DEPTH: A System for the Display of Elementary Petroleum Themes

GEOLOGICAL SURVEY PROFESSIONAL PAPER 1297

DEPTH: A System for the Display of Elementary Petroleum Themes

By PATRICIA A. FULTON, ROBERT W. COFFIN, *and* MARK S. KUTSKO

GEOLOGICAL SURVEY PROFESSIONAL PAPER 1297

A system for the display of petroleum data in the forms of maps, graphs, and charts

UNITED STATES GOVERNMENT PRINTING OFFICE, WASHINGTON : 1983

UNITED STATES DEPARTMENT OF THE INTERIOR

JAMES G. WATT, *Secretary*

GEOLOGICAL SURVEY

Dallas L. Peck, ***Director***

Library of Congress Cataloging in Publication Data

Fulton, Patricia.
DEPTH, a system for the display of elementary petroleum themes.

(U.S. Geological Survey professional paper ; 1297)
Bibliography: p.
Supt. of Docs. no.: I 19.16:1297
1. DEPTH (Computer system) I. Coffin, Robert W. II. Kutsko, Mark. III. Title. IV. Title: DEPTH., a system for the display of elementary petroleum themes. V. Series: Geological Survey professional paper ; 1297.
TN871.F84 1983 553.2'8'028542 83-600259

**For sale by the Distribution Branch, U.S. Geological Survey,
604 South Pickett Street, Alexandria, VA 22304**

CONTENTS

ILLUSTRATIONS

[Plates are in pocket]

DEPTH: A SYSTEM FOR THE DISPLAY OF ELEMENTARY PETROLEUM THEMES

By PATRICIA A. FULTON, ROBERT W. COFFIN, AND MARK S. KUTSKO

ABSTRACT

A computerized system called DEPTH (Display of Elementary Petroleum Themes) provides for the display of petroleum data in the forms of maps, charts, and graphs. The system components include graphic and nongraphic information files as well as various computer programs, all of which reside on the U.S. Geological Survey's computers in Reston, Va.

INTRODUCTION

Huge amounts of information on petroleum and other energy sources and minerals reside in computer data banks. Graphic presentation of such data often has greater impact and use than textual or tabular presentation. A practical way to create graphic displays of petroleum data is to use the computer because the data are already in machine-readable form.

PURPOSE AND SCOPE

The purpose of this project is to provide a system for the Display of Elementary Petroleum Themes (DEPTH). Elementary petroleum themes as defined herein emphasize the spatial characteristics of the data and simultaneously combine them with the descriptive portion of the data. One basic item of spatial data is, of course, absolute position on the Earth given by latitude and longitude. For most purposes, however, this information is useless until it is presented with such familiar geographic entities as county and State. A primary task of DEPTH is to organize the data so the spatial portion is obvious and the descriptive portion is enhanced. Three categories of graphics are particularly appropriate for portraying this data—maps, charts, and graphs, all of which are included in DEPTH.

The scope of this initial effort includes all pertinent data and software currently available on any of the U.S. Geological Survey (USGS) computer systems. The major mainframe computer systems include Honeywell Multics, IBM, and Amdahl.

ACKNOWLEDGMENTS

We wish to thank the many people who helped to create the graphics and data files for this paper. William Strauss, Denise Maurer, Ray Wisecarver, and the staff at Johns Hopkins Applied Physics Laboratory, assisted by digitizing the map data. Jerlene Bright, Patrick Keating, Ardoth Meek, Deborah Martin, Patricia Tracy, Tom Weaver, Steve Hinkle, Michelle Summers, and staff members of the Information Systems Programs at Oklahoma University helped in the use of GIPSY (General Information Processing System) and the PDS (Petroleum Data System) data files. Belinda Dixon of Tektronix aided in the operation of the Tektronix terminals, plotters, and software. William Nisen and Duane Niemeyer and the staff at the Laboratory for Computer Graphics and Spatial Analyses assisted in the use of the Harvard Graphics Package.

THE SYSTEM

GENERAL DESCRIPTION

Two main types of data bases comprise the system, nongraphic and graphic. The major nongraphic type of data base is the Petroleum Data System (PDS) from the University of Oklahoma (1981). The graphic type of data base is composed of computer programs that use proprietary software packages and files of geographic coordinate data. The proprietary software packages are Disspla (Integrated Software Systems Corp., 1978), Datagraphing (Eng and Laroff, 1980), and CAM (U.S. Central Intelligence Agency, 1975). Design philosophy and economic policy dictate that existing software and data be used whenever possible. The graphic files are county outlines (Fulton and Johnson, 1982), county names, international boundaries, oil field outlines and names, gas field outlines and names, and outlines of the geologic provinces (Meyer, 1970).

PETROLEUM DATA SYSTEM

The Petroleum Data System of North America was developed and is maintained by the Office of Information Systems Programs, University of Oklahoma, under a contract with the Geological Survey. The American Petroleum Institute (API) and the American Association of Petroleum Geologists (AAPG) collect and maintain the well information contained in the master well files and exploratory well files. The following oil and gas

files from the PDS are installed on the USGS computer system:

File name	Description
1. OILY	All States except Texas
2. TEXS	Texas data
3. CNDN	Canadian data
4. CHST	Canadian historical pool
Do	Reduction/injection data
5. SECR	Secondary recovery data
6. PBRF	Permian Basin reserve file

The PDS is accessed via the General Information Processing System (GIPSY) which is also resident on the USGS computer system. The GIPSY software is a product of the University of Oklahoma (1977).

COORDINATE DATA

The coordinate files for graphics are generated on a manual digitizer that has a resolution of 0.001 inch. During the digitization process the data are plotted and checked for accuracy. Necessary corrections are made, and the data are replotted. This process continues until all visible errors are eliminated.

One of the primary tasks for DEPTH is the validation of the coordinate data after the files have been installed on the USGS computers. First, the data are plotted interactively on a cathode ray tube (CRT) computer terminal and then visually inspected. When errors are seen in the data, usually big spikes, they are corrected if possible. When it is not possible to correct the data, they are deleted and a new file of the same data is requested. When the CRT plot appears correct, the output is directed to a pen plotter and plotted offline. This plot is then inspected visually.

PERMANENT COORDINATE DATA FILES

The permanent coordinate data consist of the following files:

1. Oil field and gas field outlines and centerpoints
2. County outlines
3. County names
4. Geologic basins

TEMPORARY COORDINATE DATA FILES

The temporary coordinate data files are generally subsets of the permanent files. The temporary files are created for two reasons. First, some programs demand that the data be in a specified format. CAM is such a program. Second, only certain parts of the data may be needed. This situation occurs typically when PDS has been queried and specific data elements have been selected. Figures 1 through 5 are examples of such subsets.

DOCUMENTATION

A list of the files, both permanent and temporary, and of formats 1 through 12, which contain the details of the data formats, is presented in appendix A.

The operational instructions are given in appendix B. These are the detailed instructions on how to run the computer programs. The computer program reference, appendix C, provides documentation on the computer programs including listings of the coding.

ELEMENTARY GRAPHICS

The creation of the graphics begins with a search of the files using criteria selected for a particular purpose. Production figures are of great interest and significance because they are factual, generally reliable, and basic to most statistical studies; they probably constitute a part of most file searches. These data also lend themselves to portrayal by a variety of methods, several of which are depicted in figures in this report. Figure 1 is a bar graph of gas production in Arkansas for the years 1968 through 1976. Figure 2 is a pie chart showing oil production in Wyoming for the years 1968 through 1974. Figure 3 is a patterned bar chart showing both oil and gas production in Louisiana for the years 1968 through 1974.

All the graphics contained in this paper can be used on different types of display media. Figures 1, 2 and 3 were made on a graphics terminal and hard copy unit using the Tektronix Data Graphing package. Figure 4 is a shaded line graph showing both gas and oil production in Texas for the years 1968 through 1976. Figure 5 is another shaded line graph showing both oil and gas production in Mississippi for the years 1968 through 1976.

Figures 4 and 5 were first generated on a Tektronix graphics terminal and then plotted in color on a small vector plotter. The shading is actually a solid tone; on the original output, red represents gas and blue represents oil. These plots can be used immediately for information display or they can be reproduced for use with an overhead projector or as 35mm slides.

Petroleum production and use has a spatial component and can also be illustrated effectively via maps. The Harvard Computer Graphics Software was used to create a series of map plots interactively on a graphics terminal. Gas and oil production values were extracted from the PDS files for eight of the 48 conterminous States; all other States in the series of plots were assigned a value of zero. Figure 6 shows the States for which production values were retrieved, and the rest of the map is blank. Figure 7 shows these same States, plus Wyoming, with a solid fill and the rest of the States as outlined patterned areas. Figure 8 is the same data, but default values are assigned by the software to the crosshatch pattern and to the table of class intervals (Hoel, 1958).

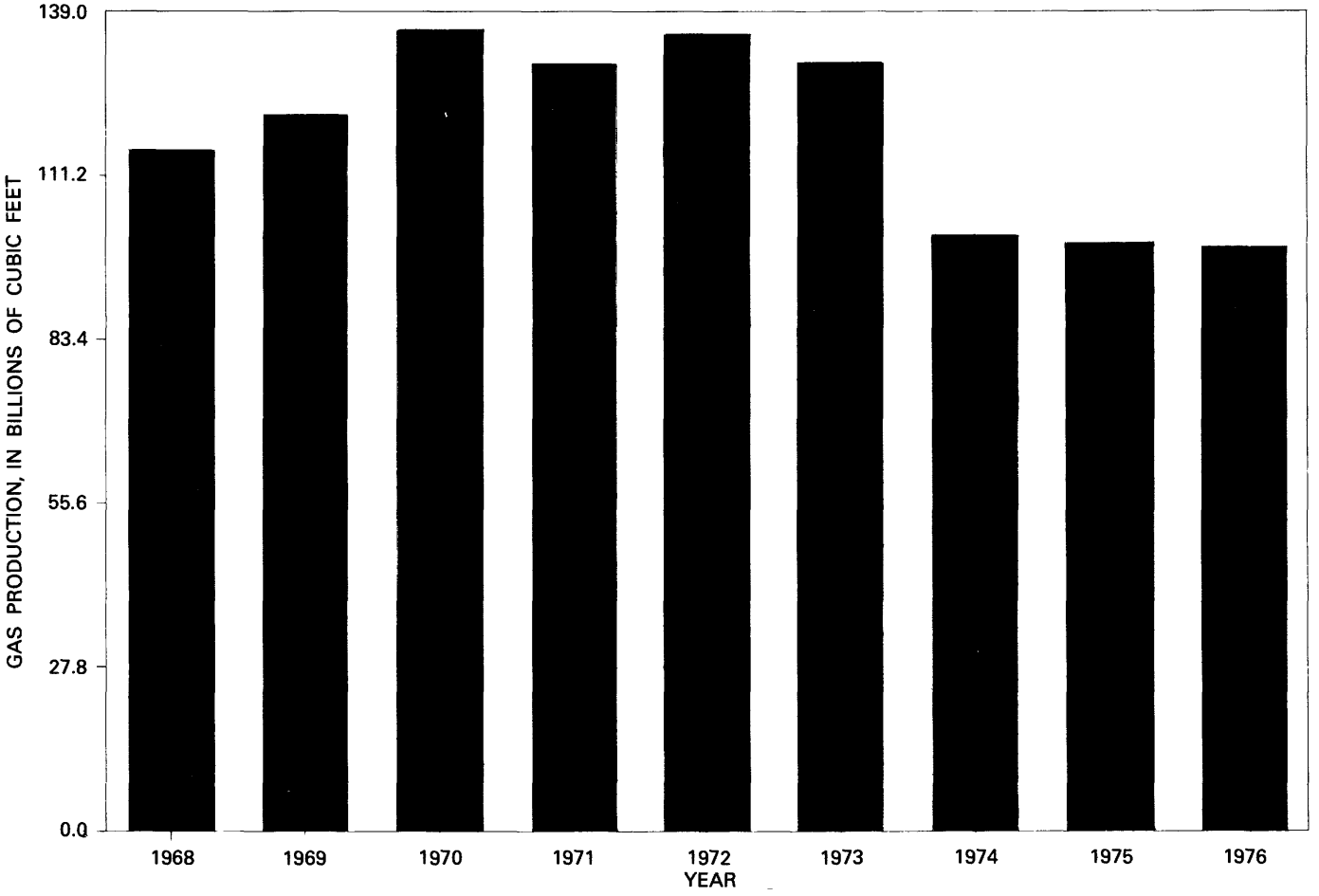


FIGURE 1.–Plot showing gas production in Arkansas for 1968 through 1976.

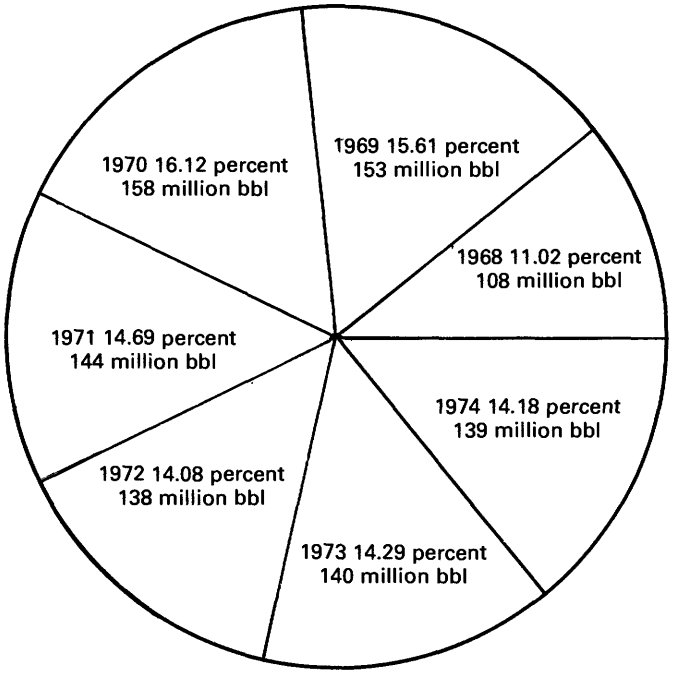


FIGURE 2.–Plot showing oil production in Wyoming for 1968 through 1974.

Figure 9 is an interactive enlargement of a portion of figure 8. Figure 10 is another version of the same data; here, however, the class intervals have been chosen to reflect the true distribution of the values. Similarly, the patterns have been chosen to provide a more pleasing appearance. A perspective view of the country can convey regional information in a striking manner–figure 11 is still the same data, but now the country appears as though a spectator is looking down upon it at an angle from above, and the States are elevated above the surface in proportion to their production values for gas. This figure typifies the default parameters of the program. The view angle and elevation can be changed to suit the data. Figure 12 illustrates gas production for selected States in 1972 as seen from a perspective view from the southeast towards the northwest. Figure 13 shows oil production for selected States in 1971 as seen from the northwest. Figure 14 is a plot of simulated geologic provinces. This particular plot was generated as a preview of applications using the digitized geologic provinces file. Most of these plots are small, about page size.

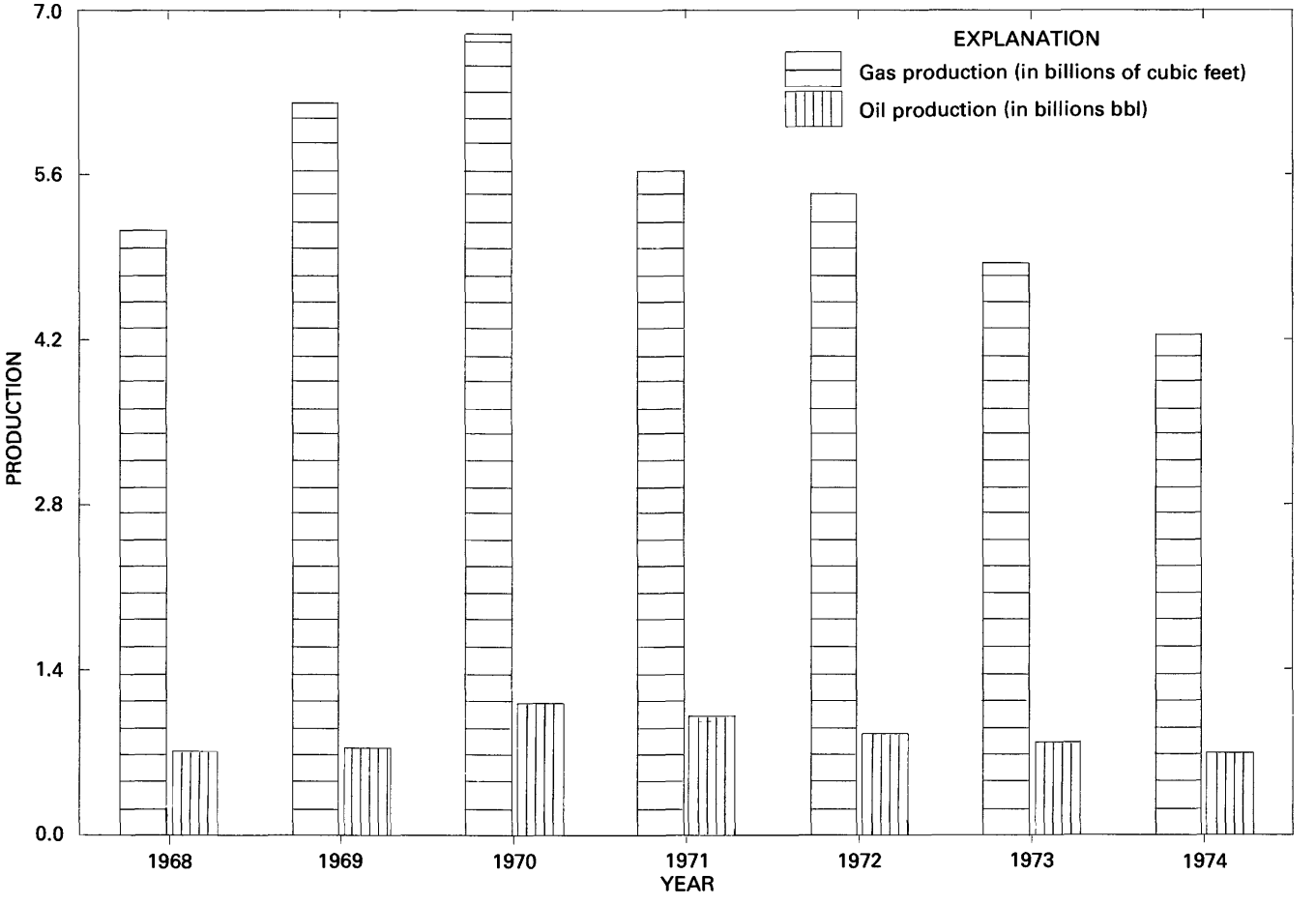


FIGURE 3.–Plot showing oil and gas production in Louisiana for 1968 through 1974.

The graphics in the next group are large. Plate 1 is a plot of the digitized file of geologic provinces. Plate 2 is a map of the oil fields and gas fields in Colorado. Plate 3 is a map of the gas fields in Oklahoma with the field names also plotted.

Plate 2, a map of oil fields and gas fields in Colorado, and plate 3, a map of gas fields in Oklahoma with field names, were created using the CAM software and a Calcomp drum plotter. Plate 4 shows the Fort Chaffee area of Arkansas. All of the oil fields and gas fields are outlined. The fields, which contain pools that were discovered between 1900 and 1950, are crosshatched. The oil fields are hatched vertically and the gas fields horizontally. Plate 5, similar to plate 4, shows the same geographic area but represents fields which contain pools that were discovered between 1960 and 1964. The gas fields are hatched at an angle of 45 degrees and the oil fields at an angle of 135 degrees. Plate 4 and 5 were generated by Disspla software and a Calcomp drum plotter.

SUMMARY

The process of generating basic graphics for petroleum data utilizes the PDS data, GIPSY, and the hardware and software available at the Geological Survey in Reston, Va. The graphics themselves range from bar charts and graphs to shaded thematic maps. The time required to create these graphics varies from minutes to days.

