OF CONTROL POINTS T0485
C XOTOL X COORDINATE TOLERANCE LEVEL T0486
C YOTOL Y COORDINATE TOLERANCE LEVEL T0487
C THETOL ROTATION ANGLE TOLERANCE LIMIT T0488
C
WRITE(6,12)NPOLY,M,XOTOL,YOTOL,THETOL T0489
M3 = MOD(M,4) 00491
IF(M3.NE.0) GO TO 23 T0492
M3 = 4 00500
23 IF(M.LE.4) GO TO 21 T0493
M1 = (M-1)/4 T0494
DO 25 12 = 1,M1 T0495
ISUB = 4*(12-1) T0496
25 READ(5,11) (X(ISUB+13),Y(ISUB+13),I3 = 1,4) T0497
C READ CALCULATED SPC CONTROL POINTS T0498
C X CONTROL POINT X T0499
C Y CONTROL POINT Y T0500
C ISUB = 4#M1 T0501
GO TO 22 T0502
21 ISUB = 0 00503
22 READ(5,11) (X(ISUB+13),Y(ISUB+13),I3 = 1,4) T0504
M3 = MOD(M,6) T0505
IF(M3.NE.0) GO TO 28 T0506
M3 = 6 00510
28 IF(M.LE.6) GO TO 27 T0507
M1 = (M-1)/6 T0508
DO 30 12 = 1,M1 T0509
ISUB = 6*(12-1) T0510
30 READ(5,13) (XD(ISUB+13),YD(ISUB+13),I3 = 1,6) T0511
C READ DIGITIZED CONTROL POINTS T0512
C XD DIGITIZED CONTROL POINT X T0513
C YD DIGITIZED CONTROL POINT Y T0514
C ISUB = 6#M1 T0515
GO TO 29 T0516
27 ISUB = 0 00517
29 READ(5,13) (XD(ISUB+13),YD(ISUB+13),I3 = 1,M3) T0518
C INITIAL ARRAYS AND INPUT DATA T0519
C FROM CALCULATED & DIGITIZED COORDINATES T0520
C
NW1 = 2*M T0521
DO 50 I = 1,N T0522
I1 = 2*I-1 T0523
A(I1,1) = 1* T0524
A(I1,2) = 0* T0525
A(I1,3) = XD(I) T0526
A(I1,4) = YD(I) T0527
C = (I) T0528
I1 = I1+1 T0529
A(I1,1) = 0* T0530
A(I1,2) = 1* T0531
A(I1,3) = XD(I) T0532
A(I1,4) = -YD(I) T0533

ORIGINAL PAGE IS
OF POOR QUALITY

A-IV-3
50 B(I) = Y(I)
DO 34 I = 1,4
DO 34 J = 1,MM1
34 AT(I,J) = A(J,I)
CALL RATMUL(ATA,AT,A,4,MM1,4)
CALL INVERS(ATA,4)
CALL RATMUL(ATA,AT,A,4,MM1)
DO 36 I = 1,4
U(I) = 0.
DO 36 J = 1,MM1
36 U(I) = U(I) + ATAT(I,J)*B(J)
60 UX = U(3)*U(3) + U(4)*U(4)
S = SQRT(UX)
UX = U(3)/S
THETA(1) = ARCOS(UX)
UX = U(4)/S
THETA(2) = ARSIN(UX)
IF(U(3)*GE.0.) GO TO 263
THETA(1) = 3.14159265 - THETA(1)
263 CONTINUE
DO 45 I = 1,2
THETA(I) = ABS(THETA(I))
IF(U(3)*GE.0.) GO TO 260
IF(U(4)*GE.0.) GO TO 261
THETA(I) = 3.14159265 + THETA(I)
GO TO 262
261 THETA(I) = 3.14159265 - THETA(I)
GO TO 262
260 IF(U(4)*GE.0.) GO TO 262
THETA(I) = 6.28318530 - THETA(I)
45 CONTINUE
THETA(I) = THETA(I)*180./3.14159265
THECOM(I,1) = AINT(THETA(I))
DUM = (THETA(I) - THECOM(I,1))*60.
THECOM(I,2) = AINT(DUM)
DUM = (DUM - THECOM(I,2))*60.
THECOM(I,3) = AINT(DUM)
C WRITE HEADING "FOLLOWING IS A CHECK ON"
C WRITE(6,14)
C WRITE(6,15)
C WRITE(6,16)
C WRITE(6,229)
C WRITE(6,17)
DO 47 I = 1,2
47 WRITE(6,18) U(1),U(2),THECOM(I,1),THECOM(I,2),THECOM(I,3)
C WRITE RESULTS OF ROTATION FROM CALCULATED AND DIGITIZED POINTS
C WRITE(6,220) S
C WRITE SCALING FACTOR - S
C TEST FOR ROTATION ANGLE AND SCALING FACTOR WITHIN TOLERANCE

C
C
DO 32 I = 1,M
C(I) = 0.
DO 32 J = 1,N
32 C(I) = C(I) + A(I,J)*U(J)
DO 33 I = 1,M
UX=(C(I)-B(I))/S
C(I) = ABS(UX)
33 CONTINUE
K = 0
L = 0
DO 37 I = 1,M
II = 2*I-1
IF(C(II).LT.XOTOL) GO TO 40
K = K+1
KI(K) = I
40 CONTINUE
IF(K.NE.0) GO TO 97
IF(L.NE.0) GO TO 97
IF((THE (I1)-THETA(2))*.60.).GT.THETOL) GO TO 97

C COMPLETION OF TOLERANCE CHECKS

C * SCALE AND ROTATE DIGITIZED POINTS, MIN-MAX & BOUNDARIES
C *** CODED WITHIN THE FRAMEWORK OF THE GEO-REF SYSTEM - PHASE I

C DO 65 I = 1,2
65 THETA(1) = (THETA(1)+THETA(2))/2.
A(1,1) = S*COS(THETA(1))
A(2,1) = -A(1,1)
WRITE(6,14)
DO 102 IB = 1,NPOLY
READ(S,19) MAPNO,MAPOL,LUCODE,POLYNO,PTYPE

C READ POLYGON CONTROL CARD

C MAPNO MAP NUMBER FOR THE POLYGONS THAT FOLLOW
C MAPOL TOTAL NUMBER OF POLYGONS ON MAP
C LUCODELAND USE OF POLYGON CODE
C POLYNO SEQUENCE NUMBER OF POLYGON U TO MAPOL
C PTYPEPOLYGON TYPE-KEY TO DONUT POLYGONS
C NORMAL POLYGONS ARE CODED 10

C READ(S,20) XMIN,YMIN,XMAX,YMAX,NP
C READ MINIMUM & MAXIMUM COORDINATES

C XMIN MINIMUM X COORDINATE
C YMIN MINIMUM Y COORDINATE
C XMAX MAXIMUM X COORDINATE
C YMAX MAXIMUM Y COORDINATE

ORIGINAL PAGE IS OF POOR QUALITY A-IV-5
C NUMER OF POINTS IN POLYGON
X = (NP-I)/6
NP1 = 1 +XX
13E = NP-6*(NP1-I)
NPF = (NP1+1)/2
NPER = (2*NPE)-NP1
WRITE(6,14)
WRITE(6,221)
WRITE(6,219) MAPNO, MAPOL, LUCODE, POLYN0, PTYPE

C WRITE POLYGON CONTROL CARD
WRITE(7,19) MAPNO, MAPOL, LUCODE, POLYN0, PTYPE
WRITE(6,222)
DO 75 I = 1,4
IF(I.EQ.4) GO TO 79
IF(I.EQ.2) GO TO 77

C ROTATE AND SCALE MINIMUM AND MAXIMUM COORDINATES
DUM1 = XMIN
DUM2 = YMIN
GO TO 76
77 DUM1 = XMAX
DUM2 = YMIN
GO TO 76
78 DUM1 = XMIN
DUM2 = YMAX
GO TO 76
79 DUM1 = XMAX
DUM2 = YMAX
C(T) = U(I) + A(1,1)*DUM1 + A(1,2)*DUM2
75 B(I) = U(2) + A(2,1)*DUM1 + A(2,2)*DUM2
XMIN = C(I)
DO 80 I = 2,4
IF(XMIN.LE.C(I)) GO TO 80
XMIN = C(I)
80 CONTINUE
XMAX = C(I)
DO 82 I = 2,4
IF(XMAX.GE.C(I)) GO TO 82
XMAX = C(I)
82 CONTINUE
YMIN = B(I)
DO 84 I = 2,4
IF(YMIN.LE.B(I)) GO TO 84
YMIN = B(I)
84 CONTINUE
YMAX = B(I)
DO 86 I = 2,4
IF(YMAX.GE.B(I)) GO TO 86
YMAX = B(I)
86 CONTINUE
IXMIN = INT(XMIN*10)
IYMIN = INT(YMIN*10)
IXMAX = INT(XMAX*10)
IYMAX = INT(YMAX*10)

C ROTATION AND SCALING COMPLETED FOR MIN-MAX COORDINATES
WRITE(6,228) IXMIN,IYMIN,IXMAX,IYMAX,NP
WRITE(6,229) IXMIN,IYMIN,IXMAX,IYMAX,NP
WRITE(6,223)
WRITE(6,224)
WRITE(6,222)
BEGIN SCALING AND ROTATION FOR POLYGON BOUNDARY
DO 55 ISM = 1,NPE
WRITE(6,233)
COMPLEX ROUTINE FOR COMPENSATING TO VARIATION IN INPUT FORMAT
NP = NUMBER OF COORDINATE PAIRS
NPU = NUMBER OF CARDS TO BE READ
ISE = PARTIAL COORDINATE PAIRS REMAINING AFTER COMPLETE CARD READ
NPE = NUMBER OF GROUPS OF COORDINATES
NPER = IF LAST SET CONTAINS ONLY ONE CARD
IF(ISM.NE.NPE) GO TO 61
IF(NPER.FO .I) GO TO 63
READ(5,13) (XD(I2),YD(I2), I2 = 1,6)
IBEG = 7
IEND = 6 + I3F
GO TO 64
63 IREG = 1
TEND = I3F
64 READ(5,13) (XD(I2), YD(I2), I2 = IBEG,IEND)
GO TO 67
61 CONTINUE
DO 62 IL=1,2
LSUB = 6*(IL-1)
READ(5,13) (XD(LSUB+I2),YD(LSUB+I2),I2 = 1,6)
67 CONTINUE
IF(ISM.NE.NPE) GO TO 68
NPP = 12
GO TO 69
68 NPP = 6*(1-NPER) + I3F
69 CONTINUE
END OF POLYGON READ ROUTINE
CALCULATE ROTATION SCALING FACTOR
DO 70 I = 1,NPP
X(I) = U(I) + A(I,1)*XD(I) + A(I,2)*YD(I)
Y(I) = U(2) + A(2,1)*XD(I) + A(2,2)*YD(I)
IF(ISM.NE.NPE) GO TO 81
DO 140 TL=1,2
LSUB = 6*(IL-1)
WRITE(6,213) (XD(LSUB+I2),YD(LSUB+I2),I2 = 1,6)
WRITE COORDINATES TO PRINTER
GO TO 72
81 IF (NP+5 *. EQ. 1) GO TO 141
   WRITE (6, 213) (XD(I2), YD(I2), I2=1, N)
   IBE = 7
   IEND = 6+13E
   GO TO 142
141 IBE = 1
   IEND = 13E
142 WRITE (6, 213) (XD(I2), YD(I2), I2=1BEGIEND)
72 IF (ISW.EQ.NPE) GO TO 74
C
C     CONVER COORDINATES TO INTEGER
C
57 \ DO 66 I2 = 2, N
56 \ IX(I2) = X(I2)*10
55 \ IY(I2) = Y(I2) * 10
DO 143 IL = 1, I4E
   LSUB = 4*(IL-1)
   WRITE (6, 235) (IX(LSUB+I2), IY(LSUB+I2), I2=1, 4)
C
C     WRITE INTEGER COORDINATES TO CARD FILE
C
143 WRITE (7, 236) (IX(LSUB+I2), IY(LSUB+I2), I2=1, 4)
   GO TO 55
74 \ XX = (NPP-1)/4
   NPP = 1+XX
   I4E = NPP - 4*(NPP-1)
   \ DO 57 I2 = 1, NPP
57 \ IX(I2) = X(I2)*10
   \ DO 145 IL = 1, NPP
   LSUB = 4*(IL-1)
   IF (IL.EQ.NPP) GO TO 147
   WRITE (6, 235) (IX(LSUB+I2), IY(LSUB+I2), I2=1, 4)
   WRITE (7, 236) (IX(LSUB+I2), IY(LSUB+I2), I2=1, 4)
   \ GO TO 145
147 \ WRITE (6, 235) (IX(LSUB+I2), IY(LSUB+I2), I2=1, I4E)
   \ WRITE (7, 236) (IX(LSUB+I2), IY(LSUB+I2), I2=1, I4E)
145 \ CONTINUE
55 \ CONTINUE
102 \ CONTINUE
   \ GO TO 100
C
C     ERROR CONDITION
C
87 IF (K.EQ.0) GO TO 98
   WRITE (6, 226) YOTOL, IBIG, (KI(I), I = 1, K)
   WRITE (6, 230)
   \ DO 87 I = 1, K
87 \ IWR = 2*K1(I)-1
88 \ WRITE (6, 231) C(IWR)
98 IF (L.EQ.0) GO TO 99
   WRITE (6, 226) YOTOL, IBIG, (LI(I), I = 1, L)
   WRITE (6, 230)
   \ DO 98 I = 1, L
98 \ IWR = 2*LI(I)
97 \ WRITE (6, 231) C(IWR)
99 IF ((THETA(I)-THETA(J))>60.) LE THETOL) GO TO 100
   WRITE (6, 227) THETOL, IBIG
100 \ CONTINUE
  \ STOP

A-IV-8
SUBROUTINE RATMUL(C,A,B,I,J,K)
DOUBLE PRECISION C(4,4), A(I,J), B(J,K)
DO 100 IR = 1:I
DO 100 JC = 1:K
C(IR,JC) = 0.
DO 100 L = 1:J
C(IR,JC) = C(IR,JC) + A(IR,L)*B(L,JC)
100 CONTINUE
RETURN
END
SUBROUTINE INVERS(ATA,N)
DOUBLE PRECISION ATA(N,N)
DIMENSION XTA(4,4)

FORMAT(1X,* NO SOLUTION. A 0 EXISTS ON THE MAIN DIAGONAL FOR*)
FORMAT(* ROW NUMBER ,13* COLUMN NUMBER ,13)
FORMAT(* REMEMBER-ROWS MAY HAVE BEEN SWITCHED PREVIOUS TO THIS *)
FORMAT(• POINT. NO MORE SWITCHING POSSIBLE*)
DO 301 IH = 1:N
DO 306 JJ = 1:N
XTA(IH,JJ) = ATA(IH,JJ)
A(IH,JJ) = 0.0
A(I,I) = 1.0
DET = 1.0
GO TO 308
100 A(I,I) = I.H
IG = IH
IF (ABS(XTA(IH,H)) GE 1.E-10) GO TO 332
C SWAP ROWS
IF (IG LE N) GO TO 308
WRITE(6,302)
WRITE(6,303) IH, IH
WRITEF(6,301)
WRITE(6,304)
GO TO 381
IF (ABS(XTA(IH,G)) LT 1.E-10) GO TO 305
DET = (-1.0)*DET
DO 314 JJ = 1:N
C = ATA(IH,JJ)
D = A(IH,JJ)
ATA(IH,JJ) = ATA(IH,JJ)/D
A(IH,JJ) = A(IH,JJ)/D
ATA(IH,JJ) = C
A(IH,JJ) = D
305 IG = IG+1
IF (IG LE N) GO TO 308
WRITE(6,302)
WRITE(6,303) IH, IH
WRITEF(6,301)
WRITE(6,304)
GO TO 381
IF (ABS(XTA(IH,G)) LT 1.E-10) GO TO 305
DET = (-1.0)*DET
ATA(IH,JJ) = ATA(IH,JJ)/DIAG
A(IH,JJ) = A(IH,JJ)/DIAG
332 DIAG = ATA(IH,IH)
DET = DET*DIAG
DO 336 JJ = 1:N
ATA(IH,JJ) = ATA(IH,JJ)/DIAG
334 A(IH,JJ) = A(IH,JJ)/DIAG
IF (IH LE N) GO TO 336
336 IN1 = N-1
DO 338 I = 1,IN1
IH = N+1-I
II = IH-1
CONTINUE
END
352 DIAG = ATA(I,1,1)  
   IG = N  
346 IF(IG .EQ. 0) GO TO 342  
   ATA(I,IG) = ATA(I,IG) - (DIAG*ATA(IH,IG))  
   A(I,IG) = A(I,IG) - (DIAG*A(IH,IG))  
   IG = IG - 1  
   GO TO 346  
342 IF(I .EQ. 1) GO TO 348  
   I = I-1  
   GO TO 352  
348 CONTINUE  
354 DO 380 I = 1,N  
   DO 380 J = 1,N  
380 ATA(I,J) = A(I,J)  
381 RETURN  
END
APPENDIX V

CHAIN PROGRAM LISTING
C YOU MUST DIMENSION THE NUMBER OF ROWS OF A(*,9), THE NUMBER OF COLUMNS OF AT(*,9), AND ATAT(*,9) AND THE NUMBER OF COMPONENTS OF B(*) AND C(*) EQUAL TO TWO TIMES THE NUMBER OF CONTROL POINTS USED ON MAP. C MAKE SURE THAT LAST BLOCK DIGITIZED IS AN INTERNAL BLOCK. C AND NOT ON TRACT BOUNDARY.