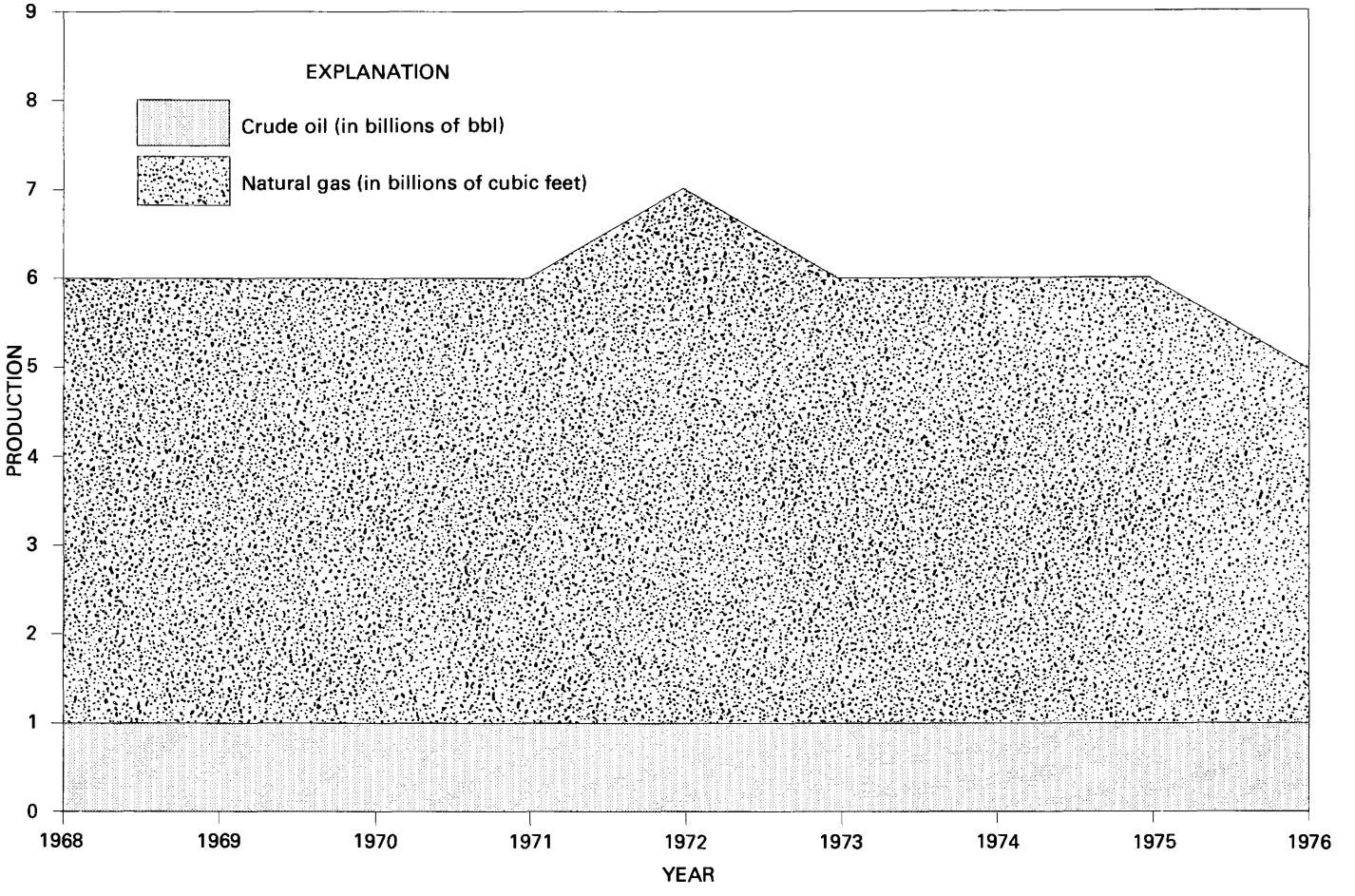


FIGURE 4.—Plot showing oil and gas production in Texas for 1968 through 1976.

REFERENCES CITED

Eng, Chuck, and Laroff, Gary, 1980, 4051 Data graphing: Tektronix.

Fulton, Patricia, and Johnson, Harold, 1982, Geoindex: U.S. Geological Survey Professional Paper 1172, 298 p.

Hoel, P.G., 1958, Introduction to mathematical statistics: New York, Wiley and Sons, p. 44–46.

Integrated Software Systems Corporation, 1978, Disspla users manual: San Diego, California.

Meyer, R. F., 1970, Geologic provinces code map for computer use: American Association of Petroleum Geologists Bulletin, v. 54, no. 7, p. 1301–1305.

U.S. Central Intelligence Agency, 1975, Cartographic automatic mapping program documentation, version 4: OGCR CD 75–1, 111 p.

University of Oklahoma, 1977, General information processing system users guide: v. 1 and 2.

———1981, Petroleum data system of North America users guide: 274 p.

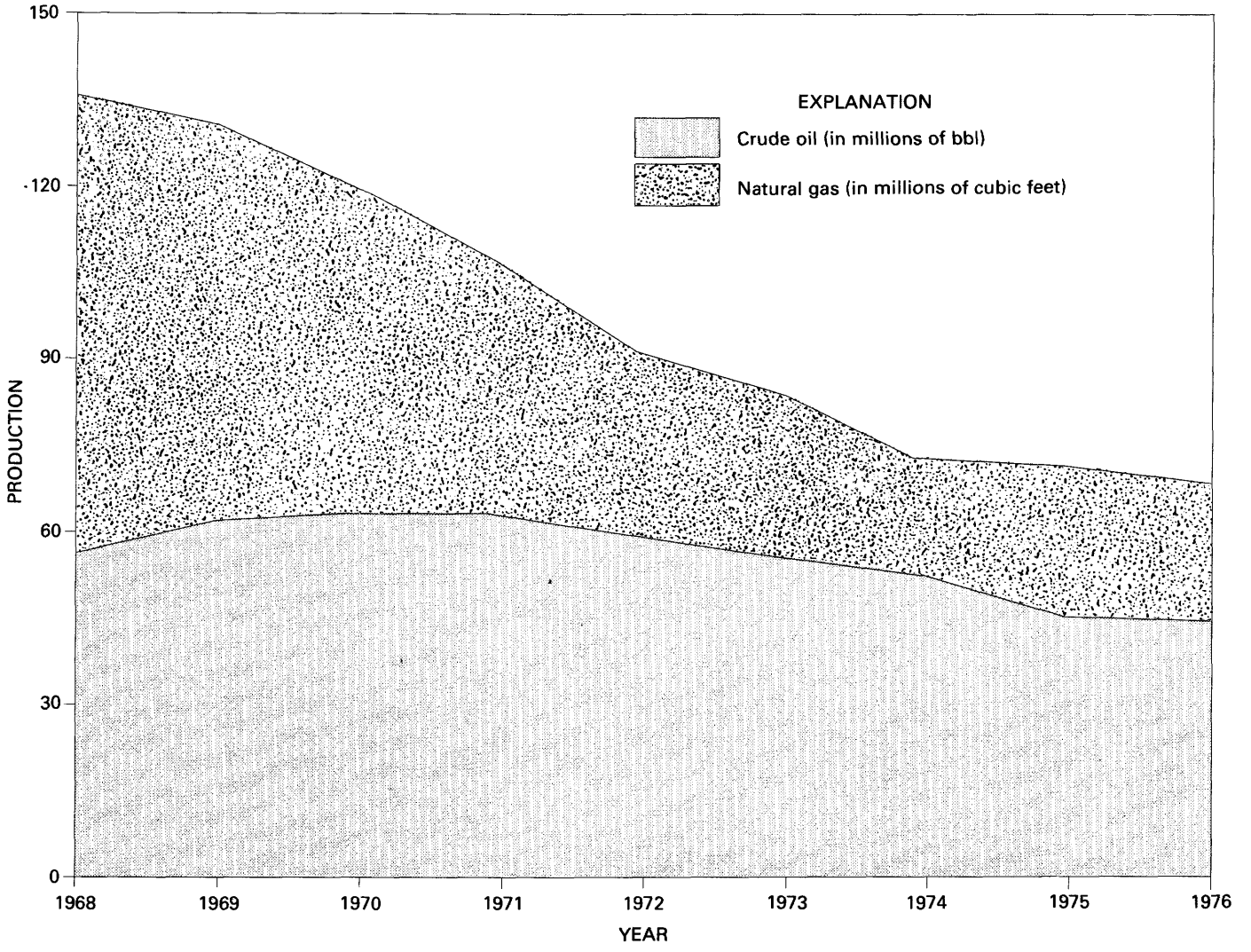


FIGURE 5. – Plot showing oil and gas production in Mississippi for 1968 through 1976.

FIGURE 6.—(Above right). Plot showing gas production in 1968 for selected States for which values were retrieved from PDS files.

FIGURE 7.—(Below right). Plot showing gas production in 1968 for the selected States, plus Wyoming, with full outline of country.

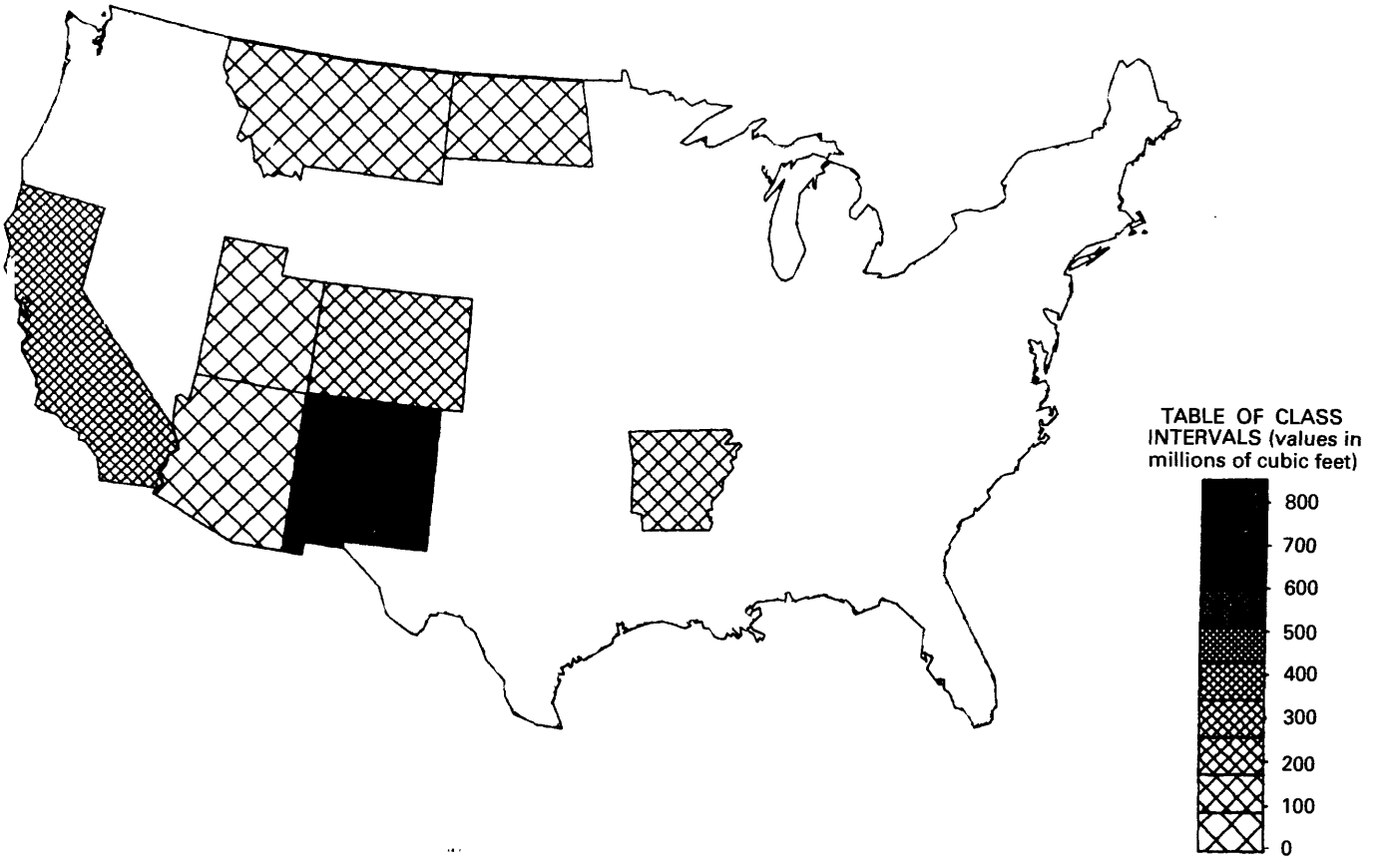


TABLE OF CLASS INTERVALS (values in millions of cubic feet)
800
700
600
500
400
300
200
100
0

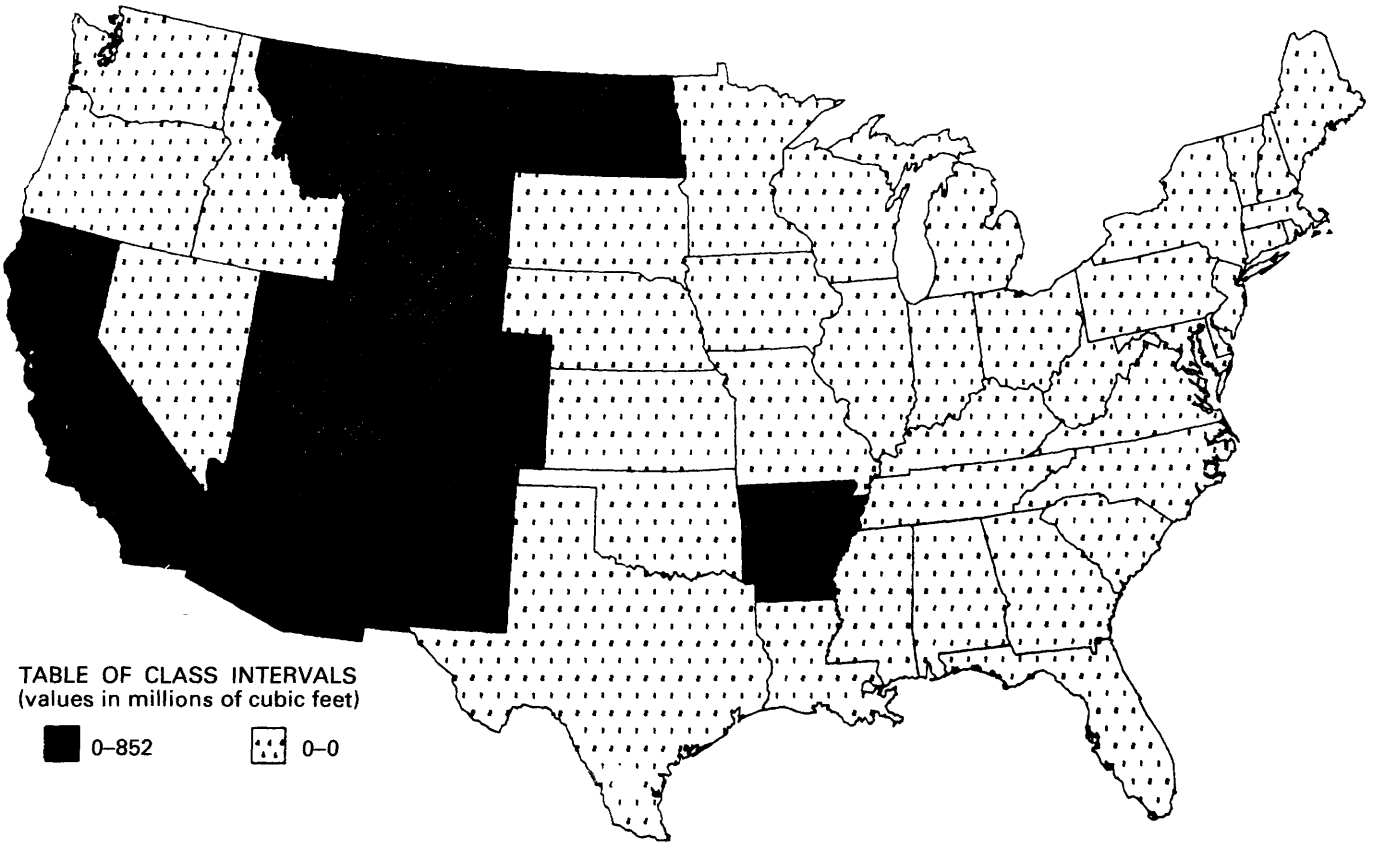


TABLE OF CLASS INTERVALS (values in millions of cubic feet)
0–852
0–0

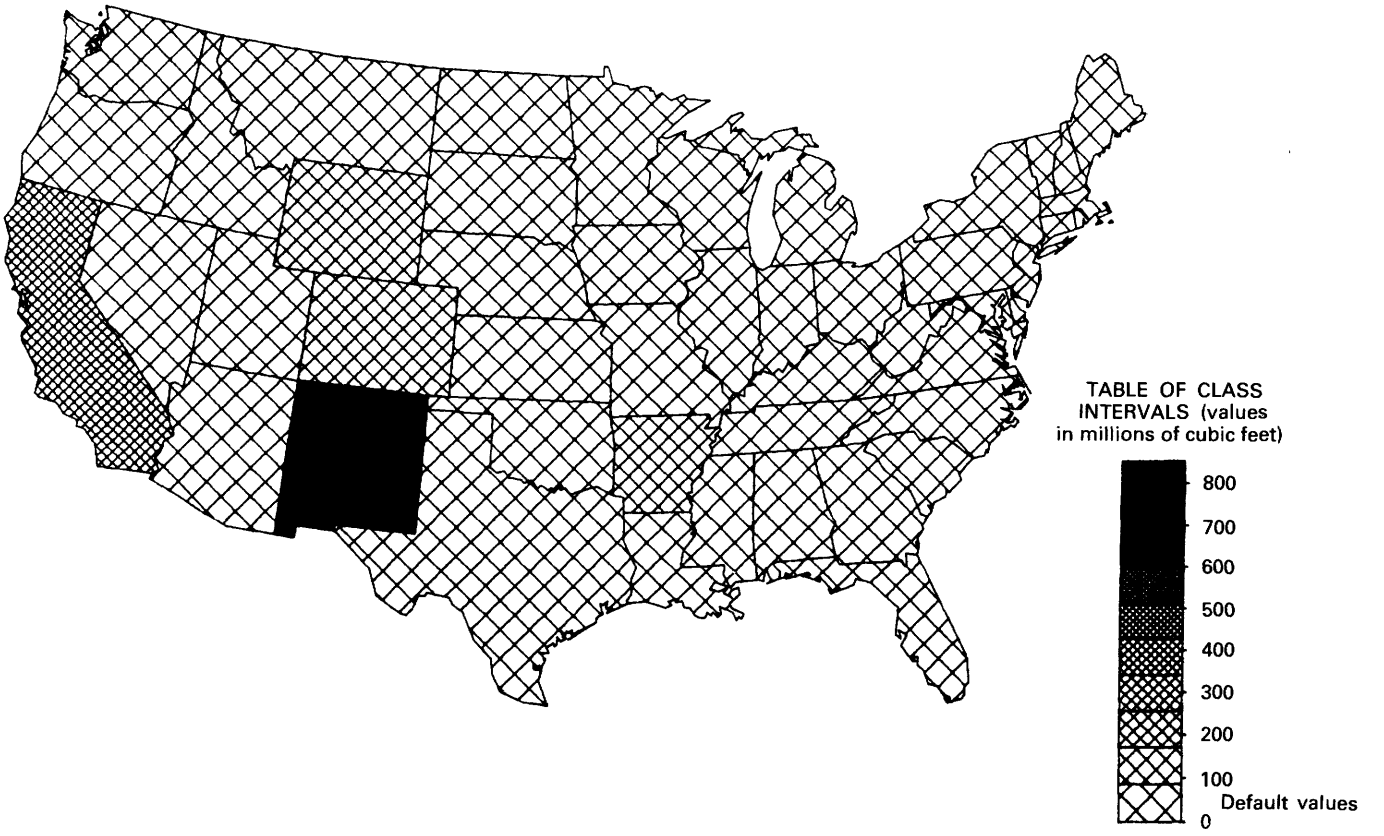


TABLE OF CLASS INTERVALS (values in millions of cubic feet)
800
700
600
500
400
300
200
100
Default values
0

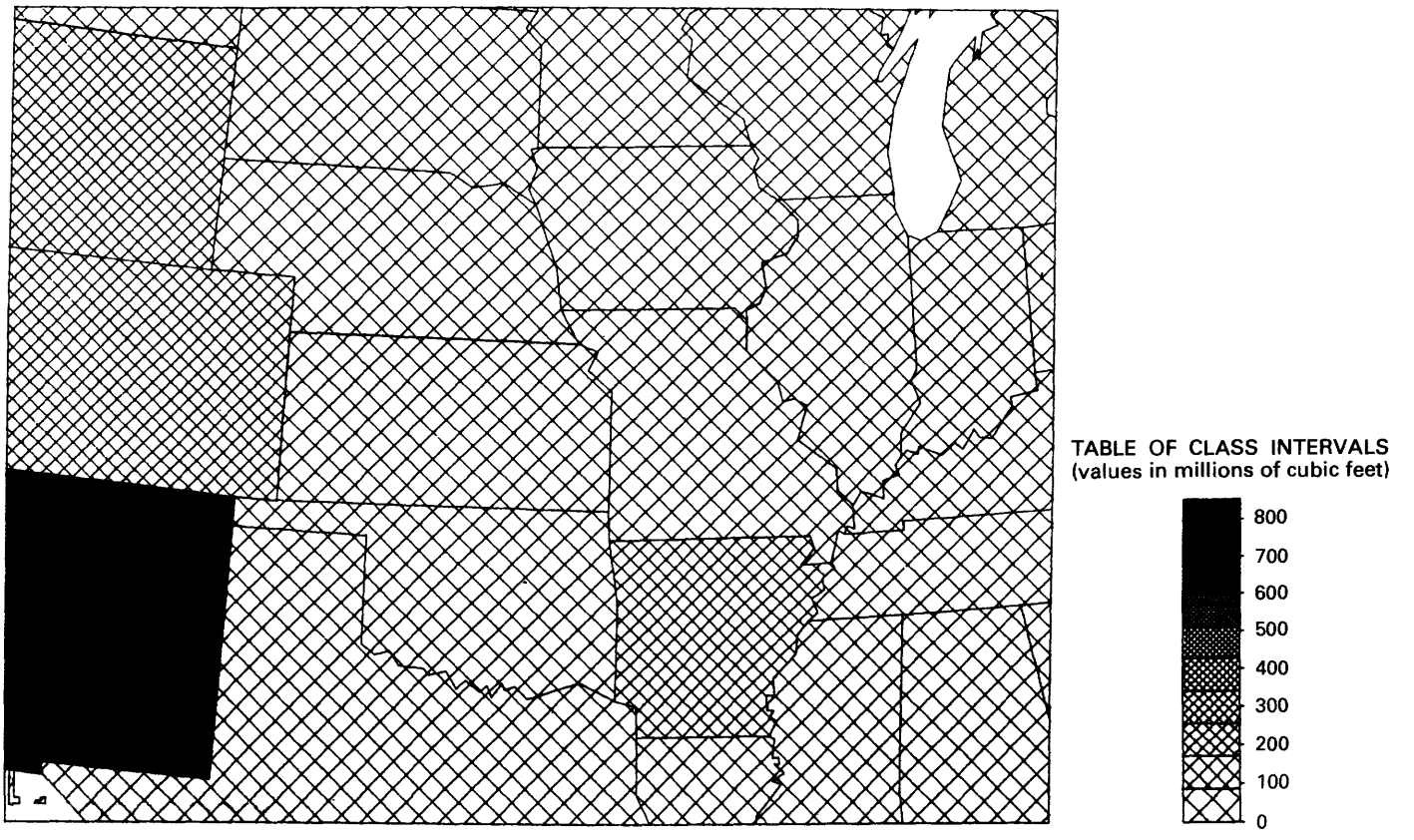


TABLE OF CLASS INTERVALS (values in millions of cubic feet)
800
700
600
500
400
300
200
100
0