INTEGER XX

DIMENSION A(B,9),AT(4,9),ATAT(4,9),B(9),C(9),X(12),Y(12),
       1X(12),Y(12),K(4),L(4),U(4),ATA(4,4),THECOM(2,3),
       2X(12),Y(12),ITLEFT(1000),ITRITE(1000),IBLOCK(100),XM(100),
       3Y(100)

DIMENSION XL(1000),YL(1000),XR(1000),YR(1000),KTBL(100)

10 FORMAT(I5)
11 FORMAT(8F9.1)
12 FORMAT(215.3F10.3)
13 FORMAT(12F6.3)
14 FORMAT(2X12F6.3)
15 FORMAT(1H1)
16 FORMAT(15x)
17 FORMAT(14X,'X0,Y0',14X,'THETA')
18 FORMAT(2F20.1,3X,'F4.0','MIN','F4.0','SEC')
19 FORMAT(2I5,13x,12)
20 FORMAT(4F6.3)
21 FORMAT(' THE SCALING FACTOR S = ',F10.1)
22 FORMAT('/. FOR THIS POLYGON — MIN X, MIN Y, MAX X, MAX Y, NUMBER OF 1 POINTS IN POLYGON')
23 FORMAT('/. ALL DIGITIZED COORDINATES AND TRANSFORMED ZONE ?')
24 FORMAT( 'CALIFORNIA STATE PLANE COORDINATES FOR THIS POLYGON')
25 FORMAT( 'FOLLOW IN GROUPS OF TWELVE PAIRS')
26 FORMAT( 'FOR THESE CONTROL POINTS, THE MAP POSITION IS OFF BY ')
27 FORMAT( 'YOUR XO ORIGIN REFERENCE SOLUTIONS DISAGREE BY MORE THAN ',F10.3,' MINUTES FOR MAP SETUP ')
28 FORMAT( 'YOUR YO ORIGIN REFERENCE SOLUTIONS DISAGREE BY MORE THAN ')
29 FORMAT( 'YOUR THETA ROTATION SOLUTIONS DISAGREE BY MORE THAN ',F10.3,' MINUTES FOR MAP SETUP ')
30 FORMAT( 'THE STREET SEGMENT ITS FOLLOW ')
31 FORMAT( 'THE NON-POLYGON CUL DE SACS FOLLOW')
32 FORMAT(19I9,19)
BEGIN MAJOR LOOP OF PROGRAM - ISIG

DO 100 IBIG = 1, KSET
READ(5,12) NPOLY, M, XOTOL, YOTOL, THETOL

READ POLYGON TOLERANCE CONTROL CARD

NPOLY	 TOTAL NUMBER OF POLYGONS
M	 NUMBER OF CONTROL POINTS
XOTOL	 X COORDINATE TOLERANCE LEVEL
YOTOL	 Y COORDINATE TOLERANCE LEVEL
THETOL	 ROTATION ANGLE TOLERANCE LIMIT

COMPLEX ROUTINE TO READ COORDINATES IN STRING FORMAT

M3 = MOD(M, 4)
IF(M3 .NE. 0) GO TO 23
M3 = 4
23 IF(M .LE. 4) GO TO 21
M1 = (M - 1) / 4
DO 25 I2 = 1, M1
ISUB = 4 * (I2 - 1)
READ(5, 11) (X(ISUB + I3), Y(ISUB + I3), I3 = 1, M3)
25 READ(DS, 11) (X(ISUB + I3), Y(ISUB + I3), I3 = 1, M3)

READ CONTROL POINTS

X	 CONTROL POINT X
Y	 CONTROL POINT Y

ISUB = 4 * M1
GO TO 22
21 ISUB = 0
22 READ(5, 11) (X(ISUB + I3), Y(ISUB + I3), I3 = 1, M3)
M3 = MOD(M, 6)
IF(M3 .NE. 0) GO TO 28
M3 = 6
28 IF(M .LE. 6) GO TO 27
M1 = (M - 1) / 6
DO 30 I2 = 1, M1
ISUB = 6 * (I2 - 1)
READ(5, 13) (XD(ISUB + I3), YD(ISUB + I3), I3 = 1, M3)
30 READ(DS, 13) (XD(ISUB + I3), YD(ISUB + I3), I3 = 1, M3)

READ DIGITIZED CONTROL POINTS

XD	 DIGITIZED CONTROL POINT X
YD	 DIGITIZED CONTROL POINT Y

ISUB = 6 * M1
GO TO 29
27 ISUB = 0
29 READ(5, 13) (XD(ISUB + I3), YD(ISUB + I3), I3 = 1, M3)

INITIAL ARRAYS AND INPUT DATA

FROM CALCULATED & DIGITIZED COORDINATES

MM1 = 2 * M
DO 50 I = 1, M
I1 = 2 * I - 1
A(I1, 1) = 1.
A(I1, 2) = 0.
A(I1, 3) = XD(I)
A(I+4) = YD(I)
U(I) = X(I)
I = I+1
A(I+1) = 0.
A(I+2) = 1.
A(I+3) = YD(I)
A(I+4) = XD(I)
50 B(I) = Y(I)
DO 34 I = 1,4
DO 34 J = 1,MM1
34 AT(I,J) = A(J,I)
CALL RATMUL(ATAT,ATAT,4,MM1,4)
CALL INVERS(ATAT,4)
CALL RATMUL(ATAT,ATAT,4,4,4,MM1)
DO 36 I = 1,4
U(I) = 0.
DO 36 J = 1,MM1
36 U(I) = U(I) + ATAT(I,J)*B(J)
60 S = SORT(U(3)*U(3) + U(4)*U(4))
THETA(1) = ARCOS(U(3)/S)
THETA(2) = ARSIN(U(4)/S)
IF(U(3) .GE. 0.) GO TO 263
THETA(1) = 3.14159265 - THETA(1)
263 CONTINUE
DO 45 I = 1,2
THETA(I) = ABS(THETA(I))
IF(U(3) .GE. 0.) GO TO 260
IF(U(4) .GE. 0.) GO TO 261
THETA(1) = 3.14159265 + THETA(1)
GO TO 262
261 THETA(I) = 3.14159265 - THETA(I)
GO TO 262
260 IF(U(4) .GE. 0.) GO TO 262
THETA(I) = 6.28318530 - THETA(I)
262 CONTINUE
THETA(I) = THETA(I)*180./3.14159265
THECOM(I,1) = AINT(THETA(I))
DUM = (THETA(I) - THECOM(I,1))*60.
THECOM(I,2) = AINT(DUM)
DUM = (DUM - THECOM(I,2))*60.
45 THECOM(I,3) = AINT(DUM)
C WRITE HEADING "FOLLOWING IS A CHECK ON"
C WRITE RESULTS OF ROTATION FROM CALCULATED AND DIGITIZED POINTS
C U(1) ORIGIN X0
C U(2) ORIGIN Y0
C THECOM(I,1) ROTATION DEGREES
C THECOM(I,2) ROTATION MINUTES
C THECOM(I,3) ROTATION SECONDS
C WRITE(6,14)
WRITE(6,15)
WRITE(6,16)
WRITE(6,229)
WRITE(6,17)
DO 47 I = 1,2
47 WRITE(6,16) U(I), U(2), THECOM(I,1), THECOM(I,2), THECOM(I,3)
WRITE(6,220) S
WRITE SCALING FACTOR - S

TEST FOR ROTATION ANGLE AND SCALING FACTOR WITHIN TOLERANCE

DO 32 I = 1,IMM1
   C(I) = 0
DO 32 J = 1,4
   C(I) = C(I) + A(I,J)*U(J)
DO 33 I = 1,IMM1
   C(I) = ABS(C(I)-B(I))/S
   K = 0
   L = 0
   DO 37 I = 1,IMM1
      I1 = 2*I-1
      IF(C(I1).LT.XOTOL) GO TO 40
      K = K+1
      K1(K) = I1
   K1(K) = I1
   IF(C(I1).LT.YOTOL) GO TO 37
   L = L+1
   L1(L) = I1
CONTINUE
   IF(K.NE.0) GO TO 97
   IF(L.NE.0) GO TO 97
   IF(ASS((THETA(1)-THETA(2))*60.).GT.THETOL) GO TO 97

COMPLETION O= TOLERANCE CHECKS

DO 65 I = 1+2
   THETA(I) = THETA(I)*3.14159265/180.
   A(1,1) = S*COS(THETA(I))
   A(1,2) = S*SIN(THETA(I))
   A(2,1) = -A(1,2)
   A(2,2) = A(1,1)
WRITE(6,14)

*---------------------------------------------------------------------*
* PHASE II
* REVERSE AND DUPLICATE SEGMENTS WHICH SERVE AS A BOUNDARY
* TO TWO BLOCKS AND CHECK FOR SAME BEGINNING AND ENDING COORD
*---------------------------------------------------------------------*

DO 102 IB = 1,NPOLY
   NPL = 2*NPOLY - I
   DO 370 I = 1,NPOLY
      ITLEFT(NPOLY+I) = ITLEFT(I)
      ITRITE(NPOLY+I) = ITRITE(I)
      XL(NPOLY+I) = XR(I)
      YL(NPOLY+I) = YR(I)
      XR(NPOLY+I) = XL(I)
      YR(NPOLY+I) = YL(I)

WE WANT TO GET RID OF ALL TRACT BOUNDARY SEGMENTS WITH OTHER THAN
PRESENT TRACT ON LEFT.
WE ASSUME THAT LAST SEGMENT DIGITIZED (NPOLY) HAS OUR DESIRED TRACT
ON LEFT. IT IS AN INTERNAL BLOCK.
NURTAK = ITLEFT(NPOLY)/10000
   I = 0
372 I = I+1
363 IACHTR = ITLEFT(I1)/10000
IF (I1.GT.0.NPV1) GO TO 375
IF (NURTRK .EQ. IACHTR) GO TO 372
DO 373 I1 = 1.NPV1
ITLEFT(I1) = ITLEFT(I1+1)
ITRITF(I1) = ITRITE(I1+1)
XL(I1) = XL(I1+1)
YL(I1) = YL(I1+1)
XR(I1) = XR(I1+1)
YR(I1) = YR(I1+1)
NP1 = NP1 - 1
GO TO 363
373 CONTINUE
IF (NURTRK .NE. IACHTR) GO TO 372
NP1 = NP1 - 1
GO TO 374
375 CONTINUE
CALL SORT6( ITLEFT, ITRITE, XL, YL, XR, YR, NP1)
NP1 = NP1 + 1
376 CONTINUE
C IBLKS IS THE NUMBER OF BLOCKS IN THIS TRACT
C NOW WE'LL CHAIN THE BLOCK TOGETHER IN SEQUENTIAL ORDER
IREGF = 0
TENDF = 0
DO 380 I10 = 1.NBLKS
NSEG = KTB7(I10)
NUMBLK = IBLKS(I10)
I10 = I10 + 1
380 CONTINUE
IF (I1.GT.0.NPV1) GO TO 378
IF (NUMBLK .NE- ITRITF(I1)) GO TO 374
378 CONTINUE
C CÚTCH CULS Æ SAC TO END OF BLOCK SEGMENTS
IDUM2 = ITRITE(I1)
DUM3 = XL(I1)
DUM4 = YL(I1)
DUM5 = XR(I1)
DUM6 = YR(I1)
ITRITF(I1) = ITRITE(I1)
XL(I1) = XL(I1)
YL(I1) = YL(I1)
XR(I1) = XR(I1)
YR(I1) = YR(I1)
ITRITE(I1) = IDUM2
XL(I1) = DUM3
YL(I1) = DUM4
XR(I1) = DUM5
YR(I1) = DUM6
C FINISHED SWITCHING
GO TO 382
384 NCULS = NCULS + 1
IMARK = IMARK - 1
IF(IMARK .LT. 1) GO TO 386
GO TO 385
382 CONTINUE
386 CONTINUE
C CUL DE SACS ARE NOW AT END OF BLOCK SEGMENTS
NSEG1 = NSEG - NCULS
C NOW WE'LL MATCH ENDS OF SEGMENTS TO BEGINNINGS OF OTHER SEGMENTS
C NSEG1 IS THE NUMBER OF BLOCK SEGMENTS EXCLUDING CUL DE SACS
II = IBEGF
IEND1 = IENDF - NCULS
IS = IREGF
390 IS = IS + 1
393 IF(IS .GT. IEND1) GO TO 392
IF(ABS(XR(II) - XL(IS)) .GT. XOTOL) GO TO 390
IF(ABS(YR(II) - YL(IS)) .GT. YOTOL) GO TO 390
C IS MARKS THE NEXT SEGMENT LINK OF THIS BLOCK
C SWITCH THIS SEGMENT BACK TO THE ONE FOLLOWING II
II = II + 1
IDUM2 = ITRITE(II)
DUM3 = XL(II)
DUM4 = YL(II)
DUM5 = XR(II)
DUM6 = YR(II)
ITRITE(II) = ITRITE(IS)
XR(II) = XR(IS)
YR(II) = YR(IS)
ITRITE(IS) = IDUM2
XL(IS) = DUM3
YL(IS) = DUM4
XR(IS) = DUM5
YR(IS) = DUM6
IS = IS + 1
GO TO 393
392 IF(I1 .EQ. IEND1) GO TO 394
C BLOCK DID NOT CHAIN IF YOU REACHED HERE
395 WRITE(6,241) NUMBLK
GO 396 I2 = IBEGF, IENDF
396 WRITE(6,240) ILEFT(12), ITRITE(12), XL(12), YL(12), XR(12), YR(12)
GO TO 380
394 IF(ABS(XR(IEND1)-XL(IBEGF)) .GT. XOTOL) GO TO 395
IF(ABS(YR(IEND1)-YL(IBEGF)) .GT. YOTOL) GO TO 395
C IF YOU REACHED HERE, BLOCK CHAINED OK
C NOW WE'LL AVERAGE THE DOUBLY DIGITIZED NODES
C SINCE THE DIGITIZATION AGREES WITHIN TOLERANCE LIMITS
C
C***********************************************************************
C* PHASE III
C* SCALE AND ROTATE DIGITIZED POINTS, MIN-MAX & BOUNDARIES
C* CODED WITHIN THE FRAMEWORK OF THE GEO-REF SYSTEM - PHASE I
C***********************************************************************
C
MAPNO = NUMBLK/10000
MAPOL = NUMBLK - (MAPNO*10000)
LUCODE = MAPOL/10
NPOLYD = I10
NYTYP = 10
XM(1) = (XL(IBEGF)+XR(IENDF))/2.
YH(I) = (YL(IBEGF)+YR(IEND1))/2.
NP = NSFG1+1
XH(NP) = XH(I)
YH(NP) = YH(I)
DO 300 I1 = 2,NSEG1
IMARK1 = IBEGF+I1-2
IMARK2 = IBEGF+I1-1
XH(I1) = (XR(IMARK1)+XL(IMARK2))/2.
300 YH(I1) = (YR(IMARK1)+YL(IMARK2))/2.
316 CONTINUE
XMIN = XM(1)
YMIN = YM(1)
XMAX = XM(1)
YMAX = YM(1)
DO 302 I1 = 2,NP
IF(XMIN LE XH(I1)) GO TO 302
XMIN = XH(I1)
302 CONTINUE
DO 304 I1 = 2,NP
IF(YMIN LE YH(I1)) GO TO 304
YMIN = YH(I1)
304 CONTINUE
DO 306 I1 = 2,NP
IF(XMAX GE XH(I1)) GO TO 306
XMAX = XH(I1)
306 CONTINUE
DO 308 I1 = 2,NP
IF(YMAX GE YH(I1)) GO TO 308
YMAX = YH(I1)
XX = (NP-1)/6
NP1 = 1 +XX
J3E = NP-6*(NP1-1)
NPE = (NP1+1)/2
NPER = (2*NPE)-NP1
WRITE(6,14)
WRITE(6,221)
WRITE(6,219)MAPNO,MAPOL,LUCODE,NPOLY,NPTYP
WRITE(7,19)MAPNO,MAPOL,LUCODE,NPOLY,NPTYP
WRITE(6,222)
DO 75 I = 1,4
IF(I.EQ.4) GO TO 79
IF(I.EQ.3) GO TO 78
IF(I.EQ.2) GO TO 77
DUM1 = XMIN
DUM2 = YMIN
GO TO 76
77 DUM1 = XMAX
DUM2 = YMIN
GO TO 76
78 DUM1 = XMIN
DUM2 = YMAX
GO TO 76
79 DUM1 = XMAX
DUM2 = YMAX
76 C(I) = U(1) + A(1,1)*DUM1 + A(1,2)*DUM2
75 B(I) = U(2) + A(2,1)*DUM1 + A(2,2)*DUM2
XMIN = C(I)
DO 80 I = 2,4
IF(XMIN LE C(I)) GO TO 80
XMIN = C(I)
80 CONTINUE
A-V-8
03570 80 CONTINUE
03580 XMAX = C(I)
03590 DO 82 I = 2,4
03600 IF(XMAX.GE.C(I)) GO TO 82
03610 XMAX = C(I)
03620 82 CONTINUE
03630 YMIN = B(I)
03640 DO 84 I = 2,4
03650 IF(YMIN.LE.B(I)) GO TO 84
03660 YMIN = B(I)
03670 84 CONTINUE
03680 YMAX = B(I)
03690 DO 86 I = 2,4
03700 IF(YMAX.GE.B(I)) GO TO 86
03710 YMAX = B(I)
03720 86 CONTINUE
03730 IXMIN = INT(XMIN*10)
03740 IYMIN = INT(YMIN*10)
03750 IXMAX = INT(XMAX*10)
03760 IYMAX = INT(YMAX*10)
03770 WRITE(6,228) IXMIN,IYMIN,IXMAX,IYMAX,NP
03780 WRITE(7,228) IXMIN,IYMIN,IXMAX,IYMAX,NP
03790 WRITE(6,223)
03800 WRITE(6,224)
03810 WRITE(6,232)
03820 DO 55 ISM = 1,NPE
03830 WRITE(6,233)
03840 IF(ISM.NE.NPE) GO TO 61
03850 IF1 = (NPE-1)*2+6
03860 IF(NPER.EQ.1) GO TO 63
03870 DO 310 I2 = 1,6
03880 XD(I2) = XH(IF1+I2)
03890 YD(I2) = YH(IF1+I2)
03900 IBEG = 7
03910 IEND = 6 + I3E
03920 GO TO 64
03930 63 IBEG = 1
03940 IEND = I3E
03950 64 DO 312 I2 = IBEG,EEND
03960 XD(I2) = XH(IF1+I2)
03970 312 YD(I2) = YH(IF1+I2)
03980 GO TO 67
03990 61 CONTINUE
04000 DO 314 I2 = 1,12
04010 IF1 = ((SM-1)*2+6
04020 XD(I2) = XH(IF1+I2)
04030 YD(I2) = YH(IF1+I2)
04040 314 CONTINUE
04050 IF(ISM.EQ.NPE) GO TO 68
04060 NPP = 12
04070 68 NPP = 6*(1-NPER) + I3E
04080 CONTINUE
04090 DO 70 I = 1,NPP
04100 X(I) = U(1) + A(1,1)*XD(I) + A(1,2)*YD(I)
04110 70 Y(I) = U(2) + A(2,1)*XD(I) + A(2,2)*YD(I)
04120 IF(ISM.EQ.NPE) GO TO 81
04130 DO 140 IL=1,2
04140 LSUB = 6*(IL-1)
04150 140 WRITE(6,213) (XD(LSUB+I2),YD(LSUB+I2),I2=1,6)
04160 GO TO 72
04170
81 IF(NPER.EQ.1) GO TO 141
    WRITE(6,213) (XD(I2),YD(I2), I2=1,6)
    IBEG = 7
    IEND = 6+I3E
    GO TO 142
141 IBEG = 1
    IEND = I3E
142 WRITE(6,213) (XD(I2),YD(I2), I2=IBEG,IEND)
72 IF(ISM.EQ.NPE) GO TO 74
    DO 56 I2 = 1+12
        IX(I2) = X(I2)*10
    DO 143 IL=1,3
        IY(12) = Y(12)*10
        LSUB = 4*(IL-1)
        WRITE(6,235) (IX(LSUB+I2),IY(LSUB+I2), I2=1,4)
143 WRITE(7,235) (IX(LSUB+I2),IY(LSUB+I2), I2=1,4)
    GO TO 55
74 XX = (NPP-1)/4
    I4E = NPP - 4*(NP2-1)
    DO 57 I2=1,NPP
        IX(I2) = X(I2)*10
        IY(I2) = Y(I2)*10
        LSUB = 4*(IL-1)
        IF(IL.EQ.NP2) GO TO 147
        WRITE(6,235) (IX(LSUB+I2),IY(LSUB+I2), I2=1,4)
        WRITE(7,235) (IX(LSUB+I2),IY(LSUB+I2), I2=1,4)
    GO TO 145
147 WRITE(6,235) (IX(LSUB+I2),IY(LSUB+I2), I2=1,14E)
    WRITE(7,235) (IX(LSUB+I2),IY(LSUB+I2), I2=1,14E)
145 CONTINUE
55 CONTINUE
    IF(NCULS.EQ.0) GO TO 380
    NPTYP = 0
    WRITE(6,243)
    NP = NCULS*2
    NPM1 = NP-1
    NCULS = 0
    DO 320 I1 = 1,NPM1,2
        IMARK = (I1+2)/2
        XH(I1) = XL(IEND1 + IMARK)
        YH(I1) = YL(IEND1 + IMARK)
        XH(I1+1) = XR(IEND1 + IMARK)
        YH(I1+1) = YR(IEND1 + IMARK)
    GO TO 316
320 CONTINUE
380 CONTINUE
    IF(K.EQ.0) GO TO 98
    WRITE(6,225) XTOL,IBIG,(KI(I), I = 1,K)
    WRITE(6,230)
    DO 87 I = 1,K
        IWR = 2*K(I)-1
    87 WRITE(6,231) (IWR)
    IF(LE.EQ.0) GO TO 99
    WRITE(6,226) YOTOL,IBIG,(LI(I), I = 1,L)
    WRITE(6,230)
    DO 88 I = 1,L
        IWR = 2*L(I)
    88 WRITE(6,231) (IWR)
    IF(ABS(THETA(I1)-THETA(2))*.60.EL.THETOL) GO TO 100
99 GO TO 100
A-V-10
WRITE(6,227) THEOLT,IBIG
100 CONTINUE
STOP
END
SUBROUTINE SORT6(ITL,ITR,XL,YL,XR,YR,N)
DIMENSION XL(1000),YL(1000),XR(1000),YR(1000),ITL(1000)
ITR(1000)
DO 20 I = 1,N
IF(ITL(I) .LE. ITL(I+1)) GO TO 20
K = I + 2
K = K-1
IF(K .LT. 2) GO TO 20
IF(ITL(K-1) .LE. ITL(K)) GO TO 20
IDUM1 = ITL(K)
IDUM2 = ITR(K)
DUM3 = XL(K)
DUM4 = YL(K)
DUM5 = XR(K)
DUM6 = YR(K)
ITL(K) = ITL(K-1)
ITR(K) = ITR(K-1)
XL(K) = XL(K-1)
YL(K) = YL(K-1)
XR(K) = XR(K-1)
YR(K) = YR(K-1)
ITL(K-1) = IDUM1
ITR(K-1) = IDUM2
XL(K-1) = DUM3
YL(K-1) = DUM4
XR(K-1) = DUM5
YR(K-1) = DUM6
GO TO 10
20 CONTINUE
RETURN
END
SUBROUTINE RATMUL(C,A,B,I,J,K)
DIMENSION C(I,K),A(1,J),B(J,K)
IR = 1
JC = 1
C(IR,JC) = 0
DO 100 I = 1,J
C(IR,JC) = C(IR,JC) + A(IR,I)*B(I,JC)
100 CONTINUE
RETURN
END
SUBROUTINE INVERS(ATA,N)
DIMENSION ATAN(N,N),A(4,4)
302 FORMAT(X, ' NO SOLUTION, A 0 EXISTS ON THE MAIN DIAGONAL FOR')
303 FORMAT(' ROW NUMBER ',I3,' COLUMN NUMBER ',I3)
301 FORMAT(' REMEMBER-ROWS MAY HAVE BEEN SWITCHED PREVIOUS TO THIS ')
304 FORMAT(' POINT, NO MORE SWITCHING POSSIBLE')
DD = 30001 = 1,N
DD = 306 JJ = 1,N
DD = 306 A(I,J) = 0.0
DD = 300 A(I,I) = 1.0
DET = 1.0
DD = 341 IH = 1,N
IG = IH
IF (ABS(ATAM(N,IH)) .GE. 1.D-10) GO TO 332
C SWAP ROWS
305 IG = IG+1
A-V-11
IF (IG+LE+N) GO TO 308
WRITE(6,302) IH, IH
WRITE(6,303) IH, IH
WRITE(6,301)
WRITE(6,304)
GO TO 381
308 IF (ABS(ATA(IG,IG))<LT.1.E-10) GO TO 305
DET = (-1.0)*DET
DD 324JJJ = I,N
C = ATA(IH,JJ)
D = A(IH,JJ)
ATA(IH,JJ) = ATA(IG,JJ)
A(IH,JJ) = A(IG,JJ)
ATA(IG,JJ) = C
324 A(IG,JJ) = D
332 DIAG = ATA(IH,IH)
DET = DET*DIAG
DD 334JJJ = I,N
ATA(IH,JJ) = ATA(IH,JJ)/DIAG
334 A(IH,JJ) = A(IH,JJ)/DIAG
IF(IH+EQ. N) GO TO 336
II = IH+1
DO 340IJ = II, N
DIAG = ATA(I,IH)
DO 340JJ = I,N
ATA(I,JJ) = ATA(I,JJ) - (DIAG*ATA(IH,JJ))
340 A(I,JJ) = A(I,JJ) - (DIAG*A(IH,JJ))
341 CONTINUE
336 INI = N-1
DD 348 I = 1,N
IH = N+1-I
II = IH - 1
352 DIAG = ATA(I,IH)
IG = N
346 IF(IG.NEQ. 0) GO TO 342
ATA(I,I) = ATA(I,I) - (DIAG*ATA(IH,I))
A(I,I) = A(I,I) - (DIAG*A(IH,I))
IG = IG - 1
GO TO 346
342 IF(II.NEQ. 1) GO TO 348
II = II-1
GO TO 352
348 CONTINUE
354 DO 380 I = 1,N
DO 380 J = 1,N
380 ATA(I,J) = A(I,J)
381 RETURN
APPENDIX VI

DACS PROGRAM LISTING
IMPLICIT INTEGER (A-Z)
INTEGRER2 NREC(22)
REAL XREC, XFIL
DIMENSION REC0(11), SLIST(2000), IREC(801), RECS(11)
DIMENSION TITLE(3)
EQUIVALENCE (IREC(I), XREC)
COMMON NCODE, COUNT, RECD, SLIST
DATA NINES, INFIL, OUTFIL/99999999, 8, 9/
DATA BLOC, TRAC, REGN, BLANK, SLCT, RNCB, BKRM
x / 'BLKS', 'TRCS', 'RNGS', 'SELE', 'REJE', 'BINA' /
DATA ACRE, SOFT, SOMI
x / 'ACRE', 'SOFT', 'SOMI' /
DATA BGPS / 'BGPS'/
SLIST(1) = NINES
COUNT = 0
READ( 5, 20 ) FILE, XFIL, CALC, CODE, ELECT, AREA, OPTION, TITLE
20 FORMAT(A4, A6, A4, 4X, A4, 4X, A4, 15X, A4, 9X, I4, 6X, 3A4 )
IF( CALC •EQ. BLANK ) CALC = TRAC
IF( AREA •EQ. BLANK) AREA = SOFT
WRITE(6, 21) FILE, XFIL, TITLE, AREA, OPTION
21 FORMAT( '1 INPUT FILE IN ', A4, A6, ' FORMAT' )
X + TITLE = ' ', 3A4/ ' AREA IN ', A4/ ' OPTION=', 14 // )
IF( CALC •NE. BLOC ) GO TO 1
LCODE = 6
WRITE(6, 1000)
GO TO 5
1 IF( CALC •NE. TRAC ) GO TO 2
LCODE = 10
WRITE(6, 1010)
GO TO 5
2 IF( CALC •NE. REGN ) GO TO 3
LCODE = 8
WRITE(6, 1020)
GO TO 5
3 IF( CALC •NE. BGPS ) GO TO 4
LCODE = 6
WRITE(6, 1025)
GO TO 5
4 WRITE(6, 1030) CALC, CODE, ELECT, AREA, OPTION, TITLE
RECD = 0
RECONT = 1
GO TO 800
C 1000 FORMAT( '0 AREAS AND CENTROIDS OF CENSUS BLOCKS REQUESTED' )
1010 FORMAT( '0 AREAS AND CENTROIDS OF CENSUS TRACTS REQUESTED' )
1020 FORMAT( '0 AREAS AND CENTROIDS OF AREA CODES REQUESTED' )
1025 FORMAT( '0 AREAS AND CENTROIDS OF BLOCK GROUPS REQUESTED' )
C 1030 FORMAT( '1 UNUSABLE CONTROL CARD SPECIFICATION' )
X + CALCULATE ', A4, 4X, A4, ' CT ', A4, 2X, A4, 13, 3A4 )
5 RCODE = LCODE + 1
   IF( CODE .EQ. BLANK) GO TO 9
   SELECT = 3
   IF( CODE .EQ. SLCT ) SELECT = 100
   IF( CODE .EQ. RJC ) SELECT = 0
   IF( SELECT .EQ. 3 ) GO TO 4
8 ECODE = 100
   IF( ELECT .EQ. BLOC ) ECODE = 7
   IF( ELECT .EQ. BGPS ) ECODE = 7
   IF( ELECT .EQ. TRAC ) ECODE = 11
   IF( ELECT .EQ. REGN ) ECODE = 9
   IF( ECODE .GT. 11 ) GO TO 4
   AMULT = 0
   IF( AREA .EQ. ACRES) AMULT = 640
   IF( AREA .EQ. SQFT ) AMULT = 5280 * 5280
   IF( AREA .EQ. SOMI ) AMULT = 1
   IF( AMULT .EQ. 0 ) GO TO 4
   WRITE( OUTFIL, 1090 ) AMULT, AREA, CALC, OPTION, TITLE
   1090 FORMAT( 18X, 111, 2A4, I4, 3A4 )
   IF( CODE .EQ. BLANK) GO TO 80
   READ( 5, 22, END=80 ) TRACT
   22 FORMAT( 19 )
   IF( TRACT .GE. NINES ) GO TO 80
   IF( ELECT.EQ.BLOC) TRACT = TRACT*1000 + BLOCK
   IF(COUNT) 309 30, 40
   30 SLIST(I)=TRACT
     COUNT=1
     GO TO 10
   40 DO 50 I=1, COUNT
     SLIST(I) = TRACT ) 50, 10, 60
   50 CONTINUE
     COUNT=COUNT + 1
     SLIST(COUNT)=TRACT
     GO TO 10
   60 COUNT=COUNT + 1
     J=COUNT
   70 K=J - 1
     SLIST(J)=SLIST(K)
     J=J - 1
     IF(J .GT. 11) GO TO 70
     SLIST(I)=TRACT
     GO TO 10
   80 IF( COUNT .EQ. 0 ) WRITE( 6,1040)
   C
   IF( COUNT .GT. 0 ) WRITE( 6,1050) COUNT, ELECT, CODE
     ,(SLIST(N),N=1,COUNT)
   1040 FORMAT( 12I0 )
   1050 FORMAT( 12I0 )
   500 RECIN=0
   RECOUT=0
READ DIME FILE RECORD AND SELECT REQUIRED INFORMATION

IF(FILE.EQ.BNRY) GO TO 555
READ(INFILE,560,END=800)RECD(1),RECD(3),RECD(5),LINK,RECD(10),
XRECD(7),FX,FY,TX,TY
560 FORMAT(28X,A1,16X,2(I3.2X),29X,I7,2I6,19X,4I4,2(13X,I5,1X),
X54X,A17)
GO TO 558
555 READ(INFILE,END=800) IREC
558 CONTINUE

NONE STREET CODE
RECD(1)=IREC(22)
FROM NODE NUMBER AND MAP NUMBER
RECD(2)=IREC(3)
RECD(3)=IREC(4)
TO NODE NUMBER AND MAP NUMBER
RECD(4)=IREC(6)
RECD(5)=IREC(7)
BLOCK NUMBER. LEFT AND RIGHT
RECD(6)=IREC(30)
RECD(7)=IREC(31)
PLACE CODE, LEFT AND RIGHT
RECD(8)=IREC(44)
RECD(9)=IREC(45)
TRACT NUMBER. LEFT AND RIGHT
RECD(10)=IREC(32)
RECD(11)=IREC(33)
STATE PLANES COORDINATES FOR FROM AND TO NODES
FX=XREC(70)
FY=XREC(71)
TX=XREC(73)
TY=XREC(74)
SAVE RECORD GEO-CODES
DO 90 J=1,11
RECS(J)=RECD(J)
90 CONTINUE
SEQUENCE NUMBER
LINK=IREC(13)
RECS=RECS + 1
MAKE BLOCK NUMBERS UNIQUE BY COMBINING WITH TRACT NUMBERS.
IF( LCODE .NE. 6 ) GO TO 5651
DO 565 K = 6,7
J = K+4
RECD(K) = MOD(RECD(K),10) + RECD(K)/100*10
RECD(J) = MOD(RECD(J),10) + RECD(J)/100*10
RECD(K) = RECD(K) + RECD(J)*10000
IF( CALC .EQ. BGPS ) RECD(K) = RECD(K) / 100
565 CONTINUE

A-VI-4
SELECT ONLY BOUNDARY SEGMENTS.