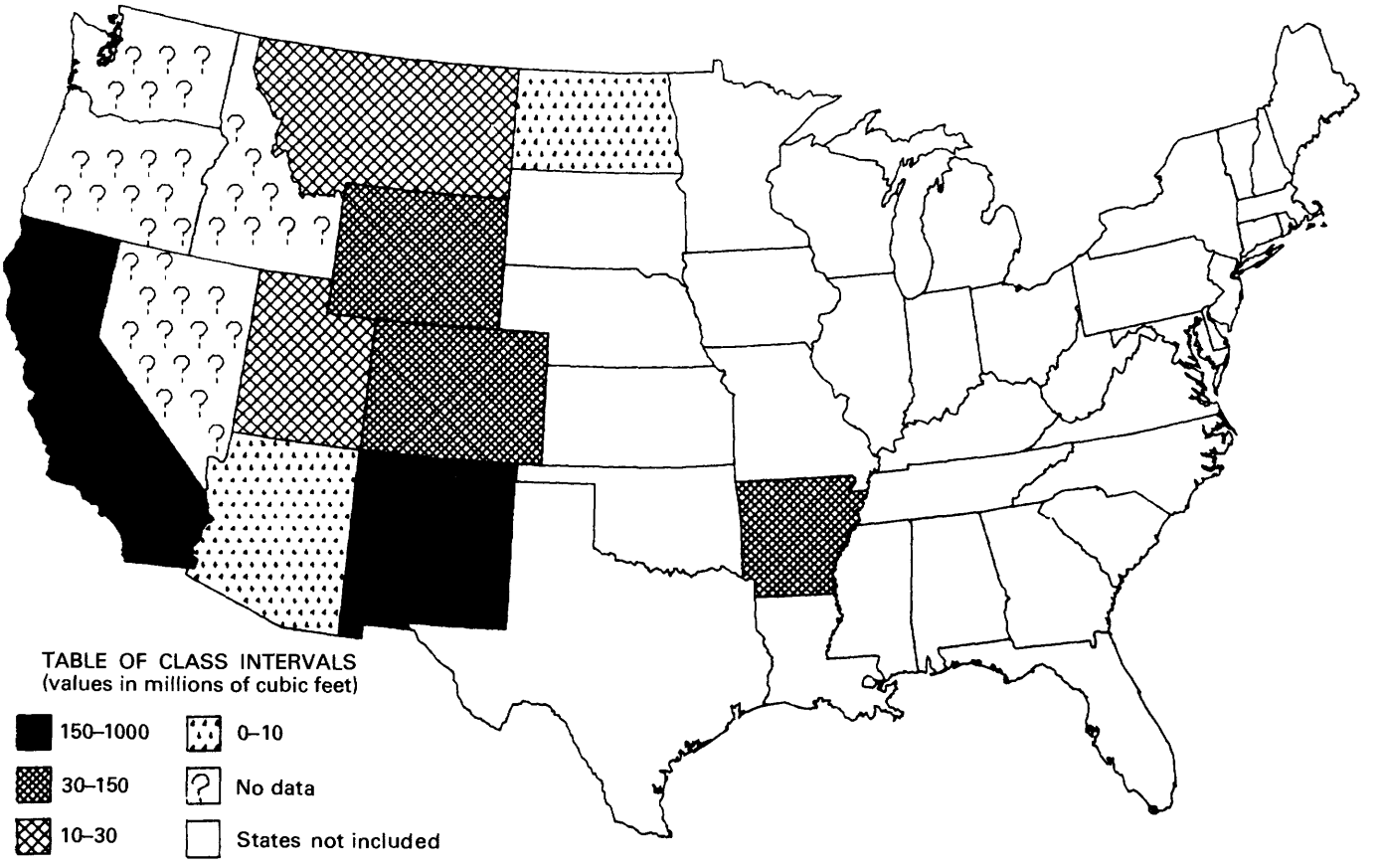


FIGURE 10.—Plot showing gas production in 1968 for the selected States, with assigned patterns and class intervals incorporated to reflect the true distribution of the values.

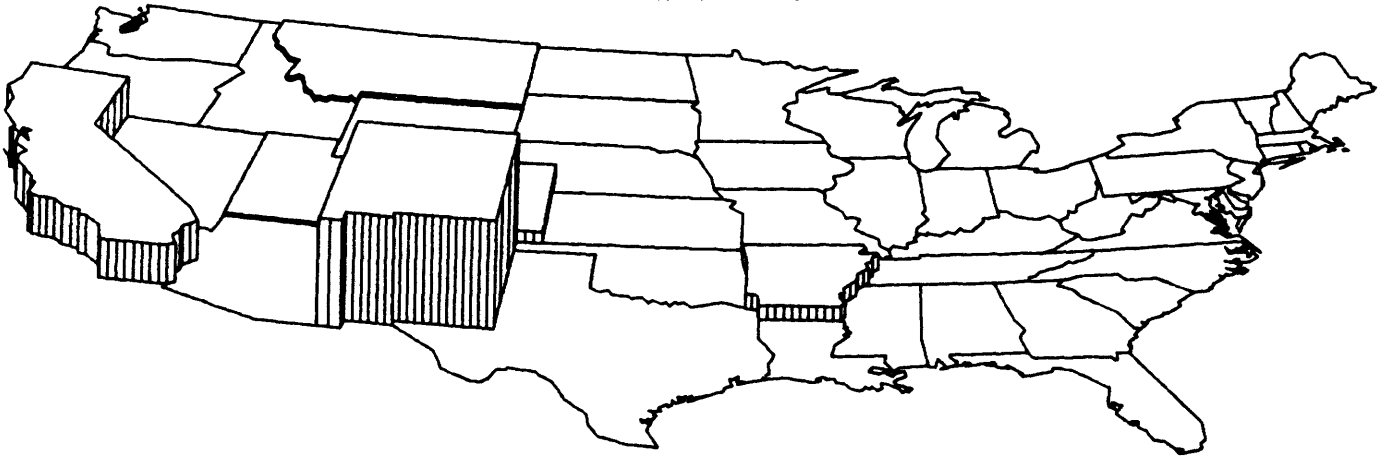


FIGURE 11.—Perspective view from the south of gas production in 1968 for the selected States.

FIGURE 8.—(Above left). Plot showing gas production in 1968 for the selected States, plus Wyoming, with class intervals and default values incorporated.

FIGURE 9.—(Below left). Plot showing interactive enlargement of part of figure 8.

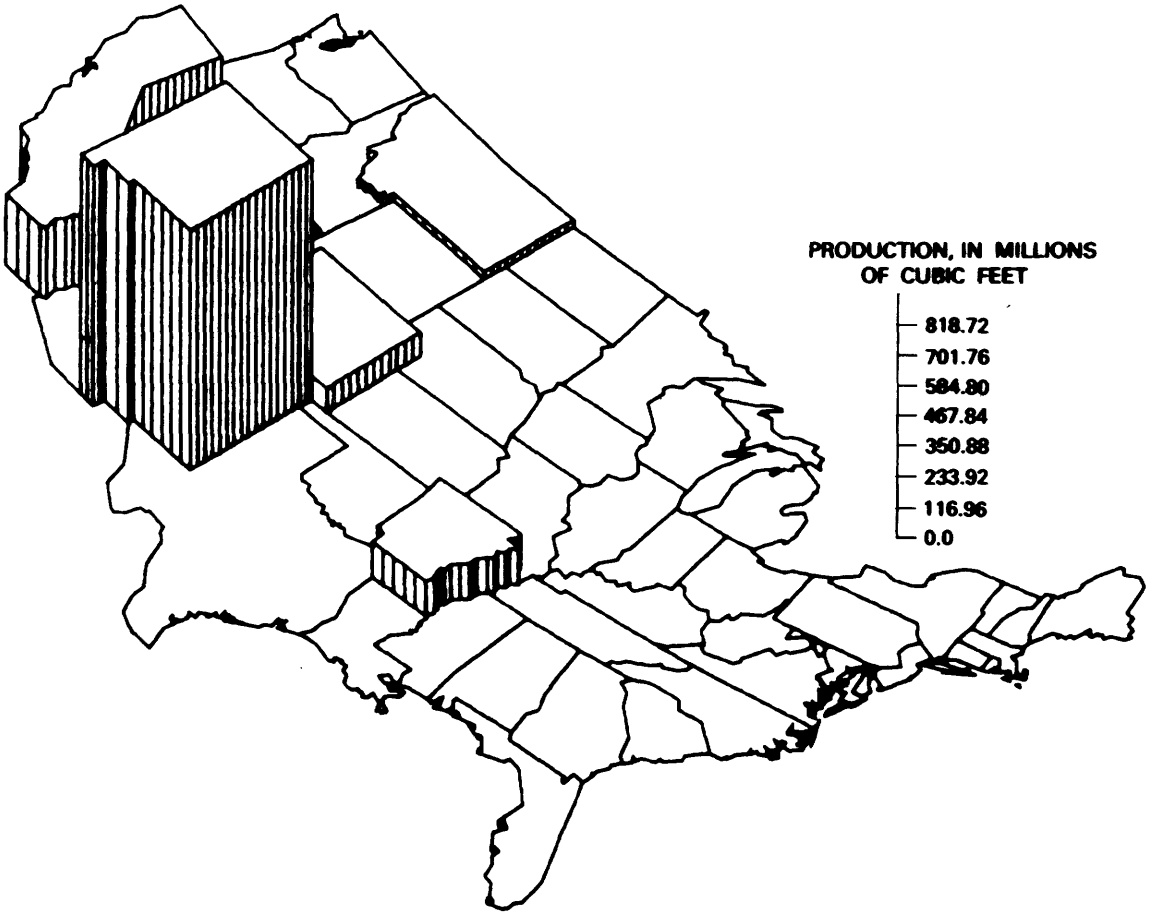


FIGURE 12.–Plot showing perspective view from the southeast of gas production in 1972 for selected States.

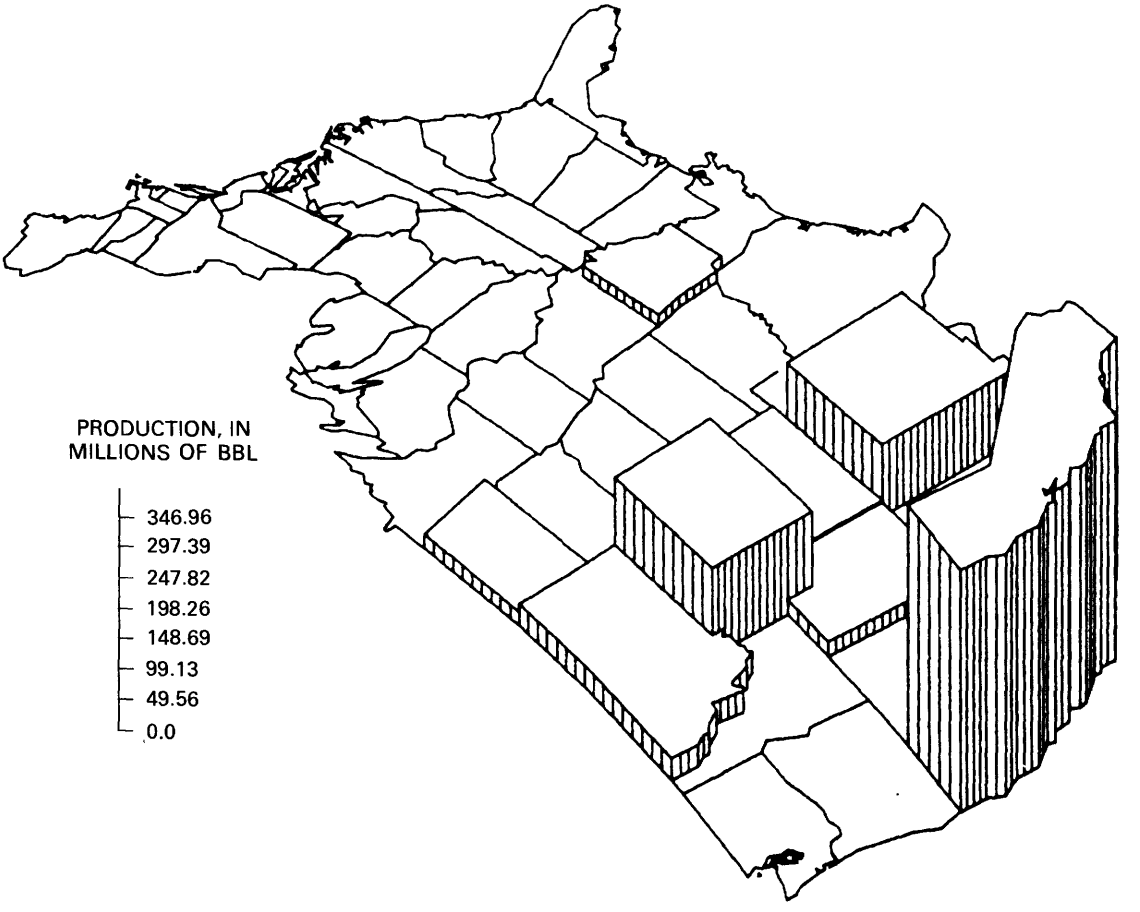


FIGURE 13. – Plot showing a perspective view from the northwest of oil production in 1971 for selected States.

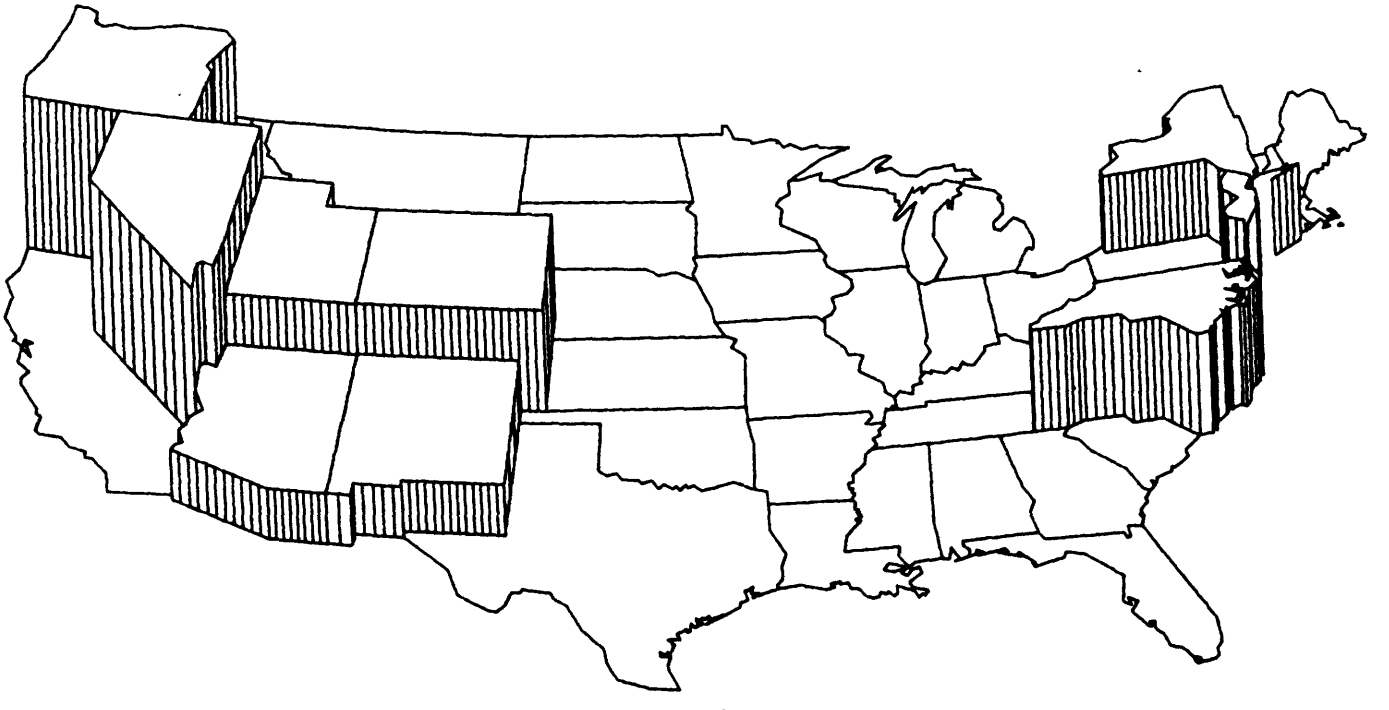


FIGURE 14. – Plot showing perspective view from the south of simulated geologic provinces.

APPENDIXES: DATA FILES AND SOFTWARE DOCUMENTATION

APPENDIX A. LIST OF AND FORMATS FOR PERMANENT AND TEMPORARY DATA FILES

APPENDIX B. OPERATIONAL INSTRUCTIONS

APPENDIX C. COMPUTER-PROGRAM REFERENCE

Note:
Program and procedure and subroutine names are printed in bold sans-serif type: **ogquesre.**
Variable names are printed in italic san-serif type: *FILE.*
Permanent-file names are printed in sans-serif type: OG$\alpha\beta$CDNAME.
Ordinary variables are printed in italics: x, y.

APPENDIX A. LIST OF AND FORMATS FOR PERMANENT AND TEMPORARY DATA FILES

LIST OF PERMANENT AND TEMPORARY DATA FILES

File	IBM name	Multics name	Permanent P Temporary T	Units	Format
Oil field and gas field outlines–original data from tape	OG.αβOG.OLD	αβog	P	decimal degree	1
Oil field and gas field outlines and header cards	OG.αβOG	αβog_str	P	do	2
County outlines in CAM 22 byte format	*OG.CAM22.αβCR*		T	radians	3
Oil field outlines in CAM 22 byte format	*OG.CAM22.αβO*		T	do	3
Gas field outlines in CAM 22 byte format	*OG.CAM22.αβG*		T	do	3
Oil field names and codes for CAM symbol plot	*OG.CAMSYM.αβOFCNM*		T	degree, min, s.	4
Gas field names and codes for CAM symbol plot	*OG.CAMSYM.αβGFCNM*		T	do	4
Oil field name only for CAM symbol plot	*OG.CAMSYM.αβONM*		T	do	5
Gas field name only for CAM symbol plot	*OG.CAMSYM.αβGNM*		T	do	5
Oil field numeric code only for CAM symbol plot	*OG.CAMSYM.αβOFC*		T	do	6
Gas field numeric code only for CAM symbol plot	*OG.CAMSYM.αβGFC*		T	do	6
County outlines and header cards	OG.αβCR	curdNM	P	radians	7
Original data for county names					8
County names	OG.αβCDNAME	αβcdname	P	decimal degrees	9
County names for CAM symbol plot	*OG.CAMSYM.αβCD*		T	deg, min, sec.	10
Input card to CAMFMT					11
Geologic basins	OG.GEOBSN	oggeobsn	P	radians	12

Format 1.–*Oil field and gas field outlines–original data from tape*

[Data sets: OG.αβOG. OLD]

Field No.	Format	Description	Column position	Field length
TYPE 1 card:				
1	2A1	State numeric FIPS[1] Code	1–2	2
2	A1	"–"	3	1
3	6A1	Field code	4–9	6
4	A1	"O" for oil, "G" for gas	10	1
5	5A1	Sequence number	11–15	5
6	1X	Blank	16	1
7	2A1	State numeric FIPS[1] code	17–18	2
8	A1	"–"	19	1
9	6A1	Field code	20–25	6
10	A1	","	26	1
11	A1	"O" for oil, "G" for gas	27	1
12	A1	","	28	1
13	F6.3	Latitude of field center	29–34	6
14	A1	","	35	1
15	F7.3	Longitude of field center	36–42	7
16	A1	","	43	1
17	nA1	Field name (variable length, maximum of 49 characters)	44 to 143+n	n
18	A1	","	(44+n)	1
19	Im	Number of latitude/longitude coordinate pairs in outline (variable length)	(45+n) to (44+n+m)	m
20	A1	","	(45+n+m)	1
21	pA1	Blank	(46+n+m) to 96	(51–n–m)
TYPE 2 card (As many *TYPE 2* cards as necessary follow a *TYPE 1* card to complete the outline.):				
1	16A1	Same as fields 1–6, *TYPE 1* card	1–16	16
2	F6.3	Latitude	17–22	6
3	A1	","	23	1
4	F7.3	Longitude	24–30	7
5	A1	","	31	1
6	___	Same as fields 2–5	32–46	15
7	___	do	47–61	15
8	___	do	62–76	15
9	___	do	77–91	15
10	5A1	Blank	92–96	5