5651 KEYL = RECD(LCODE)
KEYR = RECD(RCODE)
IF (KEYL.EQ.KEYR) GO TO 550

ENCOD NODE AND MAP NUMBERS, ALL RESULTING #S UNIQUE IN TRACT
DO 566 = 2, 42
RECD(I) = RECD(I) + 10000 * RECD(I + 1)
CONTINUE

IF (COUNT.EQ.0) GO TO 575
IF (ISKIP(RECS(ECODE-1), SELECT).EQ.1) GO TO 550

KEYR IS REGION TO LEFT OF SEGMENT WHEN DIRECTION IS REVERSED, TX, TY
FROM NODE COORDINATES.

575 WRITE (OUTFIL, 570) KEYR, KEYL, LINK, RECD(4), RECD(2), TX, TY
RECD(5) = RECD(5) + 1
580 IF (COUNT.EQ.0) GO TO 581
5801 IF (ISKIP(RECS(ECODE), SELECT).EQ.1) GO TO 550

KEYL IS COUNTERPART SEGMENT OF KEYR, TO DESCRIBE BOUNDARY OF ADJOINING REGION.

581 WRITE (OUTFIL, 570) KEYL, KEYR, LINK, RECD(2), RECD(4), FX, FY
570 FORMAT(3I9, 4I10)
RECD(5) = RECD(5) + 1
GO TO 550
800 WRITE (OUTFIL, 810) NINES
810 FORMAT(19)
END FILE OUTFIL
850 FORMAT('PRESSORT PROCESSING COMPLETED' // 10X, 'RECORDS IND')
X ' // 10X, 16, ' RECORDS PASSED TO SORT')

FUNCTION ISKIP(KEY, SELECT)
IMPLICIT INTEGER(A-Z)
COMMON NCODE, COUNT, RECD, SLIST
DIMENSION RECD(11), SLIST(2000)
ISKIP = 0
TOP = 1
BOTTOM = COUNT
TRACT = KEY
10 IF (BOTTOM .LT. TOP) GO TO 301
MIDPT = TOP + (BOTTOM - TOP) / 2
IF (TRACT .EQ. SLIST(MIDPT)) GO TO 15
15 BOTTOM = MIDPT - 1
GO TO 10
20 TOP = MIDPT + 1
GO TO 10
100 ISKIP = 1  
IF( SELECT *EO= 100) ISKIP=0  
RETURN
301 IF( SELECT *EO= 100) ISKIP = 1  
RETURN
END
IMPLICIT INTEGER (B-Y)
INTEGER*2 ID(4)
DIMENSION NO(2), NS(3)
EQUIVALENCES (NO(1), ND(1))
DOUBLE PRECISION ZMULT
DIMENSION TITLE(3), OPTION(3)
DIMENSION TLIST(2000)
COMMON ELIST(2000), SDATA DACCO0100
DATA R, INFILE, OUT1, OUT2, EOF, MAX, AREA, ATOTAL, ASUBDACCO0110
X / 1, 9, 10, 11, 99999999992000, 0.0, 0.0, 0.0 DAC CO0120
LOGICAL PIGS / .FALSE. / DACCO0130
LOGICAL ADJ / .FALSE. / DACCO0140
LOGICAL LIST / .FALSE. / DACCO0150
DATA NUM / 0 / DACCO0160
DATA SGRP / 'B6PS' DACCO0170
DATA BLANK / ' ' DACCO0180
C OPTION PROCESSING DACCO0190
CNT=0 DACCO0200
READ(5,5001,ERR=999) AMULT, LNITS, KALC, OPTION, TITLE
5001 FORMAT(F10.0,2A4,311,3A4) DACCO0210
ZMULT = AMULT * 0.0000001 DACCO0220
IF (OPTION(1) .EQ. 1) LIST = .TRUE. DACCO0230
IF (OPTION(2) .EQ. 1) BNDS = .TRUE. DACCO0240
IF (OPTION(3) .EQ. 1) ADJ = .TRUE. DACCO0250
READ( INFILE, 5002, END=500) KEY, TLIST(J), (ELIST(I,J), J=1,5) DAC CO0260
5002 FORMAT(7I9) DACCO0270
WRITE(6,5003) KALC, KEY, TITLE
5003 FORMAT( // *** 'A4, I11, ' HAS', I5, ' CR MORE SEGMENTS' )
MAX = R + 1 DACCO0280
IF( R .LT. MAX ) GO TO 120 DACCO0290
WRITE(6,5003) KALC, KEY, MAX DACCO0300
105 READ( INFILE, 5002, END=500) KEY, TLIST(I,J), (ELIST(I,J), J=1,5) DAC CO0310
105 READ( INFILE, 5002, END=500) KEY, TLIST(I,J), (ELIST(I,J), J=1,5)
NIX = I DACCO0320
NIX = I DACCO0330
IF( KEYX .EQ. KEY ) GO TO 105 DACCO0340
NIX = I DACCO0350
WRITE(6,5004) NIX DACCO0360
5004 FORMAT(24X, I5, ' SEGMENTS CROPPED. NO CALCULATIONS' )
NIX = 0 DACCO0370
R = 2 DACCO0380
120 READ( INFILE, 5002, END=500) KEY, TLIST(R), (ELIST(R,J), J=1,5) DAC CO0390
120 READ( INFILE, 5002, END=500) KEY, TLIST(R), (ELIST(R,J), J=1,5)
C IF( KEYX .EQ. KEY ) GO TO 10C DACCO0400
C $ * DACCO0410
C $ PROCESSING * DACCO0420
C $ * DACCO0430
C 130 NUM = NUM + 1 DACCO0440
C NBLSS = R-1 DACCO0450
AREA = 0.0 DACCO0460
A-VI-7
CENTX = 0
CENTY = 0
IND = 0
CALL CHAIN ( NBLS, COMPS, CLOSES, RVSLS )
IF( COMPS .NE. CLOSES ) GO TO 140
IF( COMPS .EQ. 1 ) GO TO 150
C
WRITE 6, 5005) KALC, KEY, COMPS
C
5005 FORMAT (*0** ,*A4,* I11,* *H A*S*,* I3,* *BOUNDED REGIONS - CHECK* ) DACCO0710
GO TO 150
C
C
140 WRITE 6, 5006) KALC, KEY, COMPS, CLOSES
C
5006 FORMAT (*0** ,*A4,* I11,* *H A*S*,* I3,* *COMPONENTS, ONLY* ,*I3,* ARDAC 00750
X CLOSED, REGION NOT PROCESEED* ) DACCO0760
C
WRITE 6, 5007 )
DO 145 I = 1,NBLS
NO(I) = ELIST(I,2)
NO(2) = ELIST(I,3)
NS(I) = ELIST(I,4)
NS(2) = ELIST(I,5)
145 WRITE(6,5097) NO, NS
C
5097 FORMAT(2(5X,A2,15)3112) DACCO0860
C
C
150 CALL CALC( NBLS, AREA, CENTX, CENTY, MX, MY )
C
IF( AREA .LE. 0 ) DROPT = DROPT + 1
C
GO TO 1000
C
C
150 CALL CALC( NBLS, AREA, CENTX, CENTY, MX, MY )
C
IF( CENTX .NE. 0 ) CALL POLYPT( NBLS, CENTX, CENTY, IND, MX, MY)
IF(.NOT. ENDS ) GO TO 1000
C
SYMAP=IA
KEY1=KEY/1000
KEY2=MOD(KEY, 1000)
DO 180 I = 1, NBLS
KEY2=KEY2+1
WRITE(OUT2,5011) KEY1, SYMAP, ELIST(I,5), ELIST(I,4), ELIST(I,1), XKEY2
C
5011 FORMAT(15**4X,A1,2110,20X,110,13X,17)
SYMAP=BLANK
DO 180 CONTINUE
C
KEY2=KEY2+1
WRITE(OUT2,5011) KEY1, SYMAP, ELIST(I,5), ELIST(I,4), ELIST(I,1), XKEY2
C
1000 IF( LIST ) WRITE(6,5008) KALC, KEY, NBLS, AREA, UNITS, CENTX, CENTYDACCO1180
5008 FORMAT(//5X,A4,I11,19, * SEGMENTS, AREA = ' , F15.5,1X, A4 ) DACCO1190
X / 29X , * CENTROID IS :*2115 ) DACCO1200
C
IF (PIOS) CALL REFMT (NBLS,KEY)
C
A-VI-8
IF(RVSLS .GT. 0) WRITE(6,5015) RVSLS

5015 FORMAT(9X, 111, ' REVERSALS - CHECK FOR POSS. ERRORS.' )

IF( IND .GE. 0 ) GO TO 1100

C CALL ADJUST (NBLS, CENTX, CENTY, NODE )

C WRITE(6,5009) NODE, CENTX, CENTY

5009 FORMAT( / 9X, 'CENTROID WAS CLSIDE BOUNDARY - ADJUSTED TO NODE' )

X 111 /

C

C

5010 FORMAT( 110, F20.5, 210 )

C ASUB = ASUE + AREA

C ATOTAL = ATOTAL + AREA

C 1150 IF( ADJ ) CALL ADJNCY (NELS, KEY, TLIST )

1200 ELIST(1,1) = ELIST(R,T)

C 1250 KEY = KEYX

Q = 2

C 1300 IF( KEY .LT. ECF ) GO TO 120

C IF( TDRT .EQ. 0 ) TDRT = DREPT

WRITE(6,5013) NUM, KALC, TDRT, ATOTAL, UNITS

5013 FORMAT( '19 /1X, 10('----') // 110, 1X, A4, ' ', ' ', 16, ' OMITTED' )

X // 5X, 'TOTAL AREA IS' + E13.5, 1X, A4 // 1X, 10('----' )

GO TO 1350

500 KEYX= EOF

GO TO 1250

C 9999 WRITE(6,5099)

C 5099 FORMAT( '1 ERROR IN FIRST RECORD' )

WRITE(6,5001), AMULT, UNITS, KALC, OPTION, TITLE

1350 STOP

END

SUBROUTINE CHAIN (NBLS, COMPS, CLOSES, RVSLS)

C THIS SUBROUTINE IS IDENTICAL TO THAT USED IN THE DIME EDIT PACKAG

IMPLICIT INTEGER (A-Z)

DOUBLE PRECISION ZMULT

DIMENSION HOLDER(5), ELIST(20CC, 5)

COMMON ELIST, ZMULT

BEGIN = 1

END = 1

COMPS = 0

RVSLS = 0

CLOSES = 0

HEAD = ELIST(BEGIN,2)

TAIL = ELIST(END,3)

IF( NBLS - 1 ) 1250, 1200, 1000
1000 IF( HEAD .EQ. TAIL ) GO TO 1200
START = ENC + 1
IF( START .GT. NELS ) GO TO 1200
DO 1100 I = START, NELS
IF (ELIST(I,3),EQ,HEAD) GO TO 3000
IF (ELIST(I,2),EQ,TAIL) GO TO 2000
1100 CONTINUE
DO 1150 I = START, NELS
II = I + 1
IF (II.EQ.NELS) II = 1
IF (ELIST(II,2),EQ,HEAD) GO TO 2990
IF (ELIST(II,3),EQ,TAIL) GO TO 1990
1150 CONTINUE
1200 CCMPS = CCMPS + 1
IF (II.EQ.TAIL) CLOSES = CLOSES + 1
1250 IF (END .GE. NELS ) RETURN
CHAIN **
ELIST(ENC,1) = -ELIST(END+1)
IF (CLOSES.GE.2) GO TO 1400
C
C THE FOLLOWING SECTION REARRANGES THE ORDER OF SEGMENTS IN THE CHAIN
C PERMIT CHAINING AS ONE COMPONENT FOR FIGURE EIGHT OR CHECKERBOARD.
C
C ARE TYPES OF REGIONS WHICH CONTAIN ONLY ONE COMPONENT BUT WHICH THE
C PROGRAM CAN INTERPRET AS HAVING TWO OR THREE CLOSED COMPONENTS.
C
CLOSES = 0
CCMPS = 0
J1 = ENC + 1
END1 = END
DO 1300 K1 = 1,END1
HEAD = ELIST(K1,3)
DO 1300 I = J1,NELS
IF (ELIST(I,2),EQ,ELIST(I +2)) GO TO 2000
1300 CONTINUE
DO 1350 K1 = 1,ENC1
HEAD = ELIST(K1,2)
DO 1350 I = J1,NELS
II = I + 1
IF (II.EQ.NELS) II = 1
IF (ELIST(II,3),EQ,ELIST(II +3)) GO TO 1990
1350 CONTINUE
CLOSES = 1
CCMPS = 1
C
1400 END = ENC + 1
BEGIN = END
HEAD = ELIST(BEGIN+2)
TAIL = ELIST(END+3)
GO TO 1000
C
A-VI-10
1990 TEMP = ELIST(I,2)
ELIST(I,2) = ELIST(I,3)
ELIST(I,3) = TEMP
TEMPY = ELIST(I,5)
TEMPX = ELIST(I,4)
ELIST(I,4) = ELIST(I,4)
ELIST(I,5) = ELIST(I,5)
ELIST(I,6) = ELIST(I,6)
ELIST(I,7) = TEMPY
AVSLS = AVSLS + 1
2000 END = END + 1
IF( END .EQ. I ) GO TO 2050
DO 2010 K=1,5
2010 HOLDER(K) = ELIST(I,K)
TEMP = I
2020 TEMP = TEMP - 1
DO 2025 K=1,5
2025 ELIST(TEMP+K) = ELIST(TEMP,K)
IF( TEMP .GT. END ) GO TO 2020
DO 2030 K=1,5
2030 ELIST(END+K) = HOLDER(K)
2050 TAIL = ELIST(END,3)
GO TO 1000
2990 TEMP = ELIST(I,2)
ELIST(I,2) = ELIST(I,3)
ELIST(I,3) = TEMP
TEMPY = ELIST(I,5)
TEMPX = ELIST(I,4)
ELIST(I,4) = ELIST(I,4)
ELIST(I,5) = ELIST(I,5)
ELIST(I,6) = TEMPX
ELIST(I,7) = TEMPY
AVSLS = AVSLS + 1
3000 END = END + 1
DO 3010 K=1,5
3010 HOLDER(K) = ELIST(END,K)
TEMP = END
3020 TEMP = TEMP - 1
DO 3030 K=1,5
3030 ELIST(TEMP+K) = ELIST(TEMP,K)
IF( TEMP .GT. BEGIN ) GO TO 3020
IF( I .EQ. END ) I = BEGIN
DO 3040 K=1,5
3040 ELIST(BEGIN+K) = ELIST(I,K)
3100 ELIST(I,K) = HOLDER(K)
HEAD = ELIST(BEGIN,2)
GO TO 1000
END
SUBROUTINE  
ADJUST( NBL, Bx, By, NODE )
C
ROUTINE TO ADJUST RAD CENTROIDS
C
TO NEAREST NODE
C
IMPLICIT INTEGER (B-Y)
DOUBLE PRECISION ZMULT
COMMON ELIST( 2000, 5 ), ZMLT
C
K = 0
DO 101 I = 1,NBL
DX = Bx-ELIST(I,4)
DY = By-ELIST(I,5)
A-VI-11
AXY = DX * DX + DY * DY
IF( I .EQ. 1 ) GO TO 100
IF( A .LT. AXY ) GO TO 101
100 A = AXY
ND = 1
101 CONTINUE
IF( ND .LE. C ) GO TO 200
CX = ELIST(NC,4)
CY = ELIST(NC,5)
NODE = ELIST(NC,2)
RETURN
C
200 CX = 0
CY = 0
NODE = 0
RETURN
END

SUBROUTINE CALC( NSEG, AREA, CX, CY, MINX, MINY )
C
IMPLICIT INTEGER ( B-Y )
DIMENSION ELIST(20000,5)
COMMON ELIST, ZMULT
DOUBLE PRECISION ZMULT, ARS(X, A, DX, DY, X1, Y1, X2, Y2)
DIMENSION ARS(10)
BEGIN = 1
NAR = 0
A = 0
OX = 0
DY = 0
C
C USE FIRST CCCRCINATES TO REDUCE ALL OTHERS
C
MINX = ( ELIST(1,4) / 10000 ) * 10000
MINY = ( ELIST(1,5) / 10000 ) * 10000
C
CHECK FOR MULTIPLE BOUNDARIES
5 NAR = NAR + 1
DO 20 I = BEGIN, NSEG
IF( ELIST(I,1) ) 10920920
10 END = I
GO TO 30
20 CONTINUE
END = NSEG
C
C REDUCING CCCRCINATES AND USE OF DOUBLE PRECISION
C MINIMIZES TRUNCATION AND ROLNDING ERRORS.
C
C
30 X1 = ELIST(END,4) - MINX
Y1 = ELIST(END,5) - MINY
DO 100 K = BEGIN, END
X2 = ELIST(K,4) - MINX
Y2 = ELIST(K,5) - MINY
C
A = A + (X2-X1) * (Y1+Y2)
C
X1 = X2
IF Y1 = Y2
C
A-VI-12
100 CONTINUE

ACCLMULATE AREAS IN 'ARS'

ARS(NAR) = -A
A = 0.0000000

COMPLETE CENTROID OF LARGEST AREA ONLY

BEGIN = END + 1
IF( END .LT. NSEG ) GO TO 5
BEGIN = 1
AB = ARS(1)
IAB = 1
A = AR

IF(NAR .EQ. 1 ) GO TO 180
IF (NAR .GT. 10) GO TO 500

DO 150 I = 2, NAR
A = A + ARS(I)
IF( ARS(I) .LE. AB ) GO TO 150
A8 = ARS(I)
IAB = I
150 CONTINUE
180 AREA = A * ZNULT
IF( A .EQ. 0.0 ) GO TO 500
IF(NAR .EQ. 1 ) GO TO 200
K = 1
DO 185 I = 1, NSEG
IF( ELIST(I,1) .GE. 0 ) GO TO 185
K = K + 1
IF( K .EQ. IAB )185,183,184
183 BEGIN = I + 1
GO TO 185
184 END = I
A = AB
GO TO 200
185 CONTINUE
END = NSEG
A = AB
GO TO 200

200 XI = ELIST(END,4) - MINX
Y1 = ELIST(END,5) - MINY

ARSIX = A * 3.0

NOTE THAT XI AND Y1 ARE ALREADY SET

DO 300 K = BEGIN, END
X2 = ELIST(K,4) - MINX
Y2 = ELIST(K,5) - MINY

DX = DX + ( Y2 - Y1 ) * (XI*XI + X1*X2 + X2*X2 ) / ARSIX
GY = GY + ( X2 - X1 ) * (Y1*Y1 + Y1*Y2 + Y2*Y2 ) / ARSIX

X1 = X2
Y1 = Y2

300 CONTINUE

CX = DX + MINX
CY = -GY + MINY

ORIGINAL PAGE IS OF POOR QUALITY
SUBROUTINE REFMT(NBLS,KEY)
IMPLICIT INTEGER (A-Z)
DIMENSION ELIST(2000,5)
COMMON ELIST,KEY1=KEY/10000,KEY2=MOD(KEY,10000)
WRITE(13,100) KEY1,KEY2
100 FORMAT(1X,2I5)
XMIN=ELIST(1,4)
XMAX=ELIST(1,4)
YMIN=ELIST(1,5)
YMAX=ELIST(1,5)
DO 10 J=2,NBLS
IF(XMIN.GT.ELIST(J,4)) XMIN=ELIST(J,4)
IF(XMAX.LT.ELIST(J,4)) XMAX=ELIST(J,4)
IF(YMIN.GT.ELIST(J,5)) YMIN=ELIST(J,5)
IF(YMAX.LT.ELIST(J,5)) YMAX=ELIST(J,5)
10 CONTINUE
WRITE(13,200) XMIN,YMIN,XMAX,YMAX,NBLS
200 FORMAT(4I9915)
M2=0
MK=NBLS
20 CONTINUE
M1=M2+1
M2=M2+4
IF(MK.LT.4) M2=NBLS
WRITE(13,300) (ELIST(M,4),ELIST(M,5),M=M1,M2)
300 FORMAT(4(2T9))
MK=MK-4
IF(MK.GE.0) GO TO 20
RETURN
END

SUBROUTINE INSECT(IXY, IND)
THE ENDPOINTS OF THE LINES ARE TRANSMITTED (INTEGER BINARY) IN
THE ARRAY 'IXY' (FIRST 2 ELEMENTS) (X-Y X-Y X-Y X-Y )
THE VARIABLE 'IND' IS RETURNED: 0 IF NO INTERSECTION
-1 IF LINES ARE COINCIDENT
+1 IF THEY INTERSECT
THE COORDINATES OF INTERSECTION ARE RETURNED IN IXY(9) AND IXY(10)
DIMENSION IXY(10),XY(8),S(2),P(2)
IND = 0
X = 0
Y = 0
RETURN IF NO INTERSECTION POSSIBLE
CROSS PRODUCT CALCULATION
ACX = IXY(1) - IXY(5)
ACY = IXY(2) - IXY(6)
ADX = IXY(1) - IXY(7)
ADY = IXY(2) - IXY(8)

BCX = IXY(3) - IXY(5)
BCY = IXY(4) - IXY(6)
BX = IXY(3) - IXY(7)
BDY = IXY(4) - IXY(8)

A1 = (ACX * ACY - ACV * AOX)
A2 = (BCX * BDY - BCY * BDX)

IF ( A1 * A2 > 0.0 ) RETURN
R = (ACX * BCY - ACV * BCX) * (ADX * BOY - ADY * BOX)
IF ( R > 0.0 ) RETURN

C CHECK COLINEARITY
IF (A1 .EQ. A2) AND (A2 .EQ. 0.0)  RETURN

C CALCULATE INTERSECTION
R = 0.0
IF (A1 .NE. 0.0) R = 1.0/(1.0 + ABS(A2/A1))
X = IXY(1) - (IXY(3) - IXY(1)) * R
Y = IXY(2) - (IXY(4) - IXY(2)) * R

C ROUND TO NEAREST INTEGER
X = INT (X)
Y = INT (Y)

C COLINEAR CHECK FOR OVERLAP (COINCIDENCE)
2220 IF (ACX * BCX < 0.0) IND = -1
IF (ADX * BDX < 0.0) IND = -1
RETURN
END

SUBROUTINE POLYPT (N, NX, NY, IND, MINX, PINY)

DIMENSION LXY(10), ELIST(2000*5)
COMMON ELIST, ZMULT
INTEGER ELIST
DOUBLE PRECISION ZMULT

IX = NX - MINX + 1
IV = NY - PINY + 1
IND = 0
INDX = 0
KOUNT = 100000.

LXY(1) = 0
LXY(2) = IX
LXY(3) = IX
LXY(4) = IV
LXY(9) = 0
LXY(10) = 0
IT = 0
GO TO 2000
C DRAW A LINE FROM THE PCINT TO THE Y-AXIS AND COUNT THE NUMBER OF INTERSECTIONS WITH BOUNDARY SEGMENTS. OUTSIDE
C IF AN INTERSECTION OCCURS AT A NODE
C CHANGE THE Y COORDINATE AT THE AXIS AND START OVER
C
1000 IT = IT + 1
LXY (2) = LXY (2) + IT/10 + 1
IF (IT.GT.5) RETURN
2000 INDEX = 0
DO 3000 I = 1,N
N = I + 1
IF (INDEX.EQ.N) I = 1
LXY (5) = ELIST(I,4) - MINX + 1
LXY (6) = ELIST(I,5) - MINY + 1
LXY (7) = ELIST(I,4) - MINX + 1
LXY (8) = ELIST(I,5) - MINY + 1
CHECK FOR PT AT A NODE
DO 2100 L=1,5,2
IF (LXY(L).EQ.IX.AND.LXY(L+1).EQ.IY) RETURN
2100 CONTINUE
CALL INSECT( LXY, INT )
IF (INT) 1000,3000,2400
CHECK FOR PT ON BOUNDARY
C 2400 IF (LXY(9).EQ.IX.AND.LXY(10).EQ.IY) RETURN
C CHECK FOR INTERSECTION WITH A CORNER
C DO 2500 J=6,8,2
IF (LXY(9) .NE. LXY(J-1) ) GO TO 2500
IF (LXY(10).EQ.LXY(J)) GO TO 1000
2500 CONTINUE
INDEX = INDEX + 1
3000 CONTINUE
INDEX = INDEX/2
3000 CONTINUE
INDEX = INDEX + 1
IF ((INDEX/2) + 1 .EQ. INDEX) INDEX = -1
RETURN
END
SUBROUTINE ADJACENCY (NBLS, KEY, LIST)
C ADJACENCY LIST
C DIMENSION LIST(2000)
DATA NOLT3 / 12 /
NENC = NBLS - 1
DO 1200 I = 1, NENC
IF (LIST(I) .LT. 0 ) GO TO 1200
N = LIST(I)
1200 CONTINUE
UNDUPPLICATE LIST
DO 1100 J = NEXT, NBLS
IF (LIST(J) .EQ. N ) LIST(J) = -1
1100 CONTINUE
PRINT AND COPY LIST
WRITE(6,1300) KEY
1300 FORMAT (*) LIST OF ADJACENT AREAS FOR*, 111/
IF( LIST(I) .LT. 0 ) GO TO 1500
WRITE(6,1400) LIST(I)
1400 FORMAT( 50X, III )
WRITE(NOUT3,1450 ) KEY, LIST(I)
1450 FORMAT( 2I15 )
C
1500 CONTINUE
RETURN
END
APPENDIX VII

PIOS PROGRAM LISTING
**MAIN PROGRAM FOR POLYGON OVERLAY**

**READ TEST POLYGON**

**DEFINE I/O UNITS**

**BE SURE TO EXPAND ALL ARRAYS AND DEFINE FILES TO ACCOMODATE YOUR NEEDS**

**ONLY ONE CARD IMAGE IS READ IN ON UNIT 5, PRTOUT. PRINTING OUT**

**FILE 1 I**

**FILE 2 WILL RESULT IN FILE 13 BEING PRINTED OUT**

**FILE 3 IN COLS 1-6 WILL RESULT IN FILE 14 BEING PRINTED OUT**

**FILE 4 IN COLS 1 WILL RESULT IN FILE 14 BEING PRINTED OUT**

**FILE 5 IN COLS 1 WILL RESULT IN FILE 14 NOT BEING PRINTED OUT**

**READ (5,20)PRTOUT**

**20 FORMAT (15-20)PRTOUT**

**IV1 = 1**

**IV2 = 1**

**IV3 = 1**

**IV4 = 1**

**IPTROUT = *TRUE***

**DONUT = *FALSE***

**OUT = *FALSE***
FIND (14*1) 00620020
FIND (15*1) 00630030
WRITE (6,2700) 00640000
2700 FORMAT (' CLUGE CALLED BY DRIVER/LA ENG ||||*) 00650000
CALL CLUGE 00660000
WRITE (6,2710) 00670000
2710 FORMAT (' RETURNED FROM CLUGE ENG ||||*') 00680000
FIND (11*1) 00690000
FIND (12*1) 00700000
IV1=1 00710000
IV2=1 00720000
C \[\text{CLUGE used only for test case} 00730000
C \[\text{BRINGS IN TEST DATA} 00740000
C \[UNIT=5 00750000
C JUNIT=6 00760000
C INTZ=11 00770000
C IASC=6 00780000
C IZZ=13 00790000
C\]
C SET SWITCH FOR DIAGNOSTIC PRINT OUT 00800000
C ICNT=1 00810000
C DO 50 I=1,1300 00820000
C 30 READ (12*IV2,40,ERR=70) MAPOL1,XLE(I),YLE(I),XHE(I),YHE(I),N09830000
C 1YHE(1),N,(X(K),Y(K),K=1,N) 00840000
C 40 FORMAT (215/4F9.0,15/(8F9.0,1)) 00850000
C IF (N>LE.3000) GO TO 45 00860000
C 42 FORMAT (' MAPNO, N= ',2(IS,2X) 00870000
C WRITE(6,44) MAPOL1,N 00880000
C 44 FORMAT ('----- ERROR-----ERROR-----TOO MANY POINTS*/,/*',IS,2X) 00890000
C 1N= ',16,* PIOS/II TERMINATED*) 00900000
C STOP 00910000
C 45 NMAPOL=1 00920000
C WRITE(6,42) MAPOL1 (1).N 00930000
C MAPNO = MAPOL1 (1) 00940000
C C CALCULATE RECORD COUNT 00950000
C IRE(I)=ICNT 00960000
C ICNT=ICNT+2*(N+3)/4 00970000
C IF (MAPNO.EQ.999999) GO TO 70 00980000
C 50 CONTINUE 00990000
C WRITE (6,60) 00990000
C 60 FORMAT (1X, 'MORE THAN 1300 POLYGONS-- PROCESS FIRST 1300.') 00990000
C 70 CONTINUE 00990000
C DIAG=.FALSE. 00990000
C 80 CONTINUE 00990000
C C SEARCH FOR SPECIFIED TRAFFIC ZONE 01000000
C C C C READ MAJOR POLYGON 01010000
C C CALL POLYP (6930,MAPNO,MAPOL1,LUCODE,LUAREA,XMIN,YMIN,XMAX,YMAX,N01100000
C 1AX,X,Y,INTZ,FAC,DONUT,OUT,IV1) 01110000
C WRITE(15*IV5,86) MAPNO,MAPOL1(1),LUCODE(3),MAPOL1(2),LUAREA 01120000
C WRITE(7,86) MAPNO,MAPOL1(1),LUCODE(3),MAPOL1(2),LUAREA 01130000
C 86 FORMAT (21S,14,12,F15.2) 01140000
C FIND (15*IV5) 01150000
C C \[SET THE SIGNIFICANCE VALUE FOR THE 01160000
C APPROXIMATION FOR THE INTERSECTIONS 01170000
C A-VII-3
C
SIG=.01
C
INITIALIZE THE UPPER AND LOWER
BOUNDS FOR XX AND YY.