[1]Federal Information Processing Standards.

```
05-X00001000001 05-X00001,0,33.113,092.446,BURNSIDE CREEK,32,
05-X00001000002 33.108,092.444,33.107,092.445,33.107,092.445,33.106,092.446,33.106,092.448,
05-X00001000003 33.106,092.449,33.107,092.451,33.107,092.452,33.109,092.453,33.111,092.454,
05-X00001000004 33.113,092.454,33.115,092.453,33.117,092.452,33.118,092.450,33.119,092.448,
05-X00001000005 33.119,092.446,33.119,092.444,33.118,092.442,33.117,092.441,33.116,092.440,
05-X00001000006 33.114,092.440,33.113,092.440,33.112,092.440,33.110,092.441,33.110,092.441,
05-X00001000007 33.109,092.441,33.109,092.442,33.108,092.443,33.108,092.443,33.108,092.443,
05-X00001000008 33.108,092.443,33.108,092.444
05-X00002000009 05-X00002,0,33.097,092.604,CALVARY CHURCH,05,
05-X00002000010 33.090,092.599,33.090,092.614,33.103,092.614,33.104,092.599,33.090,092.599
05-X00003000011 05-X00003,0,33.516,093.357,BURDELL BRANCH,31,
05-X00003000012 33.512,093.357,33.512,093.357,33.512,093.358,33.512,093.358,33.513,093.359,
05-X00003000013 33.513,093.359,33.514,093.359,33.515,093.359,33.516,093.359,33.516,093.359,
05-X00003000014 33.517,093.359,33.518,093.358,33.518,093.357,33.518,093.356,33.518,093.355,
05-X00003000015 33.518,093.354,33.517,093.353,33.516,093.352,33.515,093.351,33.514,093.351,
05-X00003000016 33.513,093.351,33.512,093.352,33.512,093.352,33.512,093.352,33.511,093.353,
05-X00003000017 33.511,093.354,33.511,093.355,33.512,093.355,33.512,093.356,33.512,093.356,
05-X00003000018 33.512,093.357
05-X00004000019 05-X00004,0,33.395,093.740,GALILEE CHURCH,05,
05-X00004000020 33.392,093.734,33.392,093.743,33.400,093.743,33.400,093.734,33.392,093.734
05-004597G00616 05-004597,G,35.383,093.819,AETNA,170,
05-004597G00617 35.328,093.810,35.329,093.846,35.344,093.845,35.348,093.912,35.358,093.912,
05-004597G00618 35.359,093.906,35.372,093.905,35.375,093.898,35.386,093.898,35.390,093.895,
05-004597G00619 35.390,093.880,35.410,093.876,35.411,093.869,35.410,093.840,35.413,093.840,
05-004597G00620 35.418,093.840,35.423,093.840,35.427,093.839,35.429,093.839,35.430,093.837,
05-004597G00621 35.431,093.832,35.431,093.827,35.431,093.825,35.433,093.824,35.435,093.822,
05-004597G00622 35.437,093.818,35.437,093.814,35.437,093.809,35.437,093.804,35.437,093.804,
05-004597G00623 35.440,093.805,35.443,093.805,35.446,093.804,35.448,093.804,35.450,093.804,
05-004597G00624 35.449,093.803,35.449,093.778,35.435,093.778,35.432,093.743,35.419,093.743,
05-004597G00625 35.419,093.744,35.419,093.748,35.419,093.753,35.419,093.756,35.419,093.758,
05-004597G00626 35.418,093.759,35.416,093.760,35.413,093.761,35.411,093.761,35.409,093.761,
05-004597G00627 35.407,093.761,35.406,093.763,35.406,093.766,35.406,093.769,35.406,093.772,
05-004597G00628 35.405,093.776,35.403,093.777,35.399,093.777,35.395,093.777,35.391,093.777,
05-004597G00629 35.387,093.775,35.387,093.772,35.388,093.768,35.388,093.764,35.388,093.759,
05-004597G00630 35.387,093.755,35.387,093.750,35.386,093.746,35.387,093.743,35.388,093.742,
05-004597G00631 35.392,093.741,35.396,093.742,35.399,093.742,35.401,093.741,35.403,093.740,
05-004597G00632 35.403,093.739,35.403,093.735,35.403,093.730,35.403,093.727,35.402,093.726,
05-004597G00633 35.397,093.726,35.392,093.726,35.388,093.725,35.387,093.722,35.384,093.713,
05-004597G00634 35.384,093.704,35.384,093.696,35.384,093.689,35.384,093.685,35.383,093.683,
05-004597G00635 35.381,093.682,35.376,093.683,35.372,093.683,35.371,093.683,35.370,093.686,
05-004597G00636 35.370,093.691,35.370,093.697,35.370,093.704,35.371,093.708,35.371,093.713,
05-004597G00637 35.371,093.717,35.371,093.721,35.371,093.722,35.369,093.724,35.366,093.726,
05-004597G00638 35.364,093.728,35.362,093.730,35.361,093.733,35.361,093.736,35.360,093.739,
05-004597G00639 35.360,093.740,35.358,093.740,35.356,093.741,35.353,093.742,35.351,093.743,
05-004597G00640 35.349,093.744,35.348,093.747,35.348,093.751,35.348,093.755,35.348,093.758,
05-004597G00641 35.348,093.761,35.348,093.764,35.347,093.767,35.348,093.770,35.349,093.772,
05-004597G00642 35.351,093.773,35.353,093.773,35.355,093.774,35.356,093.774,35.357,093.776,
05-004597G00643 35.357,093.778,35.357,093.779,35.357,093.781,35.356,093.783,35.355,093.786,
05-004597G00644 35.354,093.787,35.355,093.789,35.357,093.790,35.359,093.790,35.360,093.791,
05-004597G00645 35.361,093.792,35.361,093.794,35.360,093.797,35.360,093.800,35.358,093.802,
05-004597G00646 35.356,093.802,35.354,093.801,35.353,093.801,35.352,093.800,35.351,093.798,
05-004597G00647 35.351,093.797,35.350,093.795,35.350,093.794,35.347,093.793,35.344,093.793,
05-004597G00648 35.340,093.793,35.336,093.794,35.334,093.794,35.331,093.795,35.329,093.796,
05-004597G00649 35.329,093.797,35.329,093.800,35.329,093.803,35.329,093.805,35.329,093.807,
05-004597G00650 35.329,093.809,35.328,093.810,35.328,093.810,35.328,093.810,35.328,093.810
05-004597000021 05-004597,0,35.386,093.827,AETNA,144,
05-004597000022 35.349,093.742,35.349,093.771,35.349,093.771,35.352,093.771,35.354,093.772,
05-004597000023 35.356,093.773,35.356,093.774,35.357,093.778,35.358,093.780,35.358,093.782,
05-004597000024 35.357,093.783,35.356,093.785,35.355,093.787,35.356,093.788,35.357,093.789,
```

Format 2.—*Oil field and gas field outlines and header cards*

[Data set: OG.αβOG]

[Note that fields 1–14, *TYPE 1*, cards and all fields of *TYPE 2*, cards are identical on the oil field and gas field coordinate data set in its original format (format 1)]

Field No.	Format	Description	Column position	Field length
TYPE 1 card:				
1	2A1	State numeric FIPS[1] Code	1–2	2
2	A1	"–"	3	1
3	6A1	Field code	4–9	6
4	A1	"O" for oil, "G" for gas	10	1
5	5A1	Sequence number	11–15	5
6	1X	Blank	16	1
7	2A1	State numeric FIPS[1] code	17–18	2
8	A1	"–"	19	1
9	6A1	Field code	20–25	6
10	A1	","	26	1
11	A1	"O" for oil, "G" for gas	27	1
12	A1	","	28	1
13	F6.3	Latitude of field center	29–34	6
14	A1	","	35	1
15	F7.3	Longitude of field center	36–42	7
16	A1	","	43	1
17	I5	Number of latitude/longitude coordinate pairs in outline	44–48	5
18	A1	","	49	1
19	I2	Number of characters in field name	50–51	2
20	A1	","	52	1
21	nA1	Field name (variable length, maximum of 44 characters)	53 to (52+n)	n

TYPE 2 card (As many *TYPE 2* cards as necessary follow a *TYPE 1* card to complete the outline; there are five latitude/longitude coordinate pairs per line.):

1	16A1	Same as fields 1–6, *TYPE 1* card	1–16	16
2	F6.3	Latitude	17–22	6
3	A1	","	23	1
4	F7.3	Longitude	24–30	7
5	A1	","	31	1
6	___	Same as fields 2–5	32–46	15
7	___	_____do_____	47–61	15
8	___	_____do_____	62–76	15
9	___	_____do_____	77–91	15
10	5A1	Blank	92–96	5

[1]Federal Information Processing Standards.

```
05-X00001000001 05-X00001,O,33.113,092.446,00032,14,BURNSIDE CREEK
05-X00001000002 33.108,092.444,33.107,092.445,33.107,092.445,33.106,092.446,33.106,092.448,
05-X00001000003 33.106,092.449,33.107,092.451,33.107,092.452,33.109,092.453,33.111,092.454,
05-X00001000004 33.113,092.454,33.115,092.453,33.117,092.452,33.118,092.450,33.119,092.448,
05-X00001000005 33.119,092.446,33.119,092.444,33.118,092.442,33.117,092.441,33.116,092.440,
05-X00001000006 33.114,092.440,33.113,092.440,33.112,092.440,33.110,092.441,33.110,092.441,
05-X00001000007 33.109,092.441,33.109,092.442,33.108,092.443,33.108,092.443,33.108,092.443,
05-X00001000008 33.108,092.443,33.108,092.444
05-X00002000009 05-X00002,O,33.097,092.604,00005,14,CALVARY CHURCH
05-X00002000010 33.090,092.599,33.090,092.614,33.103,092.614,33.104,092.599,33.090,092.599
05-X00003000011 05-X00003,O,33.516,093.357,00031,14,BURDELL BRANCH
05-X00003000012 33.512,093.357,33.512,093.357,33.512,093.358,33.512,093.358,33.513,093.359,
05-X00003000013 33.513,093.359,33.514,093.359,33.515,093.359,33.516,093.359,33.516,093.359,
05-X00003000014 33.517,093.359,33.518,093.356,33.518,093.357,33.518,093.356,33.518,093.355,
05-X00003000015 33.518,093.354,33.517,093.353,33.516,093.352,33.515,093.351,33.514,093.351,
05-X00003000016 33.513,093.351,33.512,093.352,33.512,093.352,33.512,093.352,33.511,093.353,
05-X00003000017 33.511,093.354,33.511,093.355,33.512,093.355,33.512,093.356,33.512,093.356,
05-X00003000018 33.512,093.357
05-X00004000019 05-X00004,O,33.392,093.740,00005,14,GALILEE CHURCH
05-X00004000020 33.392,093.734,33.392,093.747,33.400,093.743,33.400,093.734,33.392,093.734
05-00459700G619 05-00459700,G,35.383,093.819,00170,05,AETNA
```

```
05-0045976000617 35.328,093.810,35.329,093.846,35.344,093.845,35.346,093.912,35.358,093.912,
05-0045976000618 35.359,093.906,35.372,093.905,35.375,093.898,35.386,093.898,35.390,093.895,
05-0045976000619 35.390,093.860,35.410,093.876,35.411,093.869,35.410,093.840,35.413,093.840,
05-0045976000620 35.418,093.840,35.423,093.840,35.427,093.839,35.429,093.839,35.430,093.837,
05-0045976000621 35.431,093.832,35.431,093.827,35.431,093.825,35.433,093.824,35.435,093.822,
05-0045976000622 35.437,093.818,35.437,093.814,35.437,093.809,35.437,093.804,35.437,093.804,
05-0045976000623 35.440,093.805,35.443,093.805,35.445,093.804,35.448,093.804,35.450,093.804,
05-0045976000624 35.445,093.803,35.449,093.776,35.435,093.776,35.432,093.743,35.419,093.743,
05-0045976000625 35.419,093.744,35.419,093.749,35.419,093.753,35.419,093.756,35.419,093.758,
05-0045976000626 35.419,093.758,35.416,093.760,35.413,093.761,35.411,093.761,35.409,093.761,
05-0045976000627 35.407,093.761,35.406,093.763,35.406,093.766,35.406,093.769,35.406,093.772,
05-0045976000628 35.405,093.776,35.403,093.777,35.399,093.777,35.395,093.777,35.391,093.777,
05-0045976000629 35.387,093.775,35.387,093.772,35.388,093.758,35.388,093.754,35.388,093.759,
05-0045976000630 35.387,093.755,35.387,093.750,35.386,093.746,35.387,093.743,35.388,093.742,
05-0045976000631 35.392,093.741,35.396,093.742,35.399,093.742,35.401,093.741,35.403,093.740,
05-0045976000632 35.403,093.739,35.403,093.735,35.403,093.730,35.403,093.727,35.402,093.726,
05-0045976000633 35.397,093.726,35.392,093.726,35.388,093.725,35.387,093.722,35.384,093.719,
05-0045976000634 35.384,093.704,35.384,093.696,35.384,093.689,35.384,093.685,35.383,093.683,
05-0045976000635 35.381,093.682,35.376,093.683,35.372,093.683,35.371,093.693,35.370,093.686,
05-0045976000636 35.370,093.691,35.370,093.697,35.370,093.704,35.371,093.708,35.371,093.713,
05-0045976000637 35.371,093.717,35.371,093.721,35.371,093.722,35.369,093.724,35.366,093.726,
05-0045976000638 35.364,093.726,35.362,093.730,35.361,093.733,35.361,093.736,35.360,093.739,
05-0045976000639 35.360,093.740,35.359,093.740,35.356,093.741,35.353,093.742,35.351,093.743,
05-0045976000640 35.349,093.744,35.348,093.747,35.348,093.751,35.348,093.755,35.348,093.758,
05-0045976000641 35.348,093.761,35.348,093.764,35.347,093.767,35.348,093.770,35.349,093.772,
05-0045976000642 35.351,093.773,35.353,093.777,35.355,093.774,35.356,093.774,35.357,093.776,
05-0045976000643 35.357,093.778,35.357,093.779,35.357,093.781,35.356,093.783,35.355,093.786,
05-0045976000644 35.354,093.787,35.353,093.784,35.357,093.790,35.359,093.790,35.360,093.791,
05-0045976000645 35.361,093.792,35.361,093.794,35.360,093.797,35.360,093.800,35.358,093.802,
```

Format 3.–*CAM22 byte data for oil fields and gas fields and counties*

[Data sets: *OG.CAM22.αβCR, OG.CAM22.αβO, OG.CAM22.αβG*]

[Note that a 4-byte IBM control word precedes each of these records]

Field No.	Type	Description	Length in bytes
1	REAL	Line identifier–pen up, pen down control	4
2	INTEGER*2	Rank or class–set to zero	2
3	REAL	Latitude in radians–positive for north latitude	4
4	REAL	Longitude in radians–negative for west longitude	4
5	REAL	Sequence count	4

```
00000001000004093ED90C119D0B300000001  00000001000004093FA25C119D0B300000011
00000001000004093EC69C119D0C500000002  00000001000004093F8FFC119D08E00000012
00000001000004093EC69C119D0C500000003  00000001000004093F7DCC119D07C00000013
00000001000004093EB47C119D0D700000004  00000001000004093F655C119D06A00000014
00000001000004093EB47C119D0FC00000005  00000001000004093F46CC119D06A00000015
00000001000004093EB47C119D10E00000006  00000001000004093F34AC119D06A00000016
00000001000004093EC69C119D13300000007  00000001000004093F223C119D06A00000017
00000001000004093EC69C119D14500000008  00000001000004093EFD9C119D07C00000018
00000001000004093EEB2C119D15800000009  00000001000004093EFD9C119D07C00000019
00000001000004093F0FCC119D16A0000000A  00000001000004093EEB2C119D07C0000001A
00000001000004093F34AC119D16A0000000B  00000001000004093EEB2C119D08E0000001B
00000001000004093F593C119D1580000000C  00000001000004093ED90C119D0A0000001C
00000001000004093F7DCC119D1450000000D  00000001000004093ED90C119D0A0000001D
00000001000004093F8FFC119D1210000000E  00000001000004093ED90C119D0A0000001F
00000001000004093FA25C119D0FC0000000F  00000001000004093ED90C119D0A0000001F
00000001000004093FA25C119D0D700000010  00000001000004093ED90C119D0B300000020
```

Format 4.—*Oil field and gas field names and codes for CAM symbol plot*

[Data sets: OG.CAMSYM.αβOFCNM, OG.CAMSYM.αβGFCNM]

Field No.	Format	Description	Column position	Field length
1	I2	Number of characters to plot from next 40 columns	1–2	2
2	2A1	"+ "—Places a cross on field center	3–4	2
3	6A1	Field code	5–10	6
4	A1	Blank—Places a blank between field code and name	11	1
5	44A1	Field name, left justified	12–55	44
6	4X	Blank	56–59	4
7	I2	Latitude—degrees	60–61	2
8	I2	—minutes	62–63	2
9	I2	—seconds	64–65	2
10	A1	"N" for north	66	1
11	I3	Longitude—degrees	67–69	3
12	I2	—minutes	70–71	2
13	I2	—seconds	72–73	2
14	A1	"W" for west	74	1
15	6X	Blank	75–80	6

```
19+ 061129 BIG BRANCH                    332252N 933256W
18+ 06140P BIG CREEK                     331641N 931655W
15+ 077529 BODCAW                        333144N 932411W
21+ 077932 BOGGY BOTTOM                  33 838N 923625W
20+ 077963 BOGGY CREEK                   331550N 935638W
19+ 078438 BOIS D'ARC                    333050N 933918W
16+ 059803 BEVERLY                       352230N 94 412W
18+ 06140P BIG CREEK                     3317 2N 931648W
14+ 071623 BLICK                         352118N 924218W
16+ 074227 BLOOMER                       351923N 94 810W
```

Format 5.—*Oil field and gas field names only for CAM symbol plot*

[Data sets: OG.CAMSYM.αβONM, OG.CAMSYM.αβGNM]

Field No.	Format	Description	Column position	Field length
1	I2	Number of characters to plot from next 40 columns	1–2	2
2	2A1	"+ "—Places a cross on field center	3–4	2
3	44A1	Field name, left justified	5–48	44
4	11X	Blank	49–59	11
5	I2	Latitude—degrees	60–61	2
6	I2	—minutes	62–63	2
7	I2	—seconds	64–65	2
8	A1	"N" for north	66	1
9	I3	Longitude—degrees	67–69	3
10	I2	—minutes	70–71	2
11	I2	—seconds	72–73	2
12	A1	"W" for west	74	1
13	6X	Blank	75–80	6

```
 7+ AETNA                                352259N 9349 8W
 6+ ALMA                                 352834N 941218W
12+ ALMA  EAST                           352942N 941124W
 7+ ALTUS                                352646N 934359W
```

```
16+ BURNSIDE CREEK                                  33 647N 922646W
16+ CALVARY CHURCH                                  33 549N 923614W
16+ BURDELL BRANCH                                  333058N 932125W
16+ GALILEE CHURCH                                  332342N 934424W
 7+ AETNA                                           352310N 934937W
10+ ARTESIAN                                        332440N 922848W
```

Format 6.–*Oil field and gas field numeric code only for CAM symbol plot*

[Data sets: *OG.CAMSYM.αβOFC, OG.CAMSYM.αβGFC*]

Field No.	Format	Description	Column position	Field length
1	I2	" 8"–the number of characters to plot from the next 40 columns	1–2	2
2	2A1	"+"–Places a cross on field center	3–4	2
3	6A1	Field code	5–10	6
4	49X	Blank	11–59	49
5	I2	Latitude–degrees	60–61	2
6	I2	–minutes	62–63	2
7	I2	–seconds	64–65	2
8	A1	"N" for north	66	1
9	I3	Longitude–degrees	67–69	3
10	I2	–minutes	70–71	2
11	I2	–seconds	72–73	2
12	A1	"W" for west	74	1
13	6X	Blank	75–80	6

```
 8+ 043473                                          353748N 933640W
```

```
 8+ 029506                                          33 958N 93 622W
 8+ 029537                                          33 611N 93 043W
 8+ 030808                                          33 224N 922346W
 8+ 039510                                          331858N 931442W
 8+ 048643                                          332031N 925317W
 8+ 051588                                          33 7 1N 924529W
 8+ 055835                                          331258N 9218 0W
 8+ 058191                                          33 524N 922017W
 8+ 059214                                          33 6 4N 922953W
```

Format 7.–*County outlines and header cards.*

[Data set: OG.αβCR]

Field No.	Format	Description	Column position	Field length
Header card:				
1	I5	County numeric FIPS[1] code	1–5	5
2	I5	Number of "closed curves" in this outline	6–10	5
3	I5	Bordering county, State, or country	11–15	5
4	I5	Number of latitude/longitude coordinates in this outline	16–20	5
5	I5	Unused (blank)	21–25	5
6	15	State numeric FIPS[1] code	26–30	5
7	I5	992–Identifies outlines as county outlines	31–35	5
8	I5	Unused (blank)	36–40	5
9	32X	Blank	41–72	32
10	8A1	Card sequence number	73–80	8
Coordinate data cards (As many of these as necessary follow header card to complete the outline.):				
1	1X	Blank	1	1
2	I11	Longitude in radians (negative for West longitude). Decimal point assumed between cc3 and cc4.	2–12	11
3	1X	Blank	13	1
4	I11	Latitude (radians). Decimal point assumed between cc15 and cc16.	14–24	11
5	1X	Blank	25	1
6	I11	Longitude (radians). Decimal point assumed between cc27 and cc28.	26–36	11
7	1X	Blank	37	1
8	I11	Latitude (radians). Decimal point assumed between cc39 and cc40.	38–48	11
9	1X	Blank	49	1
10	I11	Longitude (radians). Decimal point assumed between cc51 and cc52.	50–60	11
11	1X	Blank	61	1
12	I11	Latitude (radians). Decimal point assumed between cc63 and cc64.	62–72	11
13	8A1	Card sequence number.	73–80	8