YY=0.0
XX=1.0E6
M=MAX-1
C
FIND THE LOWER LEFT VERTEX OF
THE MAJOR POLYGON.
DO 110 J=1,M
IF (Y(J)-YY) 1 10 • l 009 90
90 YY=Y(J)
XX=X(J)
IJ=J
GO TO 110
100 IF (X(J)-XX) 9091109110
110 CONTINUE
C
OUTPUT DIAGNOSTICS IF REQUESTED
C
SET VALUE OF THE STRIPPING
EXponent
NP=2
ISTP=2**NP
STEP=(YMAX-YMIN)/ISTP
BLOW=YMIN-STEP
DO 120 J=1,ISTP
BLOW=BLOW+STEP
BH(J)=BMIN
BL(J)=BLOW
BH(J)=BH(J)+STEP
120 CONTINUE

ISTP(YMAX)=YMAX

C
INITIALIZE LEVEL AND STRING LISTS
C
DO 130 IC=1,16
NSTPL(IC)=0
DO 130 IA=1,20
DO 130 IB=1,2
130 LSTP(IC*IB*IA)=0
ISTR=0
C
START STRING ASSIGNMENTS
C

XLAST=X(IJ)
YLAST=Y(IJ)
NOXI=IJ
LLEV=LEVEL(YLAST,YMIN,STEP,ISTP,YMAX)
ISTR=ISTR+1
XSTR(ISTR)=XLAST
YSTR(ISTR)=YLAST
NSTPL(LLEV)=NSTPL(LLEV)+1
ILS=NSTPL(LLEV)
NF=ISTR
C
LOOP THRU POINTS

A-VII-4
C 0/820000
DO 310 J=2,MAX
NDX=MOD(NDX,M)+1
XCUR=X(NDX)
YCUR=Y(NDX)
LEVC=LEVEL(YCUR,YMIN,STEP,ISTP,YMAX)
IF (LLEV.LE.LEVC) GO TO 160
C
C SAME LEVEL - SAME STRING
C
IF (YCUR.NE.YSTR(ISTRG)) GO TO 140
IF (XCUR.NE.XSTR(ISTRG)) GO TO 140
GO TO 150
140 ISTRG=ISTRG+1
XSTR(ISTRG)=XCUR
YSTR(ISTRG)=YCUR
150 CONTINUE
GO TO 300
C
C DIFFERENT LEVEL
C
160 CONTINUE
C
C DEFINE DIRECTION
C
IDIR=LEVC-LLEV
IF (IDIR.LT.0) GO TO 220
C
C POSITIVE DIRECTION
C CLOSE LAST LEVEL
C
170 YTEMP=BM(LLEV)
XTEMP=(XCUR-XLAST)*(YTEMP-YLAST)/(YCUR-YLAST)+XLAST
IF (YTEMP.NE.YSTR(ISTRG)) GO TO 180
IF (XTEMP.NE.XSTR(ISTRG)) GO TO 180
GO TO 190
180 ISTRG=ISTRG+1
XSTR(ISTRG)=XTEMP
YSTR(ISTRG)=YTEMP
190 CONTINUE
IF (NF.NE.ISTRG) GO TO 200
NSTPL(LLEV)=NSTPL(LLEV)-1
GO TO 210
200 LSTP(LLEV,1,ILS)=NF
LSTP(LLEV,2,ILS)=ISTRG
210 CONTINUE
C
C START NEW LEVEL
C
LLEV=LLEV+1
NSTPL(LLEV)=NSTPL(LLEV)+1
ILS=NSTPL(LLEV)
NF=ISTRG
IF (LLEV.LT.LEVC) GO TO 170
GO TO 270
C
C NEGATIVE DIRECTION
C
CLOSE LAST LEVEL
C
220 YTEMP=BL(LLEV)
XTEMP=(XCUR-XLAST)*(YTEMP-YLAST)/(YCUR-YLAST)+XLAST
IF (YTEMP.NE.YSTR(ISTRG)) GO TO 230
IF (XTEMP.NE.XSTR(ISTRG)) GO TO 230
GO TO 270
C
GO TO 240
230 ISTRG=ISTRG+1
XSTR(ISTRG)=XTEMP
YSTR(ISTRG)=YTEMP
240 CONTINUE
IF (NF.NE.ISTRG) GO TO 250
NSTPL(LLEV)=NSTPL(LLEV)-1
GO TO 260
250 LSTP(LLEV.1,ILS)=NF
LSTP(LLEV.2,ILS)=ISTRG
260 CONTINUE
C
C START NEW LEVEL
C
LLEV=LLEV-1
NSTPL(LLEV)=NSTPL(LLEV)+1
ILS=NSTPL(LLEV)
NF=ISTRG
IF (LLEV.GT.LEVC) GO TO 220
GO TO 270
270 IF (YCUR.NE.YSTR(ISTRG)) GO TO 280
IF (XCUR.NE.XSTR(ISTRG)) GO TO 280
GO TO 290
280 ISTRG=ISTRG+1
XSTR(ISTRG)=XCUR
YSTR(ISTRG)=YCUR
290 CONTINUE
300 XLAST=XCUR
YLAST=YCUR
310 CONTINUE
C
C CLOSE LAST STRING
C
IF (NF.NE.ISTRG) GO TO 320
NSTPL(LLEV)=NSTPL(LLEV)-1
GO TO 330
320 LSTP(LLEV.1,ILS)=NF
LSTP(LLEV.2,ILS)=ISTRG
330 CONTINUE
C
C LOOP ON MINOR POLYGONS
C
C READ MINOR POLYGON
C
IDVER=0
DO 910 IW=1,NMPOLY
C CHECK AND SEE IF EVEN ONE CORNER IS IN THIS MAJOR POLYGON
IF (YHE(IW).LT.YMIN) GO TO 910
IF (YLE(IW).GT.YMAX) GO TO 910
IF (XHE(IW).LT.XMIN) GO TO 910
IF (XLE(IW).GT.XMAX) GO TO 910
C IF GET THIS FAR, THEN HAVE AN OVERLAP.
NN=IRE(IW)
FIND (12*NN)
IV2=NN
CALL POLYRD (6910,SPMAP,SPPOLY,SPCODE,SPAREA,PXMIN,PYMIN,PXMAX,PYMAX,PYM0
IAX,NPTS,PO,INSF,FAC,DONUT,OUT,IV2)
C
C IS MINOR POLYGON OUTSIDE OF TRAFFIC ZONE
340 ALLIN=.TRUE.
ALLOUT=.TRUE.
SPCODE(2)=LUCODE(2)
DO 490 I=1,NPTS
PP=P(I)
OO=0(I)
IF (00-LT.YMIN) GO TO 470
IF (OO.GT.YMAX) GO TO 470
IF (PP.LT.XMIN) GO TO 470
IF (PP.GT.XMAX) GO TO 470
J=LEVEL(QQ,YMIN,STEP,ISTP,YMAX)
JK=NSTPL(J)
FLAG=.FALSE.
IF (JK.EQ.0) GO TO 460
DO 450 JJ=1.JK
YREM=-1.0
JL=LSTP(J,1,JO)
JM=LSTP(J,2,JJ)-1
DO 450 JN=JL.JM
YP=YSTP(JN)
YYP=YSTR(JN+1)
XP=XSTR(JN)
XXP=XSTR(JN+1)
IF (QQ.GT.YP.AND.OO.GT.YYP) GO TO 440
IF (QQ.LT.YP.AND.OO.LT.YYP) GO TO 440
IF (PP.GT.XP.AND.PP.GT.XXP) GO TO 440
IF (QQ.NE.YP) GO TO 370
IF (QQ.EQ.YREM) GO TO 400
YREM=YP
LAST=JN
350 LAST=LAST-1
IF (LAST.EQ.0) GO TO 450
IF (QQ.EQ.YSTR(LAST)) GO TO 350
VLAST=YSTR(LAST)
NEXT=JN
360 NEXT=NEXT+1
IF (NEXT.GT.ISTRG) GO TO 450
IF (QQ.EQ.YSTR(NEXT)) GO TO 360
YNEXT=YSTR(NEXT)
IF (QQ.GT.YLAST.AND.OO.GT.YNEXT) GO TO 450
IF (QQ.LT.YLAST.AND.OO.LT.YNEXT) GO TO 450
IF (PP.GT.XP) GO TO 420
GO TO 450
370 IF (QQ.NE.YYP) GO TO 410
IF (PP.EQ.XXP) GO TO 430
IF (QQ.EQ.YREM) GO TO 400
YREM=YP
LAST=JN+1
380 LAST=LAST-1
IF (QQ.EQ.YSTR(LAST)) GO TO 380
VLAST=YSTR(LAST)
NEXT=JN+1
390 NEXT=NEXT+1
IF (QQ.EQ.YSTR(NEXT)) GO TO 390
YNEXT=YSTR(NEXT)
IF (QQ.GT.YLAST.AND.OO.GT.YNEXT) GO TO 450
IF (QQ.LT.YLAST.AND.OO.LT.YNEXT) GO TO 450
GO TO 360
AN ARRAY IS CONSTRUCTED THAT CONTAINS ALL THE POINTS OF THE INTERSECTING POLYGON(S). THE MINOR POLYGON IS SEARCHED IN A SERIAL MANNER. AS EACH PAIR OF POINTS ARE ENCOUNTERED THE FOLLOWING CASES AND ACTIONS OCCUR:

- **I-1 OUT** POINT I IS NOT ADDED TO THE ARRAY
  - IF (ALLOUT) WRITE (6,500) SPPOLY, SPCODE
  - IF (ALLIN) GO TO 880
  - IF (ALLOUT) GO TO 900

- **I-1 IN**
  - THE MAJOR POLYGON IS SEARCHED TO FIND THE LINE SEGMENT THAT INTERSECTS WITH THE MINOR POLYGON.
  - THIS IS ACCOMPLISHED BY FIRST SEARCHING FOR LINE SEGMENTS THAT INTERSECT WITH THE RECTANGULAR REGION DEFINED BY THE POINTS I-1 AND I. EACH LINE SEGMENT THAT DOES SO IS TESTED FOR INTERSECTION WITH THE LINE FORMED BY I-1 AND I, AT LEAST ONE SUCH INTERSECTION MUST OCCUR (OR THE POINT I WOULD BE OUT). THE POINT OF INTERSECTION IS ADDED TO THE ARRAY. STEPPING IS CONTINUED ON THE MINOR POLYGON.
  - THE POINT I IS ADDED TO THE ARRAY
  - THE MAJOR POLYGON IS SEARCHED FOR THE INTERSECTION.

**A-VII-8**
NOW STEPPING IS PERFORMED ON THE MAJOR POLYGON IN THE DIRECTION SUCH THAT THE POINTS ARE IN THE MINOR POLYGON. THESE POINTS ARE ADDED TO THE ARRAY. WHEN THE STEPPING CAUSES A POINT TO BE OUT OF THE MINOR POLYGON THE INTERSECTION POINT IS FOUND BY SEARCHING THE MINOR POLYGON, AND IS ADDED TO THE ARRAY. STEPPING THEN RESUMES ON THE MINOR POLYGON WHERE IT WAS LAST INTERRUPTED.

WRITE (6,7001)MAPNO,MAPOLI(1),SPMAP,SPCODE(1),SPCODE(3)
7001 FORMAT *(# START BUILD_ MAJOR *15,2X15,2X,MINOR *15,2XA3,2X,14)
NINTS=0
J=0
I=1
IF (PQ(I)) GO TO 560
POINT I IS OUT
510 I=I+1
IF (I.GT.NPTS) GO TO 790
IF (.NOT.PQ(I)) GO TO 510
POINT I-1 WAS OUT, POINT I IS IN
FIND INTERSECTION BY SEARCHING MAJOR POLYGON
K=0
520 K=K+1
IF (K.GT.(ISTRG-1)) GO TO 530
FIND SPAN
Y1=YSTR(K)
Y2=YSTR(K+1)
V1=0(I-1)
V2=0(I)
IF (Y1.LT.V1.AND.Y2.LT.V1.AND.Y1.LT.V2.AND.Y2.LT.V2) GO TO 520
IF (Y1.GT.V1.AND.Y2.GT.V1.AND.Y1.GT.V2.AND.Y2.GT.V2) GO TO 520
X1=XSTR(K)
X2=XSTR(K+1)
U1=P(I-1)
U2=P(I)
IF (X1.LT.U1.AND.X2.LT.U1.AND.X1.LT.U2.AND.X2.LT.U2) GO TO 520
IF (X1.GT.U1.AND.X2.GT.U1.AND.X1.GT.U2.AND.X2.GT.U2) GO TO 520
TEST FOR INTERSECTION
A=X2—X1
B=U1—U2
C=Y2—Y1
D=V1—V2
DET=A*D—C*B
IF (DET.EQ.0) GO TO 520
E=U1—X1
F=V1—Y1
T0=(D*E—B*F)/DET
PO=(A*F—C*E)/DET
IF (T0.LT.0.0.OR.PO.LT.0.0) GO TO 520
IF (T0.GT.1.0.OR.PO.GT.1.0) GO TO 520
XINT=T0*X2+(1.0—T0)*X1
YINT=T0*Y2+(1.0—T0)*Y1
J=J+1
X(J)=XINT
Y(J)=YINT
NINTS=NINTS+1
INTSP(NINTS•1)=J
GO TO 560
530 CONTINUE
IERR=1
WRITE (6,540) IERR
540 FORMAT (' COULD NOT FIND INTERSECTION AT CHECK POINT *',IS)
WRITE (6,550) SPMAP,SPPOLY
550 FORMAT (1X,IS,1X,A3,1X,12)
WRITE (6,560) P(I),Q(I),P(I-1),Q(I-1)
GO TO 910
C POINT I IS IN
560 J=J+1
X(J)=P(I)
Y(J)=Q(I)
IF (I.GT.1) GO TO 570
NINTS=NINTS+1
INTSP(NINTS•1)=J
570 CONTINUE
I=I+1
IF (I.GT.NPTS) GO TO 580
IF (PO(I)) GO TO 560
GO TO 580
580 INTSP(NINTS•2)=J
GO TO 790
590 CONTINUE
C POINT I IS OUT. POINT I-1 WAS IN
C FIND INTERSECTION BY S MAJOR POLYGON
K=0
600 K=K+1
IF (K.GT.(ISTRG-1)) GO TO 610
C FIND SPAN
Y1=YSTR(K)
Y2=YSTR(K+1)
V1=Q(I-1)
V2=Q(I)
IF (Y1.LT.V1 .AND. Y2.LT.V1 .AND. Y1.LT.V2 .AND. Y2.LT.V2) GO TO 600
IF (Y1.GT.V1 .AND. Y2.GT.V1 .AND. Y1.GT.V2 .AND. Y2.GT.V2) GO TO 600
X1=XSTR(K)
X2=XSTR(K+1)
U1=P(I-1)
U2=P(I)
IF (X1.LT.U1 .AND. X2.LT.U1 .AND. X1.LT.U2 .AND. X2.LT.U2) GO TO 600
IF (X1.GT.U1 .AND. X2.GT.U1 .AND. X1.GT.U2 .AND. X2.GT.U2) GO TO 600
C TEST FOR INTERSECTION
A=X2-X1
B=U1-U2
C=Y2-Y1
D=V1-V2
DET=A*D-C*B
IF (DET.EQ.0) GO TO 600
E=U1-X1
F=V1-Y1
TO=(D*E-B*F)/DET
PO=(A*F-C*E)/DET
IF (TO.LT.0.0 OR PO.LT.0.0) GO TO 600
IF (TO.GT.1.0 OR PO.GT.1.0) GO TO 600
XINT=TO*X2+(1.0-TO)*X1
YINT=TO*Y2+(1.0-TO)*Y1
J=J+1
X(J)=XINT
Y(J)=YINT
GO TO 630
610 IERR=2
WRITE (6,540) IERR
WRITE (6,550) SPMAP, SPPOLY
WRITE (6,620) P(I), Q(I), P(I-1), Q(I-1)
620 FORMAT (4F10.0)
GO TO 910
630 CONTINUE
C
C IIIIIIIIIIIIIIII SECTION REMOVED JPL IIIIIIIIIII
C
700 IDIR=1
K=K+1
XE=XSTR(K)
YE=YSTR(K)
710 XB=XINT
YB=YINT
CALL STPSUB (XB, YB, XE, YE, SPX, SPY, NOSTP)
DO 740 L=1, NOSTP
XB=XB+SPX
YB=YB+SPY
CALL PIP (XB, YB, INO, P, Q, NPTS)
730 IF (IND.EQ.0) GO TO 750
740 CONTINUE
C
IF IT GETS HERE THE END POINT IS IN
J=J+1
X(J)=XE
Y(J)=YE
C
TAKE NEXT POINT
K=K+IDIR
IF (K.EQ.0) K=ISTRG-1
IF (K.GT.(ISTRG-1)) K=1
XB=XE
YB=YE
XE=XSTR(K)
YE=YSTR(K)
GO TO 720
750 CONTINUE
C
POINT (XB-SPX, YB-SPY) WAS INSIDE, (XB, YB) IS OUT
760 L=0
770 L=L+1
1F (L.EQ.NPTS) GO TO 780
Y1=Q(L)
Y2=Q(L+1)
V1=YB
V2=YB-SPY
IF (Y1.LT.V1 .AND. Y2.LT.V1 .AND. Y1.LT.V2 .AND. Y2.LT.V2) GO TO 770
IF (Y1.GT.V1 .AND. Y2.GT.V1 .AND. Y1.GT.V2 .AND. Y2.GT.V2) GO TO 770
X1=Q(L)
X2=Q(L+1)
U1=XB
U2=XB-SPX
IF (X1.LT.U1 .AND. X2.LT.U1 .AND. X1.LT.U2 .AND. X2.LT.U2) GO TO 770
IF (X1.GT.U1 .AND. X2.GT.U1 .AND. X1.GT.U2 .AND. X2.GT.U2) GO TO 770
TEST FOR INTERSECTION

A=X2-X1
B=U1-U2
C=Y2-Y1
D=V1-V2
DET=A*D-C*B
IF (DET EQ 0.0) GO TO 770
E=U1-X1
F=V1-Y1
TO=(D*E-B*F)/DET
PO=(A*F-C*E)/DET
IF (TO LT 0.0 OR PO LT 0.0) GO TO 770
IF (TO GT 0.0 OR PO GT 0.0) GO TO 770
XINT=TO*X2+(1.0-TO)*X1
YINT=TO*Y2+(1.0-TO)*Y1
J=J+1
X(J)=XINT
Y(J)=YINT
INTSP(NINTS,2)=J
I=I-1
GO TO 510