[1]Federal Information Processing Standards.

```
   7     1  143     9     0     5  992     0                                    00000296
-1645839870   0632830333 -1650215093  0630128369 -1647370266  0630085815 00000297
-1647384626   0630840362 -1646424324  0630628193 -1646414990  0632112069 00000298
-1640785509   0631966118 -1640768779  0632450798 -1638637618  0632451233 00000299
 143     1  905     3     0     5  992     0                                    00000300
-1648707899   0628155797 -1650200052  0630124673 -1649204547  0624173406 00000301
 143     1   33     5     0     5  992     0                                    00000302
-1646213362   0625044740 -1649199602  0624173516 -1642717108  0623975980 00000303
-1642694895   0624238689 -1639957386  0624187604                              00000304
  33     1  905     3     0     5  992     0                                    00000305
-1647342533   0622272927 -1649214031  0624161088 -1648136543  0617833743 00000306
  33     1  131    59     0     5  992     0                                    00000307
-1646158539   0619150680 -1648156487  0617841377 -1648078203  0618008489 00000308
-1648009742   0618175386 -1647951369  0618350131 -1647911530  0618492197 00000309
-1647851668   0618618563 -1647805294  0618708324 -1647619354  0618748642 00000310
-1647430565   0618700246 -1647235712  0618615652 -1647060960  0618542715 00000311
-1646365392   0618433895 -1646728293  0618303653 -1646624663  0618140427 00000312
-1646640053   0618002939 -1646693919  0617836403 -1646683311  0617671214 00000313
-1646521361   0617563605 -1646480088  0617453613 -1646390538  0617423214 00000314
-1646299233   0617336370 -1646266096  0617220073 -1646165507  0617153585 00000315
-1645936023   0617049425 -1645894239  0616975779 -1645224575  0616955081 00000316
-1644969232   0615972349 -1644645878  0617019183 -1644396746  0617080624 00000317
-1644221125   0617132512 -1644108719  0617158936 -1644051001  0617204446 00000318
-1644049512   0617321459 -1644033266  0617438763 -1643929376  0617606187 00000319
-1643798330   0617653138 -1643689383  0617631061 -1643619215  0617588054 00000320
-1643419974   0617507209 -1643227182  0617478660 -1643021943  0617531009 00000321
-1642891863   0617630310 -1642869899  0617723510 -1642889250  0617834499 00000322
```

```
-1642997541  0618055892 -1643133431  0618158172 -1643305192  0618308161 00000323
-1643382637  0618431723 -1643318475  0618598367 -1643095328  0618719644 00000324
-1642948175  0618742631 -1642765975  0618746103 -1642552747  0618697700 00000325
-1642363815  0618636715 -1642229967  0618606953 -1642126565  0618608857 00000326
-1642042971  0518614481 -1641836500  0618630419                        00000327
  47     1    33    40     0     5  992    0                           00000328
-1639927425  0622295761 -1639042082  0624187254 -1639043179  0623650583 00000329
-1639968414  0523666893 -1639995213  0623182231 -1640266778  0623169385 00000330
-1640305929  0622410145 -1641832670  0622410783 -1641899517  0620904604 00000331
-1641035422  0620888147 -1641051560  0620573166 -1641177694  0620494226 00000332
-1641225248  0620428811 -1641277179  0620343143 -1641293711  0620225844 00000333
-1641286362  0620137222 -1641239249  0620037222 -1641173194  0619955805 00000334
-1641137267  0519914011 -1641109451  0619797522 -1641112738  0619736948 00000335
-1641154793  0619651465 -1641216881  0619577720 -1641238885  0619480498 00000336
-1641236690  0619399854 -1641239751  0619331217 -1641375495  0619248056 00000337
-1641458692  0619226363 -1641501702  0619177164 -1641436228  0619125920 00000338
-1641351717  0519099227 -1641266770  0619056402 -1641230974  0619038644 00000339
-1641249030  0518947804 -1641315785  0618865904 -1641396076  0618811957 00000340
-1541519645  0618753252 -1641617470  0618727248 -1641705330  0618697390 00000341
-1641941759  0618642424                                                 00000342
 131     1    47     8     0     5  992    0                           00000343
-1643942901  0616032745 -1641841871  0618646456 -1641966843  0614630652 00000344
-1641661608  0514604054 -1641650062  0614362278 -1641389725  0614359007 00000345
-1641371550  0514593262 -1641037753  0614599355                        00000346
  47     1    33    22     0     5  992    0                           00000347
-1639401071  0617415458 -1641052696  0614607150 -1641011430  0614898289 00000348
-1639188564  0514882114 -1639129130  0616403682 -1637293277  0616341540 00000349
-1637250501  0618036312 -1637209959  0617988562 -1637098578  0617865312 00000350
-1636947954  0617738655 -1636798793  0617676500 -1636664503  0617618132 00000351
-1636544700  0517547425 -1636438913  0617444722 -1636337685  0617324808 00000352
-1636237693  0617257807 -1636128821  0617231273 -1636005657  0617225123 00000353
-1635873400  0517251374 -1635761000  0617285376 -1635644136  0617339608 00000354
```

Format 8.–*Original data for county names*

[Data set: Card deck of map measurements]

Field No.	Field name	Format	Description	Column position	Field length
Card 1:					
1	STATE	I3	State numeric FIPS[1] code	1–3	3
2		IX	Blank	4	1
3	A	F5.0*	Distance between meridians, top of map	5–9	5
4	B	F5.0	Distance between meridians, bottom of map	10–14	5
5	C	F5.0	Distance between parallels	15–19	5
6	D	F5.0	Distance from bottom border to southernmost latitude	20–24	5
7	H	F5.0	Height of map	25–29	5
8	L	F5.0	Southernmost latitude	30–34	5
9	PRNT	A1	If nonblank, causes printing of output	35	1
10		45X	Blank	36–80	45
Subsequent cards:					
1	FIPS	F5.0*	County numeric FIPS[1] code	1–5	5
2	LAT	F4.0	Reference latitude	6–9	4
3	MLAT	F6.0	Distance from point P to latitude	10–15	6
4	LONG	F5.0	Reference longitude	16–20	5
5	MLONG	F6.0	Distance from point P to longitude	21–26	6
6	CTY	54A1	County name	27–80	54

[1]Federal Information Processing Standards.

*As usual in FORTRAN, F5.0, for example, will accept any number requiring up to five card columns, and may have non-zero digits to the right of the decimal point.

```
  8 167. 176.5 221.25.25 933. 37. X
1.    40. -31.  104. -109. ADAMS
3.    37. 124.  106.  2.   ALAMOSA
5.    40. -76.  104. -114. ARAPAHOE
7.    37. 42.   107. -40.  ARCHULETA
9.    37. 75.   103.  41.  BACA
11.   38. -5.   103. -78.  BENT
13.   40. 20.   106.  88.  BOULDER
15.   39. -51.  106. -63.  CHAFFEE
17.   39. -35.  103.  23.  CHEYENNE
19.   40. -66.  106.  28.  CLEAR CREEK
21.   37. 43.   106. -68.  CONEJOS
23.   37. 60.   106.  69.  COSTILLA
25.   38. 78.   104.  6.   CROWLEY
27.   38. 24.   106.  81.  CUSTER
29.   39. -41.  108. -10.  DELTA
31.   40. -60.  105.       DENVER
33.   38. -52.  109.  33.  DOLORES
35.   39. 73.   105. -22.  DOUGLAS
37.   40. -74.  107.  28.  EAGLE
39.   39. 70.   104. -62.  ELBERT
41.   39. -42.  104. -120. EL PASO
43.   38. -112. 105.  48.  FREMONT
45.   40. -91.  108. -72.  GARFIELD
47.   40. -29.  106.  67.  GILPIN
49.   40. 21.   106. -60.  GRAND
51.   38. -60.  107. -50.  GUNNISON
53.   38. -33.  107. -84.  HINSDALE
55.   37. 152.  105. -43.  HUERFANO
57.   41. -79.  107.  82.  JACKSON
59.   40. -94.  105. -63.  JEFFERSON
61.   38. 91.   103. -5.   KIOWA
63.   39. 73.   103.  16.  KIT CARSON
65.   39. 45.   106. -73.  LAKE
67.   37. 75.   108. -10.  LA PLATA
69.   41. -77.  105.  56.  LARIMER
71.   37. 75.   104. -69.  LAS ANIMAS
73.   39. -10.  104.  58.  LINCOLN
75.   41. -43.  103. -46.  LOGAN
77.   39. 1.    109.  51.  MESA
79.   38. -74.  107. -7.   MINERAL
81.   41. -74.  109.  80.  MOFFAT
83.   37. 80.   109.  23.  MONTEZUMA
85.   38. 82.   109.  62.  MONTROSE
87.   40. 53.   104.  2.   MORGAN
89.   38. -16.  104.  15.  OTERO
91.   38. 29.   108.  16.  OURAY
93.   39. 34.   106.  14.  PARK
95.   41. -91.  103.  85.  PHILLIPS
97.   39. 48.   107. -20.  PITKIN
99.   38. 2.    102. -104. PROWERS
101.  38. 39.   105.  40.  PUEBLO
103.  40. -6.   109.  54.  RIO BLANCO
105.  38. -84.  107.  72.  RIO GRANDE
107.  41. -106. 108.  117. ROUTT
109.  38. 10.   106. -112. SAGUACHE
111.  38. -48.  108.  34.  SAN JUAN
113.  38.       109.  32.  SAN MIGUEL
115.  41. -27.  103.  80.  SEDGWICK
117.  40. -86.  106. -25.  SUMMIT
119.  39. -29.  105. -51.  TELLER
```

Format 9.–*County names*

Data set: OG.αβCDNAME]

Field No.	Field name	Format	Description	Column position	Field length
1	STATE	I4	State numeric FIPS[1] code	1–4	4
2		A1	","	5	1
3	IFIPS	I4	County numeric FIPS[1] code	6–9	4
4		A1	","	10	1
5	LT	F9.3	Latitude, decimal degrees	11–19	9
6		A1	","	20	1
7	LG	F9.3	Longitude, decimal degrees	21–29	9
8		A1	","	30	1
9	CHS	I4	Number of characters in county name	31–34	4
10		A1	","	35	1
11	CTY	45A1	County name, left justified	36–80	45

[1] Federal Information Processing Standards.

```
3, 1, 39.850, 104.642, 5,ADAMS
3, 3, 37.561, 105.989, 7,ALAMOSA
3, 5, 39.636, 104.670, 8,ARAPAHOE
3, 7, 37.190, 107.228, 9,ARCHULETA
3, 9, 37.339, 102.766, 4,BACA
3, 11, 37.977, 103.218, 4,BENT
3, 13, 40.090, 105.490, 7,BOULDER
3, 15, 38.769, 106.346, 7,CHAFFEE
3, 17, 39.842, 102.856, 8,CHEYENNE
3, 19, 39.701, 105.835, 11,CLEAR CREEK
3, 21, 37.195, 106.387, 7,CONEJOS
3, 23, 37.871, 105.607, 8,COSTILLA
3, 25, 38.353, 103.965, 7,CROWLEY
3, 27, 38.109, 105.534, 6,CUSTER
3, 29, 38.814, 108.058, 5,DELTA
3, 31, 39.729, 105.000, 6,DENVER
3, 33, 37.755, 108.811, 7,DOLORES
3, 35, 39.330, 105.109, 7,DOUGLAS
3, 37, 39.665, 106.836, 5,EAGLE
3, 39, 39.317, 104.363, 6,ELBERT
3, 41, 38.810, 104.749, 7,EL PASO
3, 43, 38.493, 105.722, 7,FREMONT
3, 45, 39.588, 108.432, 8,GARFIELD
3, 47, 39.869, 105.605, 6,GILPIN
3, 49, 40.095, 106.354, 5,GRAND
3, 51, 38.729, 107.290, 8,GUNNISON
3, 53, 37.851, 107.482, 8,HINSDALE
3, 55, 37.638, 105.246, 6,HUERFANO
3, 57, 40.643, 106.512, 7,JACKSON
3, 59, 39.575, 105.370, 9,JEFFERSON
3, 61, 38.412, 103.029, 6,KIOWA
3, 63, 39.330, 102.906, 10,KIT CARSON
3, 65, 39.204, 106.426, 4,LAKE
3, 67, 37.339, 108.057, 8,LA PLATA
3, 69, 40.652, 105.667, 7,LARIMER
3, 71, 37.339, 104.507, 10,LAS ANIMAS
3, 73, 38.955, 103.662, 7,LINCOLN
3, 75, 40.805, 103.274, 5,LOGAN
3, 77, 39.005, 108.703, 4,MESA
3, 79, 37.665, 107.040, 7,MINERAL
3, 81, 40.665, 108.524, 6,MOFFAT
3, 83, 37.350, 108.857, 9,MONTEZUMA
3, 85, 38.371, 108.642, 8,MONTROSE
3, 87, 40.285, 103.908, 6,MORGAN
3, 89, 37.928, 103.914, 5,OTERO
3, 91, 38.131, 107.908, 5,OURAY
3, 93, 39.154, 105.916, 4,PARK
3, 95, 40.538, 102.495, 8,PHILLIPS
3, 97, 39.217, 107.117, 6,PITKIN
3, 99, 38.009, 102.598, 7,PROWERS
```

Format 10.–*County names for CAM symbol plot*

[Data set: OG.CAMSYM.αβCD]

Field No.	Format	Description	Column position	Field length
1	I2	Number of characters to plot from next 40 columns	1–2	2
2	45A1	Name of county, left justified	3–47	45
3	12X	Blank	48–59	12
4	I2	Latitude–degrees	60–61	2
5	I2	–minutes	62–63	2
6	I2	–seconds	64–65	2
7	A1	"N" for north	66	1
8	I3	Longitude–degrees	67–69	3
9	I2	–minutes	70–71	2
10	I2	–seconds	72–73	1
11	A1	"W" for west	74	1
12	6X	Blank	75–80	6

8ARKANSAS	341811N 913744W
6ASHLEY	331156N 92 058W
6BAXTER	361456N 922544W
6BENTON	361724N 942732W
5BOONE	361616N 931218W
7BRADLEY	332725N 921630W
7CALHOUN	333358N 923755W
7CARROLL	3620 6N 934634W
6CHICOT	331634N 912349W
5CLARK	34 242N 931934W
4CLAY	362053N 9036 7W
8CLEBURNE	353040N 921059W
9CLEVELAND	335328N 922010W
8COLUMBIA	331247N 932432W
6CONWAY	351829N 924724W
9CRAIGHEAD	354850N 905541W
8CRAWFORD	353235N 942259W
10CRITTENDEN	351034N 902628W
5CROSS	351652N 9056 2W
6DALLAS	335660N 924955W
5DESHA	334850N 912718W
4DREW	333549N 915331W
8FAULKNER	35 719N 9227 7W
8FRANKLIN	3533 7N 935859W
6FULTON	362234N 92 1 1W
7GARLAND	343235N 931836W
5GRANT	341634N 9235 6W
6GREENE	36 632N 904214W
9HEMPSTEAD	334149N 935429W
10HOT SPRING	3417 6N 931522W
6HOWARD	34 419N 94 159W
12INDEPENDENCE	3543 8N 9145 0W
5IZARD	36 7 5N 92 1 1W
7JACKSON	353444N 911840W
9JEFFERSON	341724N 92 831W
7JOHNSON	353340N 9336 4W
9LAFAYETTE	331616N 933958W
8LAWRENCE	36 123N 911525W
3LEE	344659N 905346W
7LINCOLN	335837N 915429W
12LITTLE RIVER	334131N 942638W
5LOGAN	351424N 934923W
6LONOKE	344522N 915942W
7MADISON	36 0 0N 935118W
6MARION	361616N 925114W
6MILLER	3320 6N 9359 2W
11MISSISSIPPI	354554N 901341W
6MONROE	343726N 912114W
10MONTGOMERY	343025N 935028W
6NEVADA	333849N 932440W
6NEWTON	355360N 932335W
8OUACHITA	3336 7N 93 235W
5PERRY	345628N 93 256W
8PHILLIPS	342635N 905642W
4PIKE	34 752N 934420W
8POINSETT	353412N 905559W
	342935N 942020W

```
4POLK                                   352425N 931117W
4POPF                                   344746N 913954W
7PRAIRIE
```

Format 11.–*Input card to* **camfmt**

Field No.	Field name	Format	Description	Column position	Field length
1	OPT	A4	"NAME", "CODE", or " "	1–4	4
2	SKIP	I1	Sets flag to process only symbol records	5	1

Format 12.–*Geologic basin boundaries*

Field No.	Format	Description	Column position	Field length
Header card:				
1	I5	Code for geologic province on one side of line	1–5	5
2	I5	Always one (1)	6–10	5
3	I5	Code for geologic province on other side of line	11–15	5
4	I5	Number of latitude/longitude coordinates in this outline	16–20	5
5	I5	Always zero (0)	21–25	5
6	I5	State numeric FIPS[1] code	26–30	5
7	I5	992–identifies outlines as county/geologic outlines	31–35	5
8	I5	Unused (blank)	36–40	5
9	32X	Blank	41–72	32
10	8A1	Card sequence number	73–80	8
Coordinate data cards (As many of these as necessary follow header card to complete the outline.):				
1	1X	Blank	1	1
2	I11	Longitude in radians (negative for West longitude). Decimal point assumed between cc3 and cc4.	2–12	11
3	1X	Blank	13	1
4	I11	Latitude (radians). Decimal point assumed between cc15 and cc16.	14–24	11
5	1X	Blank	25	1
6	I11	Longitude (radians). Decimal point assumed between cc27 and cc28.	26–36	11
7	1X	Blank	37	1
8	I11	Latitude (radians). Decimal point assumed between cc39 and cc40.	38–48	11
9	1X	Blank	49	1
10	I11	Longitude (radians). Decimal point assumed between cc51 and cc52.	50–60	11
11	1X	Blank	61	1
12	I11	Latitude (radians). Decimal point assumed between cc63 and cc64.	62–72	11
13	8A1	Card sequence number	73–80	8

[1] Federal Information Processing Standards.

APPENDIX B. OPERATIONAL INSTRUCTIONS

LIST OF GIPSY PROCEDURES FOR PDS

ogcreate	Preformats disk space for GIPSY data file.
ogrestor	Restores PDS GIPSY file to disk from tape.
ogcreakp	Preformats and saves disk space for TSO retrievals of PDS GIPSY data.
ogclist	Executes PDS GIPSY interactively under TSO.
ogdic	Lists dictionary contents of PDS GIPSY files.
ogfilook	Computes the disk space.
ogquestr	Executes a batch mode retrieval on a PDS GIPSY file.
ogqueskp	Executes a batch mode retreival on a PDS GIPSY file and saves a formatted output file for graphical or statistical applications.
ogquesk2	Executes a batch mode retreival on a PDS GIPSY file and saves a GIPSY-formatted subset for later more economical processing.
ogquesre	Executes a batch mode retreival on a PDS GIPSY file that was created at some previous time by **ogquesk2**.
ogquesku	Executes a batch mode retreival on a PDS GIPSY file that was created by **ogquesk2**. It creates a formatted output file that can be used for graphical or statistical applications.

HOW TO RUN OGCREATE ON IBM

Purpose of the program: **ogcreate** preformats disk space for a PDS GIPSY file.

To run the program:

A. Prepare an EXEC card with the following input parameters:
   - *NAME* – Name of the file for which space is to be allocated
   - *VOL* – The volume serial number of the disk that is to be used
   - *SPACE* – Number of tracks to be allocated to the file

B. Place the card in a deck with the necessary Job Control Language (JCL) to run on IBM.

Example:

```
//S1 EXEC OGCREATE, NAME='TEXDIC',VOL=
//       'CCDnnn','SPACE=6'
```

HOW TO RUN OGRESTOR ON IBM

Purpose of the program: **ogrestor** restores a PDS GIPSY file from tape and places it on disk. It uses the preformatted space allocated by **ogcreate**.

To run the program:

A. Prepare a SETUP card for the tape. Prepare an EXEC card with the following input parameters:
   - *NAME* – Last part of a PDS file name, either DIC for a dictionary file or REC for a record file
   - *FILE* – First part of PDS file name

B. Prepare a SYSWRKI card with the following input parameters:
   - *UNIT* – Tape unit number
   - *DSN* – Name of the file on tape passed from EXEC card
   - *DISP* – old
   - *VOL=SER* – Volume serial number of the tape
   - *LABEL* – Sequential number of the file on tape
   - *DCB* – Description of the data set control block for the disk (record form, block size, and density)

C. Place the cards in a deck with the necessary JCL to run on IBM.

Examples:

```
/*SETUP          ISP004/H
//S2    EXEC   OGRESTOR, NAME='DIC',
//       FILE='TEXS'
//SYSWRKI     DD  UNIT=TAPE62,
//   DSN=&FILE&NAME
//   VOL=SER=ISP004,DISP=OLD,
//   LABEL=(1,SL),
//   DCB=(RECFM=VB,
//   BLKSIZE=8000, DEN=4)
```

HOW TO RUN OGCREAKP ON IBM

Purpose of the program: **ogcreakp** preformats and saves disk space for a subsequent interactive Time Sharing Option (TSO) retrieval of a PDS GIPSY file.

To run the program:

A. Prepare an EXEC card with the following input parameters:
   TRACKS – Number of tracks to allocate to the file
   NEWSRF – Name of the file for which the space is allocated
B. Place the card in a deck with the necessary JCL to run on IBM.

Example:

```
//C1  EXEC  OGCREAKP, TRACKS='10',
//    NEWSRF='TEXS.SAV'
```

HOW TO RUN OGCLIST ON IBM

Purpose of program: **ogclist** executes GIPSY using TSO. It enables one to access the PDS files interactively.