780 IERR=3
WRITE (6,540) IERR
WRITE (6,550) SPMAP, SPPOLY
GO TO 910

790 CONTINUE

FIND MINIMUM X AND Y VALUES
TXMIN=1.0E+9
TYMIN=1.0E+9
TXMAX=0.
TYMAX=0.
DO 800 I=1,J
IF (X(I) LT TXMIN) TXMIN=X(I)
IF (Y(I) LT TYMIN) TYMIN=Y(I)
IF (X(I) GT TXMAX) TXMAX=X(I)
IF (Y(I) GT TYMAX) TYMAX=Y(I)
800 CONTINUE

TRANSLATE COORDINATES
DO 810 I=1,J
X(I)=X(I)-TXMIN
Y(I)=Y(I)-TYMIN
810 CONTINUE

AREA=0.0
NOPTS=0
DO 830 I=1,NINTS
J=INTSP(I,1)+1
K=INTSP(I,2)
IF (J GT K) GO TO 830
NOPTS=NOPTS+1
P(NOPTS)=X(J-1)+TXMIN
Q(NOPTS)=Y(J-1)+TYMIN
DO 820 II=1,J,K
NOPTS=NOPTS+1
P(NOPTS)=X(II)+TXMIN
820 CONTINUE
Q(00TS) = Y(II) + TYMIN

820 AREA = AREA + Y(II) * X(II) - Y(II-1) * X(II-1) + Y(II-1) * X(II) - Y(II-1) * X(II-1)

830 CONTINUE

840 AREA = ABS(AREA) / 2.0 / 43560.0 * FAC

C AREA ALGORITHM DOESN'T SEEM TO WORK

C AREA = 0.0

CALL AREAOF (AREA, NOPTS, P, Q, IFLAG)

IF (DONUT) AREA = -AREA

WRITE (6, 7002) MAPNO, MAPOLI(1), SPMAP, SPCODE(1), SPCODE(3)

7002 FORMAT (* ENDоф BUILD - MAJOR *A15, 2X, *MINOR *A15, 2X)

WRITE (14, 1V4, 850) MAPNO, MAPOLI(1), SPMAP, SPCODE(1), SPCODE(3), AREA

1V4X, TYMIN, TXMAX, TYMAX, NOPTS, (P(I), Q(I), I=1, NOPTS)

850 FORMAT (3I5, A3, F15.2)

860 FORMAT (3I5, A3, I4, 2X, F15.2)

861 FORMAT (3I5, A3, I4, 2X)

870 FORMAT (* ENDоф BUILD - MAJOR *A15, 2X, *MINOR *A15, 2X)

IF (IND.EQ.0) GO TO 910

C POLYGON IS ALL OUT

C SINCE ALL THE POLYGON POINTS ARE OUT IT COULD BE THAT THE SOIL
C POLYGON COMPLETELY SURROUNDS THE TRAFFIC ZONE. IF ONE POINT OF THE
C TRAFFIC ZONE IS IN THE SOIL POLYGON THEN ALL OF IT IS IN.

CALL PIP (XSTR(1), YSTR(1), IND, P, Q, NPTS)

IF (IND.EQ.0) GO TO 910

C IT IS IN SO WRITE THE AREA OUT TO SUMS FILE.

C AREA = LUAREA

A-VII-13
IF (DONUT) AREA=-AREA
NO=MAX+1
WRITE (6,7004) LUCODE(1),LUCODE(3),SPCODE(1),SPCODE(3)
7004 FORMAT ('* ALL OUT----- MAJOR ',A3,2X,'MINOR ',A3,2X) 07290000
WRITE (14,'(A850)') MAPNO,MAPOLI(1),SPMAP,SPCODE(1),SPCODE(3),AREA
1XMIN,YMIN,XMAX,YMAX,X(IJ),Y(IJ),I,J=1,MAX
WRITE (6,870) SPMAP,MAPOLI,LUAREA,SPCODE,FILL,AREA
870 FORMAT (1 EOF ON FILE 19I5/
910 CONTINUE
920 WRITE (6,950) INS,MAPOLI(1)
C REPLACEMENT IN CODE
950 FORMAT ('* EOF ON FILE ',I5,' MAJOR POLYGON= ',I5,/) 07460000
WRITE (6,960) 07470000
960 FORMAT ('* PROCESSING ENDED AT EOD*') 07480000
C 4 EOF'S ON FILE 13
DO 980 J=1,4
WRITE (13,'(I30,I30)') (ENDOF(I),I=1,6) 07510000
970 FORMAT (8A4,A2) 07520000
980 CONTINUE
C 4 EOF'S ON FILE 14
DO 1000 J=1,4
WRITE (14,'(I30)') (ENDOF(I),I=1,12) 07550000
990 FORMAT (20A4) 07560000
1000 CONTINUE
DO 1010 J=1,4
WRITE (15,'(I30)') (ENDOF(I),I=1,7) 07600000
1010 CONTINUE
C IF (.NOT.PRTOUT)GOTO 1999
DO 1100 I=1,10
WRITE (6,1109)
1109 FORMAT ('* FILE-11---') 07670025
1110 READ (11,'(I30)') (SPMAP,SPCODE(2),SPCODE(1),SPCODE(3),SPPOLY(2),ARE) 07730055
READ (11,'(I30)') (TXMIN,TYMIN,TXMAX,TYMAX,NOPTS,P(I),Q(I),I=1,NOPTS) 07790085
WRITE (6,1130) (SPMAP,SPCODE(2),SPCODE(1),SPCODE(3),SPPOLY(2),ARE)
WRITE (6,1201)
1201 FORMAT('* FILE-12---') 07740060
IV2=1
DO 1230 IJ=1,10
WRITE (6,1210) SPMAP,SPCODE(2),SPCODE(1),SPCODE(3),SPPOLY(2),ARE 07750065
1210 FORMAT(1A,TXMIN,TYMIN,TXMAX,TYMAX,NOPTS,P(I),Q(I),I=1,NOPTS) 07790085
A-VII-14
C PRINT OUT 14 ON REQUEST

CONTINUE

C FIND (* FILE-14 --- *)

DO 10 13

10 CONTINUE

STOP

C ENO

C POLYRD/TIDY

C 111111 LATEST VERSION 9 JUNE '75 11111

SUBROUTINE POLYRD(* •MAPNO •POLYNO •CODE •AREA sXMIN sYMIN sXMAX sYMAX

1 YMAX •N.X.Y.INUNIT.FAC.DONUT.OUT./INV/) 08100000

DIMENSION X(1),Y(1) 08100000

INTEGER MAPNO,POLYNO(2),CODE(3) 08100000

LOGICAL DONUTsOUT 08100000

FAC=1.o 08100000

DONUT=.FALSE. 08100000

1 CONTINUE

INQ=INV

READ (INUNIT INV,20,ERR=150) MAPNO, CODE(2), CODE(1), CODE(3), ITYPE, AREA, XMIN, YMIN, XMAX, YMAX, N, (X(K), Y(K), K=1,N)

20 FORMAT (2I59A3sI4sI2.F15.2/4F9.O9I5/(8F9.0)) 08100000

C SOFTWARE EOF FLAG

IF (MAPNO.EQ.99999) GO TO 150

IF (MAPNO.EQ.00000) GO TO 150

C CALCULATE AREA FOR POLYGON

IFLAG=.FALSE.

C SKIP IF AREAOF CALLED PREVIOUSLY ON THIS P —GON

IF (.NOT.IFLAG) GOTO 30

FIND (INUNIT INQ)

WRITE (6s200) MAPNO s CODE(2) s CODE(3)

WRITE (INUNIT 0INQ) MAPNO, CODE(2), CODE(1), CODE(3), ITYPE, AREA, XMIN, YMIN, XMAX, YMAX, N, (X(K), Y(K), K=1,N)

30 CONTINUE

IF (ITYPE.EQ.10) GO TO 40

DONUT=.TRUE.

GO TO 60

40 DO 50 J=2,N

50 CONTINUE

C 1 CONTINUE

C NO=J-1

C X1(Q)=X(J) THIS PART FOR CENTROID ONLY

C Y1(Q)=Y(J)

C 50 CONTINUE

C N=N-1

A-VII-15
A-VII-16
50 AREA = AREA + A*B1 - A*B 
90000000
51 IF (AREA) GT 75, 75, 51 
90100000
51 1 FLAG = .TRUE. 
90200000
C 
90300000
C 
90400000
C 
90500000
C 
90600000
C 
90700000
C 
90800000
C 
90900000
C 
91000000
C 
91100000
C 
91200000
C 
91300000
C 
91400000
C 
91500000
C 
91600000
C 
91700000
C 
91800000
C 
91900000
C 
92000000
C 
92100000
C 
92200000
C 
92300000
C 
92400000
C 
92500000
C 
92600000
C 
92700000
C 
92800000
C 
92900000
C 
93000000
C 
93100000
C 
93200000
C 
93300000
C 
93400000
C 
93500000
C 
93600000
C 
93700000
C 
93800000
C 
93900000
C 
94000000
C 
94100000
C 
94200000
C 
94300000
C 
94400000
C 
94500000
C 
94600000
C 
94700000
C 
94800000
C 
94900000
C 
95000000
C 
95100000
C 
95200000
C 
95300000
C 
95400000
C 
95500000
C 
95600000
C 
95700000
C 
95800000
C 
95900000
C 
96000000
C 
96000000
C 
A-VII-17
C 10 CONTINUE
C IF (DIST .LE. 20000.) GO TO 20
C I = I + 1
C HALVE THE MAGNITUDE OF THE VECTOR UNTIL THE MAGNITUDE IS LESS THAN OR EQUAL TO THE VALUE OF .1
C NOTE... DIST IS THE SQUARE OF THE MAGNITUDE
C DIST = DIST / 4.
C GO TO 10
C 20 CONTINUE
C COMPUTE THE SLOPE, IF DEFINED
C IF(ABS(DLX) .LT..005 ) GO TO 30
C SLOPE = DLY / DLX
C COMPUTE INCREMENTAL VECTORS
C SPX = SIGN (SORT(DIST / (1. + SLOPE ** 2)), DLX)
C SPY = SLOPE * SPX
C GO TO 40
C 30 CONTINUE
C COMPUTE THE SPY VECTOR INCREMENT IF SLOPE IS UNDEFINED
C SPX = 0.
C SPY = SIGN (SORT(DIST), DLY)
C 40 CONTINUE
C NOSTP = 2 ** I
C 50 CONTINUE
C RETURN
C END

FUNCTION LEVEL(Y,YMIN,STEP,ISTP,YMAX)

BLOW=YMIN
DO 10 I=2,ISTP
BLOW=BLOW+STEP
BHIH=BLOW+STEP
IF(I .EQ. ISTP) BHIH = YMAX
IF(Y.LE.BLOW) GO TO 10
IF(Y.GT.BHIH) GO TO 10
LEVEL=1
GO TO 20
10 CONTINUE
LEVEL=1
20 RETURN
C END

SUBROUTINE PIP(PP,IND,X,Y,N)
DIMENSION X(N),Y(N)
LOGICAL SW
C***************************************************************************
10210000
The purpose of this subroutine is to determine if the point \((PP,QQ)\), is in the polygon defined by the arrays \(X\) and \(Y\), which are of length \(N\). \(IND=0\) if the point is not in the polygon, and \(IND=1\) if the point is in the polygon.

The method used is to draw a directed line segment from the point \((PP,QQ)\) to \((\text{infinity},QQ)\). Hence the line starts from \((PP,QQ)\) and lies in the plus \(X\) direction with slope zero.

If this directed line segment crosses the polygon sides an odd number of times the point \((PP,QQ)\) is in the polygon. If the directed line segment crosses the polygon sides an even number of times the point lies outside of the polygon. If the point lies on the side of the polygon it is considered in the polygon.

\(SWT=\) false.

In considering the points in the polygon arrays, points \(I\) and \(I+1\) are considered in each loop. Therefore the loop limit \(M=N-1\).

At times in the algorithm it is necessary to keep track of the last \(Y\) vertex coordinate since all \(Y\) values are positive. -1 is used as a null value.

\(\text{YREM}=-1\)

\(IND=0\)

Loop thru the polygon points.

Do 10 \(I=1,M\)

Store array variables into scalars to eliminate unnecessary indexing.

\(YP=Y(I)\)
\(YYP=Y(I+1)\)
\(XP=X(I)\)
\(XXP=X(I+1)\)

The line segment defined by \((XP,YP)\) and \((XXP,YYP)\) is considered.
If the point is completely above 10840000, the line segment there will be no crossing.

If \( Q_0 > Y_P \) and \( Q_Q > Y_{YP} \), go to 77

Like wise if the point is below 10840000,

If \( Q_Q < L_T < Y_P < Y_{YP} \), go to 77

Or completely to the right of both points \( X_P \) and \( X_{XP} \)

If \( P_P < X_P \) and \( P_P < X_{XP} \), go to 77

Consider the special case when \( Q_Q = Y_P \) or branch around.

In this case the polygon side can come up to the point, turn around and go back, or it can actually cross it.

If \( Q_Q \neq Y_P \) go to 20

Are the points \( (P_P, Q_Q) \) and \( (X_P, Y_P) \) the same? If so the point is considered in.

If \( P_P \leq X_P \), go to 30

If this vertex was encountered last time thru the loop then branch out; otherwise the crossing gets counted twice.

If \( Q_Q \neq Y_{REM} \) go to 40

Remember the vertex so it can be skipped next time thru.

\( Y_{REM} = Y_P \)

Search backward until there is a vertex point with a \( Y \) value not \( Q_Q \).

\( L_{AST} = I \)

50 \( L_{AST} = L_{AST} - 1 \)

If \( L_{AST} = 0 \) go to 10

If \( Q_Q = Y(L_{AST}) \) go to 50

\( Y_{LAST} = Y(L_{AST}) \)

Search forward until there is a vertex point with a \( Y \) value not \( Q_Q \).

\( NEX_{T} = I \)

60 \( NEX_{T} = NEX_{T} + 1 \)

If \( NEX_{T} = N \), go to 10

If \( Q_Q = Y(NEX_{T}) \) go to 60

\( Y_{NEXT} = Y(NEX_{T}) \)

If \( Q_Q \) is greater than both then there is no crossing.

If \( Q_Q \) is less than both there is no crossing.

If \( Q_Q = Y(L_{AST}) \) and \( Q_Q \geq Y_{NEXT} \), go to 10
IF (PP.LE.XP.AND.PP.LE.XXP) GO TO 70

GO TO 10

CONSIDER THE SPECIAL CASE WHEN QQ=YYP OR BRANCH AROUND.

IN THIS CASE THE POLYGON SIDE CAN COME UP TO THE POINT, TURN AROUND AND GO BACK, OR IT CAN ACTUALLY CROSS IT

20 IF (QQ.NE.YYP) GO TO 80

ARE THE POINTS (PP,QQ) AND (XXP,YYP) THE SAME. IF SO THE POINT IS CONSIDERED IN.

IF (PP.EQ.XXP) GO TO 30

IF THIS VERTEX WAS INCOUNTERED LAST TIME THRU THE LOOP THEN BRANCH OUT;

OTHERWISE THE CROSSING GETS COUNTED TWICE.

IF (QQ.EQ.YREM) GO TO 40

REMEMBER THE VERTEX SO IT CAN BE SKIPPED NEXT TIME THRU.

YREM=YYP

SEARCH BACKWARD UNTIL THERE IS A VERTEX POINT WITH A Y VALUE NOT QQ.

LAST=I+1

IF (QQ.EQ.Y(LAST)) GO TO 90

VLAST=Y(LAST)

SEARCH FORWARD UNTIL THERE IS A VERTEX POINT WITH A Y VALUE NOT QQ.

NEXT=I+1

IF (QQ.GT.YLAST.AND.QQ.GT.YNEXT) GO TO 10

IF QQ IS GREATER THAN BOTH THEN THERE IS NO CROSSING.

IF (QQ.GT.YLAST.AND.QQ.GT.YNEXT) GO TO 10

IF QQ IS LESS THAN BOTH THERE IS NO CROSSING.