To run the program:

A. Logon to TSO and when the system is ready, type the EX command with the following input parameters:
   FILE – Name of the PDS file to be accessed
   WORK – Name of the disk space previously allocated by **ogcreakp**

General example:

```
ex 'og.proc (ogclist)' 'file  work'
```

Specific example:

```
ex 'og.proc (ogclist)' 'oily  sav'
```

HOW TO RUN OGDIC ON IBM

Purpose of the program: **ogdic** lists the contents of the dictionary of a PDS GIPSY file.

To run the program:

A. Prepare an EXEC card with the following input parameter:
   FILE – Name of the PDS file to be accessed
B. Place the card in a deck with the necessary JCL to run on IBM.

Example:

```
//S1  EXEC  OGDIC, FILE='OILY'
```

HOW TO RUN OGFILOOK ON IBM

Purpose of the program: **ogfilook** computes the disk space used by a PDS GIPSY file and reports the amount in *TRACKS*.

To run the program:

A. Prepare an EXEC card with the following input parameter:
   FILE – Name of the PDS file to be accessed
B. Place the card in a deck with the necessary JCL to run on IBM.

Example:

```
//C1  EXEC  OGFILOOK,FILE='OILY'
```

HOW TO RUN OGQUESTR ON IBM

Purpose of the program: **ogquestr** executes a PDS GIPSY search and retrieval in batch mode.

To run the program:

A. Prepare an EXEC card with the following input parameter:
   FILE – Name of the PDS file to be accessed
B. Prepare a QUESTRAN.SYSRDR card.
C. Prepare the cards to execute the GIPSY commands to search and retrieve data from the file. Then place the cards in a deck with the necessary JCL to run on IBM.

Example:

```
//S3  EXEC  OGQUESTR,FILE='OILY'
//    QUESTRAN.SYSRDR  DD  *
```

HOW TO RUN OGQUESKP ON IBM

Purpose of the program: **ogqueskp** executes a PDS GIPSY search and retrieval in batch mode. It saves a formatted output file for subsequent use for graphics applications.

To run the program:

A. Prepare an EXEC card with the following input parameters:
   FILE – Name of the PDS file to be accessed
   SAVFIL – Name of the formatted output file, which will be cataloged
B. Prepare a QUESTRAN.SYSWRKO card and a QUESTRAN.SYSRDR card.
C. Prepare cards to execute the GIPSY commands to search and retrieve data from the file. Use the COPY command to format and save the data values for later use. Then place the cards in a deck with the necessary JCL to run on IBM.

Examples:

```
//QK1   EXEC   OGQUESKP,FILE='OILY',
//      SAVFIL='OG.CHAFEE'
//QUESTRAN.SYSWRKO   DD  DCB=,
//      (BLKSIZE=80,LRECL=80)
//QUESTRAN.SYSRDR    DD  *
```

HOW TO RUN OGQUESK2 ON IBM

Purpose of the program: **ogquesk2** executes a PDS GIPSY search and retrieval. It then saves and catalogs the retrieved file in GIPSY format (now a subset of the original PDS file) for more economical processing at a later time.

To run the program:

A. Prepare an EXEC card with the following input parameters:
   FILE–Name of the PDS file to be accessed
   NEWSRF–Name of the output file, which is in GIPSY format (a subset of the PDS file) and will be cataloged
B. Prepare a QUESTRAN.SYSRDR card.
C. Prepare cards to execute the GIPSY commands to search and retrieve data from the input file. Then place the cards in a deck with the necessary JCL to run on IBM.

Examples:

```
//QK2   EXEC   OGQUESK2,FILE='OILY',
//     NEWSRF='OG.CHAFEE'
//QUESTRAN.SYSRDR   DD   *
```

HOW TO RUN OGQUESRE ON IBM

Purpose of the program: **ogquesre** executes a PDS GIPSY search and retrieval on a subset of the PDS file. This subset PDS file was created and saved by prior execution of **ogquesk2. ogquesre** operates on a PDS subset the way **ogquestr** operates on a PDS main file.

To run the program:

A. Prepare an EXEC card with the following input parameters:
   FILE–Name of the original PDS file accessed (allocates the proper dictionary)
   OLDSRF–The input file, which was created by a prior execution of **ogquesk2**
B. Prepare a QUESTRAN.SYSRDR card.
C. Prepare cards to execute commands to search the input file. Remeber to use only the ITERATE command to initiate a search. Remember to use the BACK command after each search to retain the input subset of PDS. Then place the cards in a deck with the necessary JCL to run on IBM.

Examples:

```
//QKR   EXEC   OGQUESRE,FILE='OILY',
//     OLDSRF='OG.CHAFEE'
//QUESTRAN.SYSRDR   DD   *
```

HOW TO RUN OGQUESKU ON IBM

Purpose of the program: **ogquesku** executes a GIPSY search and retrieval on a subset of a PDS file. This subset PDS file was created and saved by a prior execution of **ogquesk2. ogquesku** then saves a formatted output file for subsequent use for graphic applications. **ogquesku** operates on a PDS subset file the way **ogqueskp** operates on a PDS main file.

To run the program:

A. Prepare an EXEC card and a SAVFIL card with the following input parameters:
   FILE–Name of the original PDS file accessed (allocates the proper dictionary)
   OLDSRF–The input file, which was created by a prior execution of **ogquesk2**
   SAVFIL–The output file, which will be formatted and cataloged
B. Prepare a QUESTRAN.SYSWRKO card and a QUESTRAN.SYSRDR card.
C. Prepare cards to execute GIPSY commands to search and retrieve data from the file. Remember to use the ITERATE command to initiate a search. Remember to use the BACK after each search to retain the input subset of PDS. Use the COPY command to format and save the data values for later use. Then place the cards in a deck with the necessary JCL to run on IBM.

Examples:

```
//QU1      EXEC      OGQUESKU,FILE='OILY',
//  OLDSRF='OG.CHAFEE',
//  SAVFIL='OG.RET6'
//QUESTRAN.SYSWRKO DD DCB=(BLKSIZE=80,
//  LRECL=80)
//QUESTRAN.SYSRDR   DD   *
```

LIST OF CAM PROGRAMS

reformat——Reformats the original oil field and gas field outline data. Output file is OG.αβOG.

camfmt——Reformats oil field and gas field outlines to CAM 22 byte format and symbol format. Output files are *OG.CAM22.αβO* and *OG.CAMSYM.αβ(FC)(NM)*
OG.CAM22.αβG *OG.CAMSYM.αβG(FC)(NM)*

camcty——Reformats county file, OG.αβCR, to CAM 22 byte format file, *OG.CAM22.αβCR.*

ctynames——Computes and formats county names file, OG.αβCDNAME.

camnames——Reformats county names file to CAM symbol file *OG.CAMSYN.αβCD.*

camselog——Selects only specified oil fields and gas fields from the OG.αβOG files for later processing and subsequent plotting by CAM.

HOW TO RUN REFORMAT

Purpose of the program: **reformat** modifies the header cards of the oil field and gas field coordinate data sets.

To run the program: If you are using tape, first be sure the oil field and gas field coordinate data do not already exist in their original format on disk. Also, include a setup card as shown:

A. Prepare DD cards with the following variables for the input, which will be read from unit 10.
   UNIT–Kind of tape (tracks and density)
   VOL = SER–Name of disk unit
   DISP–Old
   LABEL–Type of tape (labeled or unlabeled) and file number
   DCB–Description of the data set control block for the tape (record form, logical record length, blocksize, and density)

B. Prepare DD cards with the following variables for the output, which will be written to unit 8.
   DSN–Name of file
   DISP–New and catalog
   UNIT–Type of disk
   VOL = SER–Name of disk
   DCB–Description of the data set control block for the disk (record form, logical record length, and blocksize)
   SPACE–Number of tracks required for data

Examples:

```
/*SETUP        tttnnn/9N
//GO.FT10F001   DD   UNIT=TAPE9,
//    VOL=SER= CCDnnn,
//  DISP=(OLD,KEEP),LABEL=(1,NL),
//  DCB=(RECFM=FB,LRECL=96,
//    BLKSIZE=9600,DEN=3)
// GO FT08F001   DD   DSN=OG.αβOG,
//    DISP=(NEW,CATLG,DELETE),
//  UNIT=3330,VOL=SER=CCDnnn,
//  DCB=(RECFM=FB,LRECL=96,
//    BLKSIZE=6432),
//  SPACE=(TRK,(45,3,),RLSE)
```

HOW TO RUN CAMFMT

Purpose of the program: **camfmt** is used to create the line plot and symbol plot CAM records from the oil field and gas field coordinate data sets. Separate line plot files (one for oil and one for gas) are produced so that plots may be made separately of one or the other, or so that different colors may be chosen (usually green for oil and red for gas). Separate symbol plot files are also created. The user can control the format of the symbol plot records with an entry on the single data card input to the program. Furthermore, the program can be executed more than once to produce other symbol plot file formats without reprocessing the *TYPE 2* coordinate records.

To run the program:

A. **reformat** must first be executed to create the data set OG.αβOG from its original format.

B. A single control card is read from the source deck (see format 11) and operates as follows:

Field No.	Column Position	Description
1	1–4	Blank–produces oil and gas symbol cards to plot both the field code and name.
		NAME–produces oil and gas symbol cards to plot only the field name
		CODE–produces oil and gas symbol cards to plot only the field code
2	5	Blank–creates the oil and gas line plot files.
		1–causes **camfmt** to skip processing the *TYPE 2* coordinate records, and no oil and gas line plot files are created.

C. If a printed listing of one or more of the files created by this program is desired, a subsequent step must be added to execute, for example, IEBPTPCH.

D. The following is a listing of an actual deck to compile and execute **camfmt**. The user must supply the output disk volume serial number, the State FIPS (Federal Information Processing Standards) code part of the output data sets, and the single control card. In this listing, the control card indicates that symbol plot cards are to be created for oil fields and gas fields with only the field code included, and with no processing of *TYPE 2* coordinate records. In an actual run, it is recommended that the user save only the DD cards which refer to data sets to be used. In this run, therefore, only the following DD cards would be included:

```
//GO.FT01F001
//GO.FT12F001
//GO.FT22F001
```

The units and their corresponding files follow:

Unit	File	Description
1	OGαβOG	Oil field and gas field coordinates (input)
8	*OG.CAM22.αβO*	Oil field coordinates (output)
9	*OG.CAM22.αβG*	Gas field coordinates (output)
10	*OG.CAMSYM.αβOFCNM*	Oil field code and name (output)
11	*OG.CAMSYM.αβONM*	Oil field name (output)
12	*OG.CAMSYM.αβOFC*	Oil field code (output)
20	*OG.CAMSYM.αβGFCNM*	Gas field code and name (output)
21	*OG.CAMSYMαβGNM*	Gas field name (output)
22	*OG.CAMSYMαβGFC*	Gas field code (output)

HOW TO RUN CAMCTY

Purpose of the program: **camcty** creates the line plot CAM file from the county outline latitude/longitude coordinate data sets.

To run the program: The following is a listing of an actual deck to compile and execute **camcty**. The user must supply the output disk volume serial number and the State FIPS code part of the input and output data set names.

A. Prepare a DD card with the following variables for the input file which is read from unit 10.
   DSN–File name
   DISP–Share

B. Prepare DD cards with the following variables for the output file which is written to unit 12.
   DSN–File name
   DISP–New and catalog
   UNIT–Type of disk
   VOL=SER–Name of disk
   DCB–Description of the data set control block for the disk (record form, logical record length, and blocksize)
   SPACE–Space to be allocated to the file

Examples:

```
//GO.FT10F001        DD
//  DSN=OGαβCDNAME, DISP=SHR
//GO.FT12F001        DD
//  DSN=OGCAMSYM.αβCD,
//  DISP=(NEW,CATLG,DELETE),
//  UNIT=3330,VOL=SER=CCDnnn,
//  DCB=(RECFM=FB,LRECL=80,
//  BLKSIZE=6400),
//  SPACE=(TRK,(10,2),RLSE)
```

HOW TO RUN CTYNAMES

Purpose of program: **ctynames** facilitates the creation of a data set containing the name and name length of each county in a State, and the location of the first letter of the county name within the county borders as measured from USGS base maps of scale 1:500,000. This information can subsequently be used by **camnames** to create a symbol plot file of county name locations and character strings. The measurements are taken in millimeters or any consistent unit; **ctynames** converts the measurements to latitude/longitude coordinates, makes sure the cards are in order by numeric FIPS code, and counts the characters in the county name. Furthermore, the convergence of the meridians–quite noticeable at this scale–is taken into account, thus preserving the accuracy of the measurements taken by hand.

To run the program:

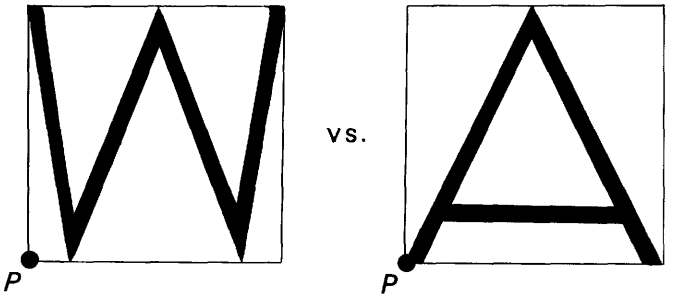
A. First, the card deck of measurements and special parameters must be made. The first card contains the following parameters:
   1. Field 1 is the State FIPS code.
   2. Field 2 is measured at the northern border of the map and is the average distance between the meridians.
   3. Field 3 is the average distance between meridians measured at the southern border of the map.
   4. Field 4 is the average distance between the parallels, measured along either the east or west border.
   5. Field 5 is the average distance between the bottom border of the map and the southernmost latitude. The easiest way to determine this distance is to take the first measurement on the eastern border of the map and then take the second measurement wherever this distance is least–usually halfway between the east and west map boundaries–and then average the two measurements.

6. Field 6 is the height of the map from southern to northern boundary.
7. Field 7 is the latitude of the parallel to which field 5 was measured.
8. Field 8, if nonblank, causes printing of the card file and the output file.

See format 8 for the format of this card.

B. All subsequent cards contain the name location measurements for each county. This information is determined by choosing a point *P* at the lower left corner of an imaginary rectangle in which the first letter of the county name is positioned. Thus, for the letters T, V, W, Y and others, *P* may not touch the character. See format 8 for the format of these cards.

For each point *P*, an arbitrary reference latitude and longitude is chosen, usually the closest one. The distance from *P* is measured in any consistent units, but preferably in millimeters. The distance is recorded as negative when *P* is south of the reference latitude for the first measurement. The distance is negative when *P* is west of the reference longitude for the second measurement.



C. For certain counties, some judgment must be exercised, keeping in mind the plotter restriction that each name must be printed on one line with consistently spaced letters. For example, *P* must be chosen arbitrarily for county names which are in two parts and printed in two lines on the maps. Some names are printed off center to avoid major cities and map detail, and *P* may have to be located elsewhere. The shapes of a few counties require the names to be printed along a curve or in some manner other than east-west. Also, where map borders cut through a county (in States which cover more than one map), the county name is usually not centered. Finally, the size of some counties requires peculiar spacing of the printed name, and some attempt should be made to compensate for this.

D. To compile and execute **ctynames**, the user must supply the output disk volume serial number, the State FIPS code part of the output data set, and the deck of measurements of county names and name locations.

1. Prepare DD cards with the following variables for the output, which is written to unit 8:
   DSN–Name of file
   DISP–New and catalog
   UNIT–Type of disk
   VOL=SER–Name of disk
   DCB–description of the data set control block for the disk (record form, logical record length, and blocksize)
   SPACE–Space to be allocated for the file.

Examples:

```
//GO.FT08F001 DD DSN=OG.ssCDNAME,
//  DISP=(NEW.CATLG),UNIT=3330,
//  VOL=SER=CCDnnn,
//  DCB=(RECFM=FB, LRECL=80,
//  BLKSIZE=1600),
//  SPACE=(1600,(5,5),RLSE)
//GO.FT05F001 DD *
56  157.  167.  222.  20.  938.  41.  X
1.   42.  -62   106.  27.    ALBANY
3.   45.  -100. 109.  107   BIG HORN
5.   44.  53.   106.  31.    CAMPBELL
7.   42.  -64.  107.  -11.  CARBON
9.   43.  -10.  106.  41.    CONVERSE
11.  45.  -97.  105.  33.    CROOK
13.  43.  -33.  109.  42.    FREMONT
15.  42.  22.   105.  78.    GOSHEN
17.  44.  -64.  109.  30.    HOT SPRINGS
19.  44.  3.    107.  29.    JOHNSON
21.  41.  67.   105.  15.    LARAMIE
23.  42.  44.   111.  35.    LINCOLN
25.  43.  -8.   107.  -15.  NATRONA
27.  43.  11.   105.  51.    NIOBRARA
29.  44.  93.   110.  66.    PARK
31.  42.  28.   105.  -21.  PLATTE
33.  45.  -51.  108.  130.   SHERIDAN
35.  43.  -47.  111.  121.   SUBLETTE
37.  42.  -86.  110.  94.    SWEETWATER
39.  44.  -15.  111.  47.    TETON
41.  41.  62.   111.  49.    UINTA
43.  44.  -13.  108.  3.     WASHAKIE
45.  44.  -37.  105.  30.    WESTON
/*
//
//
```

HOW TO RUN CAMNAMES

Purpose of the program: **camnames** creates the symbol plot CAM file from the county names and locations data set OG$\alpha\beta$CDNAME.

To run the program:

A. **ctynames** must first be executed to create the input file OG$\alpha\beta$CDNAME.

B. The following is a listing of an actual deck to compile and execute **camnames**. The user must supply the output disk volume serial number and the State FIPS code part of the input and output data set names.

1. Prepare a DD card with the following variables for the input file, which is read from unit 10.
   DSN – File name
   DISP – Share
2. Prepare DD cards with the following variables for the output file, which is written to unit 12.
   DSN – File name
   DISP – New and catalog
   UNIT – Type of disk
   VOL = SER – Name of disk
   DCB – Description of the data set control block for the disk (record form, logical record length, and blocksize)
   SPACE – Space to be allocated to the file

Examples:

```
//GO.FT10F001        DD
//  DSN=OGαβCDNAME, DISP=SHR
//GO.FT12F001        DD
//  DSN=OGCAMSYM.αβCD,
//  DISP=(NEW, CATLG,DELETE),
//  UNIT=3330,VOL=SER=CCDnnn,
//  DCB=(RECFM=FB,LRECL=80,
//  BLKSIZE=6400),
//  SPACE=(TRK,(10,2),RLSE)
```

HOW TO RUN CAMSELOG ON IBM

Purpose of the program: **camselog** selects specified petroleum fields from the oil and gas outline files so that these fields can be plotted later by CAM. It also creates a file of the remaining oil and gas outlines.

To run the program:

A. First retrieve and catalog the data from the PDS files using either **ogqueskp** or **ogquesku**. These data are created using the COPY command and the following instructions:

```
COPY
' '
FLDCODE 9
' '
FIELD 40
' '
STCODE 6
```

B. Prepare a DD card for these selected data, which will be read from unit 10. The input parameters are:
   DSN – Name of the file
   DISP – Share

C. Prepare a DD card for the oil and gas coordinate file, which will be read from unit 12. The input parameters are:
   DSN – Name of the file
   DISP – Share

D. Prepare DD cards for the selected oil and gas coordinate file (output), which will be written to unit 14. The input parameters are:
   VOL = SER – Name of disk pack
   UNIT – Type of disk
   DCB – Description of the data set control block for the disk (record form, logical record length, and blocksize)
   DSN – File name
   DISP – New
   SPACE – Number of disk tracks to be allocated to this file

E. Prepare a DD card for the oil and gas coordinates that were not selected (output), which will be written to unit 16. This DD card could be a dummy statement; otherwise, the input parameters are:
   VOL = SER – Name of disk pack
   UNIT – Type of disk
   DCB – Description of the data set control block for the disk (record form, logical record length, and blocksize)
   DSN – File name
   DISP – New
   SPACE – Number of disk tracks to be allocated to this file

Examples:

```
//GO.FT10F001        DD
//    DSN=COPENNY,DISP=SHR
//GO.FT12F001        DD
//    DSN=COOG,DISP=SHR
//GO.FT14F001        DD
//    VOL=SER=CCDnnn,UNIT=3330,
//    DCB=(RECFM=FB,LRECL=96,
//    BLKSIZE=6432),
//    DSN=SEL.COOG,DISP=(NEW,
//     CATLG,DELETE),
//    SPACE=(TRK,(42,3),RLSE)
//GO.FT16F001        DD
//    VOL=SER=CCDnnn,UNIT=3330,
//    DCB=(RECFM=FB,LRECL=96,
//    BLKSIZE=6432),
//    DSN=NOTSEL.COOG,
//    DISP=(NEW,CATLG,DELETE),
//    SPACE=(TRK,(42,3),RLSE)
```

LIST OF DISSPLA PROGRAMS

map.ec_____Plots State and county boundaries for a particular State and then plots the oil and gas fields located in that State. The maps can be generated on a Tektronix 4014 or 4016 terminal or on Calcomp 1055 plotter.

mapp.ec_____Plots State and county boundaries for a particular State. The maps can be generated on a Tektronix 4014 or 4016 terminal or on a Calcomp 1055 plotter.

mapog.ec_____Plots State and county boundaries for a particular State and then plots either all the oil or all the gas fields located in that State. The map can be generated on a Tektronix 4014 or 4016 terminal on a Calcomp 1055 plotter.

mapit2.ec_____Plots state and county boundaries for Arkansas and then plots selected oil and gas fields located in the Fort Chafee area. The map can be generated on a Tektronix 4014 or 4016 terminal or on a Calcomp 1055 plotter.