IF (QQ.LT.YLAST.AND.QQ.LT.YNEXT) GO TO 10

CHECK FOR CROSSING

IF (PP.LE.XP.AND.PP.LE.XXP) GO TO 70

RESET TO NULL VALUE

40 IF (YP.NE.YYP) YREM=-1.0

GO TO 10

IF THE PROGRAM FLOW GETS TO HERE AND PP IS COMPLETELY TO THE LEFT OF THE LINE SEGMENT THERE IS A CROSSING

80 YREM=-1.0

IF (PP.LE.XP.AND.PP.LE.XXP) GO TO 70

A-VII-21
IF THE LINE SEGMENT IS VERTICAL
THE POINT IS ON THE SIDE AND HENCE
IS IN

IF(XP.EQ.XXP) GO TO 30

IF THE LINE SEGMENT IS HORIZONTAL
THE POINT IS ON THE SIDE AND HENCE
IS IN.

IF(YP.EQ.YYP) GO TO 30

COMPUTE THE X COORDINATE
VALUE OF THE INTERSECTION OF THE
LINE SEGMENT FROM (-INFINITY,QQ) TO
(INFINITY,QQ) AND THE SEGMENT
(XP,YP),(XXP,YYP)

XTEMP=(XP-XXP)*(QQ-YYP)/(YP-YYP)+XXP

IF(PP.GT.XTEMP) GO TO 10

IF PP IS TO THE RIGHT THERE IS NO
CROSSING

IF(PP.EQ.XTEMP) GO TO 30

IF PP IS ON THE LINE THE POINT
IS IN

OTHERWISE THERE IS A CROSSING
AND SWT IS FLIPPED.

70 SWT=.NOT.SWT
GO TO 10

THE POINT IS ON THE SIDE AND
IS DEFINED AS IN.

30 SWT=.TRUE.
GO TO 200

77 YREM=-190
10 CONTINUE
200 CONTINUE

IF SWT IS TRUE SET IND TO 1
RETURN
END

ROUTINE TO READ CARDS FOR MAJOR MINOR POLYGONS ONTO FILE11.
SUBROUTINE CLUGE
INTEGER IBUF(20)
INTEGER MINR/\MINR*/.LEND./'END\ END*/'MORE*/MORE*/
INTEGER NINE\NIN,\ BLANK(17)
DATA NINE/9999/\NIN /9/\BLANK/17*'/
DATA ICT/1378*/
COMMON /ALPHA/IV1,IV2,IV3,IV4,IV5
IV1=1
IV2=1
FIND (11'1)
FIND (12'1)

START BY READING MAJOR POLY CARDS
MAJOR FIRST
10 READ (3,100,END=20) (IBUF(I),I=1,20)
IF (IBUF(1).LT.ICT) GO TO 10
100 FORMAT (20A4)
WRITE (6,111) (IBUF(I),I=1,20),IV1
WRITE (11*IV1,100) (IBUF(I),I=1,20)
111 FORMAT (' FROM 11 *20A4,' IV1='*16)
   FIND (11*IV1)
   GO TO 10
20 CONTINUE
   DO 21 J=1,5
21 WRITE(11*IV1,100) NINE,NIN,BLANK
   20 READ (4,100,END=40) (IBUF(I),I=1,20)
   WRITE(12*IV2,100) (IBUF(I),I=1,20)
   C
   WRITE (6,112) (IBUF(I),I=1,20)
112 FORMAT (' FROM 12 *20A4,' IV2='*16)
   FIND (12*IV2)
   GO TO 30
40 CONTINUE
   DO 31 J=1,5
31 WRITE(12*IV2,100) NINE,NIN,BLANK
   FIND (12*IV1)
   IV1=1
   IV2=1
   DO 511 IJ=1,10
511 READ (11*IV1,100) (IBUF(I),I=1,20)
   WRITE (11*IV1)
   FIND (11*IV1)
   WRITE (6,111) (IBUF(I),I=1,20),IV1
   511 CONTINUE
   DO 611 IJ=1,10
611 READ (12*IV2,100) (IBUF(I),I=1,20)
   WRITE (12*IV2)
   FIND (12*IV2)
   WRITE (6,112) (IBUF(I),I=1,20),IV2
   611 CONTINUE
   RETURN
END
APPENDIX VIII

MAJOR POLYGON DATA REPORT PROGRAM LISTING
DIMENSION PTAREA(30), PERCT(30), POIFF (30)
INTEGER SUMAP(30), SUCODE(30), SUORDR(30), SUTYPE(30)
INTEGER CTRNO, BLKNC, CTORDR, CTYPE
INTEGER BLANK
LOGICAL SKIP
DATA BLANK /4H
SKIP = .FALSE.*
LINECT = 0
1 CONTINUE
WRITE (6,2001)
2001 FORMAT(H16.52X, 'SUMMARY OUTPUT'/
 & #8X, 'LOS ANGELES-SANTA MONICA STUDY AREA')
WRITE (6,2002)
2002 FORMAT(56X, 'MAJOR POLYGON FILE')
WRITE (6,2003)
2003 FORMAT(H16.52X, 'CENSUS CENSUS POLY POLY LAND USE LAND USE /
 & 1*POLY POLY ORIGINAL AREA RESIDUAL AREA DIFFERENCE',
 & 2 PERCENT NUM OF')
WRITE (6,2004)
2004 FORMAT(H16.52X, 'TRACT BLOCK NUM TYPE MAP CODE /
 & 2*NUM TYPE, 1X*RES POLY')
WRITE (6,2005)
2005 FORMAT(H16.52X, '--------- ------- ------ ------ ------ ------
 & 2* ------- ------- ------- ------- ------- -------
 & 3* ------- ------- ------- ------- ------- -------
 & IF(LINECT.EQ.0) GO TO 5
 LINECT = 0
 GO TO 59
5 INUM = 0
 PTAREA (1) = 0.0
 PERCT(1) = 0.0
 POIFF(1) = 0.0
 SUMAP(1) = 0
 SUCODE(1) = 0
 SUORDR(1) = 0
 SUTYPE(1) = 0
C READ (15,101) CTRNO, BLKNC, CTORDR, CTYPE, CTAREA
 READ (15,101) CTRNO, BLKNC, CTORDR, CTYPE, CTAREA
101 FORMAT (215,14,12,FI5.2)
 IF (CTNO.EQ.99999) GO TO 999
 TOAREA = TOAREA + CTAREA
 IF(SKIP) GO TO 15
C 10 READ (13,TV3,102) MAPNO, MAPNO1, LUMAP, LUCODE, LUORDR, LUTYPE, AREA
 10 READ (13,102) MAPNO, MAPNO1, LUMAP, LUCODE, LUORDR, LUTYPE, AREA
102 FORMAT (15,14,12,FI5.2)
 15 CONTINUE
 SKIPI + FALSE*
 IF (MAPNO = CTNCN) 10, 20, 30
20 IF(MAPNO = BLKNC) 10, 40, 30
40 INUM = INUM + 1
 PTAREA(INUM) = AREA
 PERCT(INUM) = 100.0*AREA/CTAREA
 POIFF(INUM) = CTAREA-AREA
 SUMAP(INUM) = LUMAP
 SUCODE(INUM) = LUCODE
 SUORDR(INUM) = LLORDR
 SUTYPE(INUM) = LUTYPE
 IF (PERCT(INUM) < 0.009) PERCT(INUM) = 0.00
 IF (LUTYPE <= 10) GO TO 50
 PTAREA(INUM) = -PTAREA(INUM)
PERCT(INUM) = -PERCT(INUM)
PDIFF(INUM) = -PDIFF(INUM)

50 CONTINUE
TOTPR = TOTPR + PTAREA(INUM)
GO TO 10

30 CONTINUE
MCTRNO = MOD(CTRNO,10)
MBLKNO = MOD(BLKNO,10)
CTRNO = CTRNO/10*10+MCTRNO
BLKNO = BLKNO/10*10+MBLKNO
STOT=0.0
SDIFF=0.0
SPERD=0.0
IF(LINECT.GT.50)GO TO 1

59 CONTINUE
IF(INUM.LE.0) GO TO 61
DO 60 J=1,INUM
LINECT=LINECT+1
60 CONTINUE
WRITE(6*201) SVMAP(J),SUCODE(J),
*SUORCR(J),SU.TYPE(J),PTAREA(J)
201 FORMAT(34X,9A3,6X,12,4X,19,18X,F18.2)
STOT=STOT+PTAREA(J)
60 CONTINUE
SDIFF=CTAREA-STOT
SPERD=100.*SDIFF/CTAREA
WRITE(6*203) CTRNO,BLKNO,CTORCR,CTTYPE,CTAREA,STOT,SDIFF,SPERD,
*INUM
203 FORMAT(IX,16,4X,15,3X,12,16,6X,'TOTAL',
*2F18.2,F13.2,F9.2,F18.2)
GO TO 62

61 CONTINUE
WRITE(6*204) CTRNC,BLKN0,CTORCR,CTTYPE,CTAREA
204 FORMAT(IX,16,4X,15,3X,12,4X,12,36X,
*4X,'*** NO RESIDUAL OVERLAY POLYGON FOUND ***')
62 CONTINUE
LINECT=LINECT+2
SKIP = .TRUE.
GO TO 5

999 CONTINUE
WRITE(6*202) TCAREA,TOTPR
202 FORMAT(1M09,'TOTAL AREA OF CENSUS TRACTS=','F15.2/
*1M09,'TOTAL AREA OF RESIDUAL POLYGONS=','F15.2)
END
APPENDIX IX

MINOR POLYGON DATA REPORT PROGRAM LISTING
DEFINE FILE 10 (3000,800,E,11)
DIMENSION SAREA(30),ISBLK(30)
I=1
J=1
IF(RRS=O)
ICTR=0
JCTR=0
JUMP=0
ASSIGN 100 TO JUMP
ASSIGN 200 TO JUMP
AREA1=0.
AREA2=0.
TOT1=0.
TOT2=0.
ITA2=0
ILINES=0
1 WRITE (6,2021)
2021 FORMAT(1H4.52X, 'SUMMARY OUTPUT'/
#48X, 'LOS ANGELES-SANTA MONICA STUDY AREA')
WRITE (6,2022)
2022 FORMAT(56X, 'MINOR POLYGON FILE')
WRITE (6,2023)
2023 FORMAT(1X, 'CENSUS CENSUS LAND USE LAND USE POLY POLY /
1. ORIGINAL AREA RESIDUAL AREA DIFFERANCE /
2. PERCENT NUM OF')
WRITE (6,2024)
2024 FORMAT(1H4 + ' ' TRACT BLOCK MAP CODE NUM /
2 TYPE -70X, 'RES POLY')
WRITE (6,2025)
2025 FORMAT (1H4 + ' 'ORIGINAL AREA RESIDUAL AREA DIFFERENCE /
1. ORIGINAL AREA RESIDUAL AREA DIFFERENCE /
2. PERCENT NUM OF')
WRITE(6,2024)
IF(ILINES.EQ.0)GO TO 5
GO TO 205
C READ(10'11,1000) IMAP,ITAZ,ILUC,IPOLY,ITYPE,AREA1
5 READ(10,1000) IMAP,IBLK,ITAZ,ILUC,IPOLY,ITYPE,AREA1
1000 FORMAT (I5,15,15,A3,14,I2, F15.2)
ILINES=ILINES+6
INUM=1
C FIND(10'11)
ICTR=ICTR+1
C 33 READ(11'12,1000) JMAP,JTAZ,JLUC,JPOLY,JTYPE,AREA2,N
READ(11,1001) JMAP,JTAZ,JLUC,JPOLY,JTYPE,AREA2,N
1001 FORMAT (15, 15,A3,14,I2, F15.2/ 36X,15)
C FIND(11'12)
A=N
A=(A/5.0)+.9
N=A
I=I+1
JCTR=JCTR+1
C 10 READ(10'11,1000) IIMAP,ITAZ,ILUC,IIPOLY,IITYPE,AREA1,N
10 READ(10,1000) IIMAP,IBLK,ITAZ,ILUC,IIPOLY,IITYPE,AREA1
IF(IIMAP.EQ.99999) GO TO 900
C FIND(10'11)
ICTR=ICTR+1
20 IF(IIMAP.NE.IMAP)GO TO 30
IF(IILUC.NE.ILUC)GO TO 30

A-IX-2
TF(TPOLY.NE.IPOLY)GO TO 30
IF(IITYPE.NE.ITYPE)GO TO 30
ISBLK(1)=IBLK
AREA(1)=AREAI
AREAI=AREAI+AREAI
INUM=INUM+1
SAREA(INUM)=AREAI
ISBLK(INUM)=IBLK
GO TO 10
30 GO TO JUMP,(100,140,200)
40 IMAP=IMAP
IBLK=IBLK
ITAZ=ITAZ
ILUC=ILUC
IPOLY=IPOLY
AREAI=AREAI
ITYPE=ITYPE
INUM=1
GO TO 10
C 100 READ(11,12,1000)JJMAP,JJTAZ,JJLUC,JJPOLY,JJTYPE,AREAJ,N
100 READ(11,1001)JJMAP,JJTAZ,JJLUC,JJPOLY,JJTYPE,AREAJ,N
IF(JJMAP.EQ.99999)GO TO 900
A=N
A=(A/5.0)+.9
N=A
12=I2+N+1
C FINO(11*12)
JCTR=JCTR+1
IF(JJTYPE.GT.10)AREAJ=-AREAJ
120 IF(JJMAP.NE.JMAP)GO TO 130
IF(JJLUC.NE.JLUC)GO TO 130
IF(JJPOLY.NE.JPOLY)GO TO 130
IF(JJTYPE.NE.JTYPE)GO TO 130
AREA2=AREA2+AREAJ
GO TO 100
130 GO TO 1GO,(200)
140 JTYPE=JTYPE
JMAP=JMAP
JTAZ=JTAZ
JLUC=JLUC
JPOLY=JPOLY
AREA2=AREAJ
GO TO 100
200 IF(IPOLY.NE.JPOLY)GO TO 320
ASSIGN 140 TO JUMP
DELTA=AREA2-AREA1
IF(ABS(DELTA).EQ.0.0)GO TO 210
OPCT=(ABS(DELTA)/AREA2)*100.0
IF(ILINES.LT.50)GO TO 210
GO TO 1
205 ILINES=5
210 CONTINUE
DO 215 J=1,INUM
MBLKN=MOD(ISBLK(J),10)
ISBLK(J)=ISBLK(J)/10+MBLKN
WRITE(6,2015)ISBLK(J),JMAP,JLUC,JPOLY,JTYPE,SAREA(J)
2015 FORMAT(1X,1S,16X,7X,A3,6X,12,4X,12,8X,F13.2)
ILINES=ILINES+1
215 CONTINUE
MIMAP=MOD(IMAP,10)
IMAP=IMAP/10+100+MIMAP
WRITE(6,2002) IMAP, JMAP, JLUC, JPOLY, JTYPE, AREA2, AREA1, DELTA*, IDPCT, INUM
ILINES=ILINES+1
TOT1=TOT1+AREA1
TOT2=TOT2+AREA2
IF(Delta.NE.0.0)IERRS=IERRS+1
GO TO 40
320 IF(JPOLY.LT.JPOLY)GO TO 350
GO TO 360
350 ASSIGN 200 TO JUMP
WRITE(6,2001)
WRITE(6,2003) IMAP, JLUC, JPOLY, JTYPE, AREA1
WRITE(6,2001)
GO TO 40
360 WRITE(6,2001)
WRITE(6,2004) JMAP, JLUC, JPOLY, JTYPE, AREA2
WRITE(6,2001)
GO TO 140
900 WRITE(6,2001)
DELTA=TOT2-TOT1
DPCI=(ABS(Delta)/TOT2)*100.0
WRITE(6,2005) TOT2, TOT1, DELTA, DPCT
A=IERRS
B=JCTR
C=(A/B)*100.0
WRITE(6,2006) JCTR, ICTR9, IERRS, C
STOP
2000 FORMAT(1•, 'MAP TAZ LUC POLYGON TYPE ORIGINAL AREA',
1 RESIDUAL AREA DIFFERENCE PERCENT*,
2 RESIDUAL POLYGONS*)
2001 FORMAT(1X)
2002 FORMAT(17,18X,15,7X,A3,6X,I2,4X,I2,6X,F15.2,5X,F15.2,5X,
1F10.2,5X,F6.2,7X,12/)
2003 FORMAT(25X,15,7X,A3,6X,I2,4X,I2,6X,F15.2,4X,
1 **** UNMATCHED RESIDUAL POLYGON *****)
2004 FORMAT(25X,15,7X,A3,6X,I2,4X,I2,6X,F15.2,4X,
1 **** UNMATCHED ORIGINAL POLYGON *****)
2005 FORMAT(4•, 'OVERALL TOTALS ****', 10X,F10.2,6X,F10.2,6X,
1 F10.2,4X,F6.2)
2006 FORMAT(1X, 'ORIGINAL POLYGONS', 15, ' RESIDUAL POLYGONS', 15,
1 ERRORS', 15, ' PERCENT', 15)
END