HOW TO RUN MAP.EC

Purpose of the program: **map.ec** is the EXEC_COM that executes the program **map**, which is used to plot State and county boundaries for a particular State and then the oil fields and gas fields located in that State.

To run the program:

A. Before running **map.ec** you must be in the following working directory. Type:

cwd >udd>Og>PFulton>oilgas

B. If you want a Calcomp plot, first take your tape to the computer room. Be sure the operator has the tape before running the program. Type:

sm sys opr do you have tape *nnnnnn*?

where

nnnnnn is the six-digit tape number.

C. After you have been notified that the tape has been found, type:

ec map curd*nm* *αβ*og_str

where

nm is the FIPS numeric code for the State you want to plot, and

αβ is the FIPS alpha code for the same State.

D. The program will then ask the user for the required information via a simple message such as *enter two-digit FIPS code number for the State you want to plot.*

HOW TO RUN MAPP.EC

Purpose of the program: **mapp.ec** is the EXEC_COM that executes the program **mapp**, which is used to plot State and county boundaries for a particular State.

To run the program:

A. Before running **mapp.ec** you must be in the following working directory. Type:

cwd >udd>Og>PFulton>oilgas

B. If you want a Calcomp plot, first take your tape to the computer room. Be sure the operator has the tape before running the program. Type:

sm sys opr do you have tape *nnnnnn*?

where

nnnnnn is the six-digit tape number.

C. After you have been notified that the tape has been found, type:

ec mapp curd*nm*

where

nm is the FIPS numeric code for the State you want to plot.

D. The program will then ask the user for the required information via a simple message such as *enter two-digit FIPS code number for the State you want to plot.*

HOW TO RUN MAPOG.EC

Purpose of the program: **mapog.ec** is the EXEC_COM that executes the program **mapog**, which is used to plot State and county boundaries for a particular State and then plot either the oil fields or gas fields located in that State.

To run the program:

A. Before running **mapog.ec** you must be in the following working directory. Type:

cwd >udd>Og>PFulton>oilgas

B. If you want a Calcomp plot, first take your tape to the computer room. Be sure the operator has the tape before running the program. Type:

sm sys opr do you have tape *nnnnnn*?

where

nnnnnn is the six-digit tape number.

C. After you have been notified that the tape has been found, type:

ec mapog curd*nm* *αβ*og_str

where

nm is the FIPS numeric code for the State you want to plot, and

αβ is the FIPS alpha code for the same State.

D. The program will then ask the user for the required information, via a simple message such as *enter two-digit FIPS code number for the State you want to plot.*

HOW TO RUN MAPIT2.EC

Purpose of the program: **mapit2.ec** is the EXEC_COM that executes the program **mapit2** which is used to plot State and county boundaries for Arkansas and then plot selected oil fields and gas fields located in the Fort Chaffee area.

To run the program:

A. Before running **mapit2.ec** you must be in the following working directory. Type:

cwd >udd>Og>PFulton>oilgas

B. If you want a Calcomp plot, first take your tape to the computer room. Be sure the operator has the tape before running the program. Type:

sm sys opr do you have tape *nnnnnn*?

where

nnnnnn is the six-digit tape number.

C. After you have been notified that the tape has been found, type:

ec mapit2 curd*nm* *αβ*og_str_sort

where

nm is the FIPS numeric code for the State you want to plot, and

αβ is the FIPS alpha code for the same State.

D. The program will then ask the user for the required information via a simple message such as *enter two-digit FIPS code number for the State you want to plot.*

APPENDIX C. COMPUTER PROGRAM REFERENCE

PROCEDURE NAME: OGCREATE

Author: Information Systems Programs, University of Oklahoma. **ogcreate** has been modified for use on the USGS computer system.

Purpose of the program: **ogcreate** preformats disk space for a PDS GIPSY file.

Data base: PDS

Computer: IBM 370/165

Operating system: Multiple programming variable task (MVT)

Calling sequence: EXEC OGCREATE, NAME='*file*',VOL='*CCDnnn*','SPACE='*n*'

Arguments:

NAME–Name of the file for which space is to be allocated

VOL–Volume serial number of the disk that is to be used

SPACE–Number of tracks to be allocated to the file

Procedure called: **create**

Common data referenced: None

Input files: None

Output files: None

Arrays used: None

Called by: None

Error checking and reporting: None

Constants: None

Program logic:

1. **ogcreate** passes the input parameter to **create**, a GIPSY-cataloged procedure.
2. **create** then preformats the disk space.

PDS USER PROCEDURES

```
MEMBER NAME  OGCREATE
//OGCREATE  PROC SPACE=1,NAME='',VOL=''                                  00000010
//*************************************************************************00000020
//* GIPSY PROCEDURE TO PREFORMAT DISK SPACE                              **00000030
//* TECHNICAL CONSULTING - INFORMATION SYSTEMS PROGRAMS                  **00000040
//* PARAMETERS REQUIRED:  NAME  VOL                                      **00000050
//*************************************************************************00000060
//CREATE EXEC PGM=CREATE,PARM='&SPACE'                                   00000070
//STEPLIB   DD DSN=SYS1.GIPLIB,DISP=SHR                                  00000080
//SYSPRT    DD SYSOUT=A,DCB=(BLKSIZE=120,RECFM=F)                        00000090
//GIPNEW    DD DCB=(BLKSIZE=13030,DSORG=DA),DISP=(NEW,CATLG,DELETE),     00000100
//             DSN=OG.&NAME,SPACE=(TRK,(&SPACE),,CONTIG),UNIT=3330,      00000110
//             VOL=SER=&VOL                                              00000120
```

PROCEDURE NAME: OGRESTOR

Author: Information Systems Programs, University of Oklahoma. **ogrestor** has been modified for use on the USGS computer system.

Purpose of the program: **ogrestor** restores a PDS GIPSY file from tape and places the file on disk. It uses the preformatted space allocated by **ogcreate**. Previously, the PDS file had been written to tape by the DUMP command.

Data base: PDS
Computer: IBM 370/165
Operating system: MVT

Calling sequence:
EXEC OGRESTOR, NAME = *'last part'*,
FILE = 'first part'

Arguments:
NAME–Last part of the PDS file name either **DIC** for a dictionary file or **REC** for a record file
FILE–First part of a PDS file name

Procedure called: **restore**
Common data referenced: None
Input files: PDS dictionary or record file
Output files: PDS dictionary or record file
Arrays used: None
Called by: None
Error checking and reporting: None
Constants: None
Program logic:

1. **ogrestor** passes the input parameter to **restore**, a GIPSY-cataloged procedure.
2. **restore** then reads the data from tape and writes the data to the preformatted disk space.

PDS USER PROCEDURES

```
MEMBER NAME  OGRESTOR
//OGRESTOR    PROC NAME=REC,FILE=''                                      00000010
//*************************************************************************00000020
//* GIPSY PROCEDURE TO RESTORE DISK FILE FROM TAPE                       **00000030
//* TECHNICAL CONSULTING - INFORMATION SYSTEMS PROGRAMS                  **00000040
//* PARAMETER REQUIRED:  FILE  OR  NAME                                  **00000050
//* DD CARD REQUIRED:  //SYSWRKI  DD  ...                                **00000060
//*************************************************************************00000070
//RESTORE   EXEC PGM=RESTORE,REGION=104K                                 00000080
//STEPLIB    DD DSN=SYS1.GIPLIB,DISP=SHR                                 00000090
//SYSPRINT   DD SYSOUT=A                                                 00000100
//SYSPRT     DD SYSOUT=A,DCB=(BLKSIZE=121,LRECL=121,RECFM=FA)            00000110
//SYSREC     DD DSN=OG.&FILE&NAME,DISP=SHR,UNIT=3330,                    00000120
//             VOL=SER=CCD943                                            00000130
```

PROCEDURE NAME: OGCREAKP

Author: Information Systems Programs, University of Oklahoma. **ogcreakp** has been modified for use on the USGS computer system.

Purpose of the program: **ogcreakp** preformats and saves disk space for a subsequent interactive TSO retrieval of a PDS GIPSY file.

Data base: PDS

Computer: IBM 370/165

Operating system: MVT

Calling sequence:
EXEC OGCREAKP, TRACKS='*n*',
NEWSRF="*filename*"

Arguments:
TRACKS–The number of track to allocate to the file
NEWSRF–Name of the file for which the space is allocated

Procedure called: **create**
Common data referenced: None
Input files: None
Output files: None
Arrays used: None
Called by: None
Error checking and reporting: None
Constants: None
Program logic:
1. **ogcreakp** passes the input parameters to **create**, GIPSY-cataloged procedure.
2. **create** then preformats the disk space.

PDS USER PROCEDURES

```
MEMBER NAME  OGCREAKP
//OGCREAKP    PROC TRACKS=50,TRACK=13030                                    00000010
//*****************************************************************************00000020
//* GIPSY PROCEDURE TO EXECUTE SPECIAL VERSION OF CREATE PROGRAM       **00000030
//* AND SAVE THE OUTPUT FILE                                           **00000040
//* PARAMETERS REQUIRED:            NEWSRF                             **00000050
//*****************************************************************************00000060
//CREATE   EXEC PGM=CREATE,PARM='&TRACKS'                               00000070
//*             CREATE - GIPSY PROC TO PREFORMAT DISK SPACE             00000080
//STEPLIB    DD DSN=SYS1.GIPLIB,DISP=SHR                                00000090
//SYSPRT     DD SYSOUT=A,DCB=(BLKSIZE=120,LRECL=120,RECFM=F)            00000100
//GIPNEW     DD DCB=(BLKSIZE=&TRACK,DSORG=DA),DSN=OG.&NEWSRF,           00000110
//             SPACE=(TRK,(&TRACKS)),DISP=(NEW,CATLG),UNIT=3330,        00000120
//             VOL=SER=CCD943                                           00000130
```

PROCEDURE NAME: OGCLIST

Author: Information Systems Programs, University of Oklahoma. **ogclist** has been modified for use on the USGS computer system.

Purpose of the program: **ogclist** executes GIPSY using TSO. It enables users to access the PDS files interactively.

Data base: PDS

Computer: IBM 370/165

Operating system: MVT

Calling sequence:
EX 'OG.PROC(OGCLIST)' '*file work*'

Arguments:
FILE–Name of the PDS file to be accessed
WORK–Name of the disk space previously allocated by **ogcreakp**

Procedure called: **TSO2741**
Common data referenced: None
Input files: PDS file named in the argument list
Output files: None
Arrays used: None
Called by: None
Error checking and reporting: None
Constants: None
Program logic:
1. **ogclist** passes the input parameters to **TSO2741**, the interactive version of GIPSY.
2. **TSO2741** then performs all the functions of GIPSY, but in an interactive mode.

PDS USER PROCEDURES

```
MEMBER NAME  OGCLIST
             PROC 3 POS1 POS2                                                    00000010
             ATTR LPRINT RECFM(F,A) BLKSIZE(133)                                 00000020
             ALLOC DA(*) FI(SYSWORKO) USING(LPRINT)                              00000030
             ALLOC DA(*) FI(SYSPCH)                                              00000040
             ALLOC DA('OG.&POS2.*') FI(SYSGWRK) OLD                              00000050
             ALLOC DA('OG.&POS1.REC*') FI(SYSREC) SHR                            00000060
             ALLOC DA('OG.&POS1.DIC*') FI(SYSDICT) SHR                           00000070
             ALLOC DA('OG.&POS1..INDEX1*') FI(SYSDIR) SHR                        00000080
             ALLOC DA('SYS1.SP01SORT') FI(SORTLIB) SHR                           00000090
             ALLOC   FI(SORTWK01) SPACE(700,100) BLOCK(1024) NEW                 00000100
             ALLOC   FI(SORTWK02) SPACE(700,100) BLOCK(1024) NEW                 00000110
             ALLOC   FI(SORTWK03) SPACE(700,100) BLOCK(1024) NEW                 00000120
             ALLOC DA(*) FI(SORTMSG)                                             00000130
             CALL 'SYS1.GIPLIB(TSO2741)'                                         00000140
             FREE FI(SORTWK01,SORTWK02,SORTWK03)                                 00000150
             FREE FI(SYSGWRK,SYSREC,SYSDICT)                                     00000160
             FREE FI(SYSDIR)                                                     00000170
```

PROCEDURE NAME: OGDIC

Author: Information Systems Programs, University of Oklahoma. **ogdic** has been modified for use on the USGS computer system.
Purpose of the program: **ogdic** lists the contents of the dictionary of a PDS GIPSY file.
Data base: PDS
Computer: IBM 370/165
Operating system: MVT
Calling sequence: EXEC OGDIC,FILE='name'
Arguments: *FILE*–Name of the PDS file to be accessed
Procedure called: **dictlook**
Common data referenced: None
Input files: The dictionary file of the PDS file named in the argument *FILE*.
Output files: Listing of the contents of the dictionary
Arrays used: None
Called by: None
Error checking and reporting: None
Constants: None
Program logic:

1. **ogdic** passes the input parameter to **dictlook**, a GIPSY-cataloged procedure.
2. **dictlook** then prints the contents of the dictionary.

PDS USER PROCEDURES

```
MEMBER NAME  OGDIC
             //OGDIC    PROC FILE=**                                           00000010
             //*********************************************************************00000020
             //* GIPSY PROCEDURE TO LIST DICTIONARY CONTENTS                  **00000030
             //* TECHNICAL CONSULTING - INFORMATION SYSTEMS PROGRAMS          **00000040
             //* PARAMETER REQUIRED:  FILE                                    **00000050
             //*********************************************************************00000060
             //DICTLOOK EXEC PGM=DICTLOOK                                      00000070
             //STEPLIB    DD DSN=SYS1.GIPLIB,DISP=SHR                          00000080
             //SYSPRT     DD SYSOUT=A,DCB=(LRECL=121,BLKSIZE=121,RECFM=FA)     00000090
             //SYSDICT    DD DSN=OG.&FILE.DIC,DISP=SHR,UNIT=3330,VOL=SER=CCD943 00000100
```

PROCEDURE NAME: OGFILOOK

Author: Information Systems Programs, University of Oklahoma. **ogfilook** has been modified for use on the USGS computer system.
Purpose of the program: **ogfilook** computes the disk space used by a PDS GIPSY file and reports the amount in tracks.
Data base: PDS
Computer: IBM 370/165
Operating system: MVT
Calling sequence:
EXEC OGFILOOK, FILE='*name*'
Arguments: *FILE*–Name of the PDS file to be accessed
Procedure called: **filelook**
Common data referenced: None
Input files: The PDS file named in the argument *FILE*
Output files: None
Arrays used: None
Called by: None
Error checking and reporting: None
Constants: None
Program logic:

1. **ogfilook** passes the input parameter to **filelook**, a GIPSY-cataloged procedure.
2. **filelook** then computes the disk space used by the input file and reports the amount in tracks.

PDS USER PROCEDURES

```
MEMBER NAME  OGFILOOK
//OGFILOOK PROC NAME=REC,FILE=''                                          00000010
//*************************************************************************00000020
//* GIPSY PROCEDURE TO GIVE SPACE USED IN DISK FILE                      **00000030
//* TECHNICAL CONSULTING - INFORMATION SYSTEMS PROGRAMS                  **00000040
//* PARAMETER REQUIRED:  FILE  OR  NAME                                  **00000050
//*************************************************************************00000060
//FILELOOK EXEC PGM=FILELOOK                                              00000070
//STEPLIB  DD DSN=SYS1.GIPLIB,DISP=SHR                                    00000080
//SYSPRT     DD SYSOUT=A,DCB=(LRECL=121,BLKSIZE=121,RECFM=FA)             00000090
//SYSGWRK    DD DSN=OG.&FILE&NAME,DISP=SHR,UNIT=3330,                     00000100
//             VOL=SER=CCD943                                             00000110
```

PROCEDURE NAME: OGQUESTR

Author: Information Systems Programs, University of Oklahoma. **ogquestr** has been modified for use on the USGS computer system.

Purpose of the program: **ogquestr** executes a PDS GIPSY search and retrieval in batch mode.

Data base: PDS

Computer: IBM 370/165

Operating system: MVT

Calling sequence:

EXEC OGQUESTR, FILE='*name*'

Arguments: ***FILE***–Name of the PDS file to be accessed

Procedures called: **create, gipexec**

Common data referenced: None

Input files: The PDS file named in the argument ***FILE***

Output files: None

Arrays used: None

Called by: None

Error checking and reporting: None

Constants: None

Program logic:

1. **ogquestr** calls **create**, a GIPSY-cataloged procedure. **create** then formats the disk space that will be used during the run.
2. **ogquestr** then calls **gipexec**, the batch version of Gipsy. **gipexec** performs all the functions of GIPSY in batch mode.

PDS USER PROCEDURES

```
MEMBER NAME  OGQUESTR
//OGQUESTR    PROC SPACE=20,TRACKS=50,TRACK=13030                         00000010
//*************************************************************************00000020
//* GIPSY PROCEDURE TO EXECUTE QUESTRAN BATCH PROGRAM                    **00000030
//* TECHNICAL CONSULTING - INFORMATION SYSTEMS PROGRAMS                  **00000040
//* PARAMETER REQUIRED:  FILE                                           **00000050
//* DD CARD REQUIRED:  //QUESTRAN.SYSRDR  DD  ...                       **00000060
//*              OR    //QUESTRAN.SYSIN   DD  ...                       **00000070
//*************************************************************************00000080
//CREATE   EXEC PGM=CREATE,PARM='&TRACKS'                                 00000090
//*            CREATE - GIPSY PROC TO PREFORMAT DISK SPACE                00000100
//STEPLIB    DD DSN=SYS1.GIPLIB,DISP=SHR                                  00000110
//           DD DSN=SYS1.PPLNKSRT,DISP=SHR                                00000120
//SYSPRT     DD SYSOUT=A,DCB=(BLKSIZE=120,LRECL=120,RECFM=F)              00000130
//GIPNEW     DD DSN=&&SRF,DCB=(BLKSIZE=&TRACK,DSORG=DA),                  00000140
//             UNIT=SYSDK,DISP=(NEW,PASS),SPACE=(TRK,(&TRACKS))           00000150
//QUESTRAN   EXEC PGM=GIPEXEC,REGION=120K                                 00000160
//STEPLIB    DD DSN=SYS1.GIPLIB,DISP=SHR                                  00000170
//           DD DSN=SYS1.PPLNKSRT,DISP=SHR                                00000180
//SYSPRT     DD SYSOUT=A,DCB=(BLKSIZE=121,LRECL=121,RECFM=FA)             00000190
//SYSDICT    DD DSN=OG.&FILE.DIC,DISP=SHR,UNIT=3330,VOL=SER=CCD943        00000200
//SYSREC     DD DSN=OG.&FILE.REC,DISP=SHR,UNIT=3330,VOL=SER=CCD943        00000210
//SYSGWRK    DD DSN=&&SRF,DISP=(OLD,DELETE)                               00000220
//SORTLIB    DD DSN=SYS1.SM01SORT,DISP=SHR                                00000230
//SORTWK01   DD UNIT=SYSDK,SPACE=(TRK,(&SPACE),,CONTIG)                   00000240
//SORTWK02   DD UNIT=SYSDK,SPACE=(TRK,(&SPACE),,CONTIG)                   00000250
//SORTWK03   DD UNIT=SYSDK,SPACE=(TRK,(&SPACE),,CONTIG)                   00000260
//SORTWK04   DD UNIT=SYSDK,SPACE=(TRK,(&SPACE),,CONTIG)                   00000270
//SORTWK05   DD UNIT=SYSDK,SPACE=(TRK,(&SPACE),,CONTIG)                   00000280
//SORTWK06   DD UNIT=SYSDK,SPACE=(TRK,(&SPACE),,CONTIG)                   00000290
//SORTWK07   DD UNIT=SYSDK,SPACE=(TRK,(&SPACE),,CONTIG)                   00000300
//SORTWK08   DD UNIT=SYSDK,SPACE=(TRK,(&SPACE),,CONTIG)                   00000310
//SORTMSG    DD SYSOUT=A                                                  00000320
//SYSOUT     DD SYSOUT=A                                                  00000330
//SYSPCH     DD DUMMY                                                     00000340
//SYSDIR     DD DUMMY                                                     00000350
//SYSWRK0    DD SYSOUT=A,DCB=(BLKSIZE=121,LRECL=121,RECFM=FA)             00000360
//SYSRDR     DD DDNAME=SYSIN,DCB=BLKSIZE=80                               00000370
```

PROCEDURE NAME: OGQUESKP

Author: Information Systems Programs, University of Oklahoma. **ogqueskp** has been modified for use on the USGS computer system.

Purpose of the program: **ogqueskp** executes a PDS GIPSY search and retrieval in batch mode. It then saves a formatted output file for subsequent use for graphic applications. The COPY command formats the data.

Data base: PDS

Computer: IBM 370/165

Operating system: MVT

Calling sequence:

EXEC OGQUESKP, FILE='*name*',SAVFIL='*data*'

Arguments:

FILE–Name of the PDS file to be accessed

SAVFIL–Name of the formatted output file, which will be cataloged

Procedures called: **create, gipexec**

Common data referenced: None

Input files: The PDS file named in the argument ***FILE***

Output files: The formatted data named in the argument ***SAVFIL***

Arrays used: None

Called by: None

Error checking and reporting: None

Constants: None

Program logic:

1. **ogqueskp** calls **create**, a GIPSY-cataloged procedure. **create** then formats the disk space that will be used during the run.
2. **oqueskp** then calls **gipexec**, the batch version of GIPSY. **gipexec** performs all the functions of GIPSY in batch mode.
3. **ogqueskp** uses ***SYSWRKO*** to store the output file, ***FILSAV***. The output file is usually a card image format, a maximum of 80 columns. Use the COPY command to format ***SAVFIL***.

PDS USER PROCEDURES

```
MEMBER NAME  OGQUESKP
//OGQUESKP    PROC SPACE=20,TRACKS=50,TRACK=13030,TRAKK=50                  00000010
//*************************************************************************00000020
//* GIPSY PROCEDURE TO EXECUTE QUESTRAN BATCH PROGRAM                     **00000030
//* AND SAVE THE FORMATTED OUTPUT FILE FOR PLOTTING                        00000040
//* PARAMETERS REQUIRED:  FILE AND  SAVFIL                                 00000050
//* DD CARD REQUIRED:  //QUESTRAN.SYSRDR  DD  ...                        **00000060
//*                OR      //QUESTRAN.SYSIN   DD  ...                    **00000070
//*************************************************************************00000080
//CREATE   EXEC PGM=CREATE,PARM='&TRACKS'                                  00000090
//*             CREATE - GIPSY PROC TO PREFORMAT DISK SPACE                00000100
//STEPLIB     DD DSN=SYS1.GIPLIB,DISP=SHR                                  00000110
//SYSPRT      DD SYSOUT=A,DCB=(BLKSIZE=120,LRECL=120,RECFM=F)              00000120
//GIPNEW   DD DCB=(BLKSIZE=&TRACK,DSORG=DA),DSN=&&SRF,                     00000130
//              UNIT=SYSDK,DISP=(NEW,PASS),SPACE=(TRK,(&TRACKS))           00000140
//QUESTRAN EXEC PGM=GIPEXEC,REGION=250K                                    00000150
//STEPLIB     DD DSN=SYS1.GIPLIB,DISP=SHR                                  00000160
//             DD DSN=SYS1.PPLNKSRT,DISP=SHR                               00000170
//SYSPRT      DD SYSOUT=A,DCB=(BLKSIZE=121,LRECL=121,RECFM=FA)             00000180
//SYSDICT     DD DSN=OG.&FILE.DIC,DISP=SHR,UNIT=3330,VOL=SER=CCD943        00000190
//SYSREC      DD DSN=OG.&FILE.REC,DISP=SHR,UNIT=3330,VOL=SER=CCD943        00000200
//SYSGWRK     DD DSN=&&SRF,DISP=(OLD,DELETE)                               00000210
//SORTLIB     DD DSN=SYS1.SM01SORT,DISP=SHR                                00000220
//SORTWK01    DD UNIT=SYSDK,SPACE=(TRK,(&SPACE),,CONTIG)                   00000230
//SORTWK02    DD UNIT=SYSDK,SPACE=(TRK,(&SPACE),,CONTIG)                   00000240
//SORTWK03    DD UNIT=SYSDK,SPACE=(TRK,(&SPACE),,CONTIG)                   00000250
//SORTWK04    DD UNIT=SYSDK,SPACE=(TRK,(&SPACE),,CONTIG)                   00000260
//SORTWK05    DD UNIT=SYSDK,SPACE=(TRK,(&SPACE),,CONTIG)                   00000270
//SORTWK06    DD UNIT=SYSDK,SPACE=(TRK,(&SPACE),,CONTIG)                   00000280
//SORTWK07    DD UNIT=SYSDK,SPACE=(TRK,(&SPACE),,CONTIG)                   00000290
//SORTWK08    DD UNIT=SYSDK,SPACE=(TRK,(&SPACE),,CONTIG)                   00000300
//SORTMSG     DD SYSOUT=A                                                  00000310
//SYSOUT      DD SYSOUT=A                                                  00000320
//SYSPCH      DD DUMMY                                                     00000330
//SYSDIR      DD DUMMY                                                     00000340
//SYSWRKO     DD DSN=&SAVFIL,                                              00000350
//          SPACE=(TRK,(&TRAKK,5),RLSE),DISP=(NEW,CATLG,DELETE),          00000360
//             DCB=(BLKSIZE=6435,LRECL=117,RECFM=FB),                     00000370
//              UNIT=3330,VOL=SER=CCD943                                  00000380
//SYSRDR      DD DDNAME=SYSIN,DCB=BLKSIZE=80                               00000390
```

PROCEDURE NAME: OGQUESK2

Author: Information Systems Programs, University of Oklahoma. **ogquesk2** has been modified for use on the USGS computer system.

Purpose of the program: **ogquesk2** executes a PDS GIPSY search and retrieval. It then saves and catalogs the retrieved file in GIPSY format (now a subset of the original PDS file) for more economical processing at a later time.

Data base: PDS

Computer: IBM 370/165

Operating System: MVT

Calling sequence:

EXEC OGQUESK2, FILE='*name*', NEWSRF='*save*'

Arguments:

FILE–Name of the PDS file to be accessed

NEWSRF–Name of the output file, which is in GIPSY format (a subset of the PDS file) and will be cataloged

Procedures called: **create, gipexec**

Common data referenced: None

Input files: The PDS file named in the argument *FILE*

Output files: The GIPSY file named in the argument *NEWSRF*

Arrays used: None

Called by: None

Error checking and reporting: None

Contents: None

Program logic:

1. **ogquesk2** calls **create,** a GIPSY-cataloged procedure. **create** then formats the permanent disk space that will be used during the run.
2. **ogquesk2** uses *GIPNEW* to assign the name given by *NEWSRF* to the output file and to catalog the output file.
3. **ogquesk2** then calls **gipexec,** the batch version of GIPSY. **gipexec** performs all the function of GIPSY in batch mode.

PDS USER PROCEDURES

```
MEMBER NAME  OGQUESK2
//OGQUESK2    PROC SPACE=20,TRACKS=50,TRACK=13030                          00000010
//*************************************************************************00000020
//* GIPSY PROCEDURE TO EXECUTE QUESTRAN BATCH PROGRAM                     **00000030
//* AND SAVE THE GYPSY SUBSET FOR FURTHER RETREIVALS                      **00000040
//* PARAMETERS REQUIRED:  FILE  NEWSRF                                    **00000050
//* DD CARD REQUIRED:  //QUESTRAN.SYSRDR  DD  ...                         **00000060
//*              OR      //QUESTRAN.SYSIN   DD  ...  OR *                 **00000070
//*************************************************************************00000080
//CREATE   EXEC PGM=CREATE,PARM='&TRACKS'                                  00000090
//*             CREATE - GIPSY PROC TO PREFORMAT DISK SPACE                00000100
//*                      AND SAVE AND CATALOG THE NEWSRF                   00000110
//STEPLIB    DD DSN=SYS1.GIPLIB,DISP=SHR                                   00000120
//SYSPRT     DD SYSOUT=A,DCB=(BLKSIZE=120,LRECL=120,RECFM=F)               00000130
//GIPNEW     DD DSN=&NEWSRF,DCB=(BLKSIZE=&TRACK,DSORG=DA),                 00000140
//             UNIT=3330,VOL=SER=CCD943,SPACE=(TRK,(&TRACKS)),             00000150
//             DISP=(NEW,CATLG,DELETE)                                     00000160
//QUESTRAN EXEC PGM=GIPEXEC,REGION=120K                                    00000170
//STEPLIB    DD DSN=SYS1.GIPLIB,DISP=SHR                                   00000180
//             DD DSN=SYS1.PPLNKSRT,DISP=SHR                               00000190
//SYSPRT     DD SYSOUT=A,DCB=(BLKSIZE=121,LRECL=121,RECFM=FA)              00000200
//SYSDICT    DD DSN=OG.&FILE.DIC,DISP=SHR,UNIT=3330,VOL=SER=CCD943         00000210
//SYSREC     DD DSN=OG.&FILE.REC,DISP=SHR,UNIT=3330,VOL=SER=CCD943         00000220
//SYSCWRK    DD DSN=&NEWSRF,DISP=(OLD,KEEP,DELETE)                         00000230
//SORTLIB    DD DSN=SYS1.SM01SORT,DISP=SHR                                 00000240
//SORTWK01   DD UNIT=SYSDK,SPACE=(TRK,(&SPACE),,CONTIG)                    00000250
//SORTWK02   DD UNIT=SYSDK,SPACE=(TRK,(&SPACE),,CONTIG)                    00000260
//SORTWK03   DD UNIT=SYSDK,SPACE=(TRK,(&SPACE),,CONTIG)                    00000270
//SORTMSG    DD SYSOUT=A                                                   00000280
//SYSOUT     DD SYSOUT=A                                                   00000290
//SYSPCH     DD DUMMY                                                      00000300
//SYSDIR     DD DUMMY                                                      00000310
//SYSWRK0    DD SYSOUT=A,DCB=(BLKSIZE=121,LRECL=121,RECFM=FA)              00000320
//SYSRDR     DD DDNAME=SYSIN,DCB=BLKSIZE=80                                00000330
```

PROCEDURE NAME: OGQUESRE

Author: Information Systems Programs, University of Oklahoma. **ogquesre** has been modified for use on the USGS computer system.

Purpose of the program: **ogquesre** executes a PDS GIPSY search and retrieval on a subset of a PDS file. A prior execution of **ogquesk2** created and saved this subset PDS file. **ogquesre** operates on a PDS subset the way **ogquestr** operates on a PDS main file.

Data base: PDS

Computer: IBM 370/165

Operating systems: MVT

Calling sequence:

EXEC OGQUESRE,FILE = *'name'*, OLDSRF = *'save'*

Arguments:

FILE – Name of the original PDS file accessed (allocates the proper dictionary)

OLDSRF – The input file, which was created by a prior execution of **ogquesk2.**

Procedures called: **gipexec**

Common data referenced: None

Input files: The PDS dictionary named in the argument *FILE*. The PDS subset named in the argument *OLDSRF*.

Output files: None

Arrays used: None

Called by: None

Error checking and reporting: None

Constants: None

Program logic:

1. **ogquesre** calls **gipexec**, the batch version of GIPSY. **gipexec** performs all the functions of GIPSY in batch mode.
2. **ogquesre** uses *SYSGWRK* to access *OLDSRF*.
3. To initiate a search and retrieval, use only the ITERATE command. After each search, use the BACK command to retain the entire input subset of PDS. Failure to use these two commands properly will cause the entire file to be lost.

PDS USER PROCEDURES

```
MEMBER NAME  OGQUESRE
//OGQUESRE    PROC SPACE=20                                                  00000010
//*************************************************************************00000020
//* GIPSY PROCEDURE TO EXECUTE QUESTRAN BATCH PROGRAM                     **00000030
//* ON A FILE SAVED FROM A PREVIOUS GIPSY RETREIVAL                       **00000040
//* PARAMETERS REQUIRED:  FILE AND  OLDSRF                                **00000050
//* DD CARD REQUIRED:  //QUESTRAN.SYSRDR  DD  ...                         **00000060
//*              OR      //QUESTRAN.SYSIN   DD  ...                      **00000070
//*************************************************************************00000080
//QUESTRAN EXEC PGM=GIPEXEC,REGION=120K                                     00000090
//STEPLIB     DD DSN=SYS1.GIPLIB,DISP=SHR                                   00000100
//            DD DSN=SYS1.PPLNKSRT,DISP=SHR                                 00000110
//SYSPRT      DD SYSOUT=A,DCB=(BLKSIZE=121,LRECL=121,RECFM=FA)              00000120
//SYSDICT     DD DSN=OG.&FILE.DIC,DISP=SHR,UNIT=3330,VOL=SER=CCD943         00000130
//SYSREC      DD DSN=OG.&FILE.REC,DISP=SHR,UNIT=3330,VOL=SER=CCD943         00000140
//SYSGWRK     DD DSN=&OLDSRF,DISP=SHR                                       00000150
//SORTLIB     DD DSN=SYS1.SM01SORT,DISP=SHR                                 00000160
//SORTWK01    DD UNIT=SYSDK,SPACE=(TRK,(&SPACE),,CONTIG)                    00000170
//SORTWK02    DD UNIT=SYSDK,SPACE=(TRK,(&SPACE),,CONTIG)                    00000180
//SORTWK03    DD UNIT=SYSDK,SPACE=(TRK,(&SPACE),,CONTIG)                    00000190
//SORTMSG     DD SYSOUT=A                                                   00000200
//SYSOUT      DD SYSOUT=A                                                   00000210
//SYSPCH      DD DUMMY                                                      00000220
//SYSDIR      DD DUMMY                                                      00000230
//SYSWRKO     DD SYSOUT=A,DCB=(BLKSIZE=121,LRECL=121,RECFM=FA)              00000240
//SYSRDP      DD DDNAME=SYSIN,DCB=BLKSIZE=80                                00000250
```

PROCEDURE NAME: OGQUESKU

Author: Information Systems Programs, University of Oklahoma. **ogquesku** has been modified for use on the USGS computer system.

Purpose of the program: **ogquesku** executes a PDS GIPSY search and retrieval on a subset of a PDS file. A prior execution of **ogquesk2** created and saved this subset PDS file. **ogquesku** then saves a formatted output file for subsequent use for graphic applications. **ogquesku** operates on a PDS subset file the way **ogqueskp** operates on the PDS main file.

Data base: PDS

Computer: IBM 370/165

Operating System: MVT

Calling sequence: EXEC OGQUESKU,FILE='*name*', OLDSRF='*save*',SAVFIL='*newsave*'

Arguments:

FILE–Name of the original PDS file accessed (allocates the proper dictionary)

OLDSRF–The input file, which was created by a prior execution of **ogquesk2**

SAVFIL–The output file, which will be formatted and cataloged

Procedure called: **gipexec**

Common data referenced: None

Input files:

The PDS dictionary named in the argument *FILE*

The PDS subset named in the argument *OLDSRF*

Output files: The formatted data named in the argument *SAVFIL*

Arrays used: None

Called by: None

Error checking and reporting: None

Constants: None

Program logic:

1. **ogquesku** calls **gipexec,** the batch version of GIPSY. **gipexec** performs all the functions of GIPSY in batch mode.
2. **ogquesku** uses *SYSGWRK* to access *OLDSRF*. It uses *SYSWRKO* to store the output file, *FILSAV*. The output file is usually in card image format, a maximum of 80 columns.
3. To initiate a search and retrieval, use only the ITERATE command. After each search, use the BACK command to retain the entire input subset of PDS. Failure to use these two commands properly will cause the entire file to be lost. Use the COPY command to format *SAVFIL*.

PDS USER PROCEDURES

```
MEMBER NAME  OGQUESKU
//OGQUESKU    PROC SPACE=20,TRACKS=50                                          00000010
//*****************************************************************************00000020
//* GIPSY PROCEDURE TO EXECUTE QUESTRAN BATCH PROGRAM                         **00000030
//* ON A FILE SAVED FROM A PREVIOUS GIPSY RETREIVAL                           **00000040
//* AND SAVE THE OUTPUT FOR LATER PLOTTING                                    **00000050
//* PARAMETERS REQUIRED:  FILE, OLDSRF, AND SAVFIL                            **00000060
//* DD CARD REQUIRED:  //QUESTRAN.SYSRDR  DD  ...                             **00000070
//*              OR     //QUESTRAN.SYSIN   DD  ...                           **00000080
//*****************************************************************************00000090
//QUESTRAN EXEC PGM=GIPEXEC,REGION=120K                                        00000100
//STEPLIB    DD DSN=SYS1.GIPLIB,DISP=SHR                                       00000110
//           DD DSN=SYS1.PPLNKSRT,DISP=SHR                                     00000120
//SYSPRT     DD SYSOUT=A,DCB=(BLKSIZE=121,LRECL=121,RECFM=FA)                  00000130
//SYSDICT    DD DSN=OG.&FILE.DIC,DISP=SHR,UNIT=3330,VOL=SER=CCD943             00000140
//SYSREC     DD DSN=OG.&FILE.REC,DISP=SHR,UNIT=3330,VOL=SER=CCD943             00000150
//SYSGWRK    DD DSN=&OLDSRF,DISP=SHR                                           00000160
//SORTLIB    DD DSN=SYS1.SM01SORT,DISP=SHR                                     00000170
//SORTWK01   DD UNIT=SYSDK,SPACE=(TRK,(&SPACE),,CONTIG)                        00000180
//SORTWK02   DD UNIT=SYSDK,SPACE=(TRK,(&SPACE),,CONTIG)                        00000190
//SORTWK03   DD UNIT=SYSDK,SPACE=(TRK,(&SPACE),,CONTIG)                        00000200
//SORTMSG    DD SYSOUT=A                                                       00000210
//SYSOUT     DD SYSOUT=A                                                       00000220
//SYSPCH     DD DUMMY                                                          00000230
//SYSDIR     DD DUMMY                                                          00000240
//SYSWRKO    DD DSN=&SAVFIL,                                                   00000250
//           SPACE=(TRK,(&TRACKS),RLSE),DISP=(NEW,CATLG,DELETE),               00000260
//           DCB=(BLKSIZE=121,LRECL=121,RECFM=F),                              00000270
//           UNIT=3330,VOL=SER=CCD943                                          00000280
//SYSRDR     DD DDNAME=SYSIN,DCB=BLKSIZE=80                                    00000290
```

PROGRAM NAME: REFORMAT

Author: R. W. Coffin

Purpose of the program: **reformat** is used to modify the header, *TYPE 1*, records of the oil field and gas field coordinate data sets to a more convenient form. The coordinate records, *TYPE 2*, are left unaltered. The major change makes the oil field and gas field name the last field, along with a count of its characters.

Data base: PDS

Computer: IBM 370/165

Operating system: MVT

Calling sequence: None

Arguments: None

Subroutines called: None

Common data referenced: None

Input file: Tape file, oil field and (or) gas field coordinate data

Output file: OG.αβOG
Printer file

Arrays used:

A (96) INTEGER*2 The alpha array which contains the old *TYPE 1* card

B (96) INTEGER*2 The alpha array used to construct the new *TYPE 1* card

Called by: None

Error checking and reporting: If the name field exceeds 45 characters in length, a message is sent to the printer specifying the *TYPE 1* card number and the length of the alpha field. The program is aborted. Summary statistics are printed giving the number of *TYPE 1* cards processed and the number of *TYPE 2* cards (left unchanged).

Constants: None

Program logic:

1. A record (96 bytes) is read into the alpha array *A*. The absence of a decimal point in the 27th position (which all *TYPE 2* cards have) identifies a *TYPE 1* card.
2. The first 43 columns are unaffected. Columns 44 through 96 are blanked in array *B*.
3. The column numbers of the commas after the field name and after the last numeric field are determined. From this, the lengths of the name and numeric fields are calculated.
4. If the alpha field is greater than 45 characters, an error message is printed and the program stops.
5. The first 43 characters of array *A* are moved to array *B* unaltered.
6. The next five columns are reserved for the variable length numeric field which used to be the last item. It is right justified into this field and so becomes of fixed length.
7. The alpha field is moved to array *B* nine columns to the right, leaving room to the left for the five-column numeric field (step 6), a comma, a two-digit numeric field (the length of the alpha field), and another comma.
8. Finally, the two-digit length of the alpha field (computed in step 3) is placed to the left of the alpha field. Commas are left between all fields and are moved with the data during the above steps.
9. After the last *TYPE 1* card is processed, statistics are printed.

```
//  - JOB -
/*SETUP     ttt/9N
//*  REFORMAT
//STEP  EXEC FORTXCG
//FORT.SYSIN  DD *
C  THIS PROGRAM REFORMATS THE OIL AND GAS FIELD COORDINATE DATA
C  TYPE ONE CARDS.  THE TYPE TWO CARDS ARE UNCHANGED.
C     NEW FORMAT TYPE ONE CARDS:
C  CC 1 TO 43  SAME AS OLD TYPE 1 CARDS
C  CC 44 TO 48  NUMBER OF COORDINATES TO PLOT,
C     RIGHT JUSTIFIED, AND ZERO FILLED
C  CC 49  COMMA
C  CC 50 TO 51  LENGTH OF ALPHA FIELD (= N)
C  CC 52  COMMA
C  53 TO 52 + N  ALPHA FIELD (NAME OF OIL OR GAS FIELD)
C  THE LAST COMMA OF THE OLD TYPE 1 CARD HAS BEEN DELETED
C
```

```
      IMPLICIT INTEGER*2(A-Z)
      DIMENSION A(96), B(96)
      DATA DEC /1H./, COMMA /1H,/
      DATA BLANK /1H /
C
      KOUNT1 = 0
      KOUNT2 = 0
1     READ(10,10,END=100) A
10    FORMAT(96A1)
C
C  TYPE 1 OR TYPE 2 CARD?
      IF(A(27) .NE. DEC) GO TO 15
C
C  TYPE 2 CARDS UNCHANGED
      WRITE(8,10) A
      KOUNT2 = KOUNT2 + 1
      GO TO 1
C
C  BLANK OUT THE PART OF ARRAY TO BE MODIFIED
15    DO 20 I = 44, 96
20       B(I) = BLANK
C
C  COUNT TYPE 1 CARDS
      KOUNT1 = KOUNT1 + 1
C
C  FIND FIRST AND SECOND COMMA PAST CC 43
      K1 = 0
      DO 25 I = 44, 96
         IF (A(I) .NE. COMMA) GO TO 25
         IF (K1 .NE. 0) GO TO 30
C
C  K1 IS POSITION OF FIRST COMMA
         K1 = I
25    CONTINUE
      I = 96
C
C  K2 IS POSITION OF SECOND COMMA
30    K2 = I
C
C   N IS LENGTH OF NUMERIC FIELD
      N = K2 - K1 - 1
C
C  M IS LENGTH OF ALPHA FIELD
      M = K1 - 44


      IF (M .GE. 45) GO TO 2000
C
C  FIRST 43 CHARACTERS UNCHANGED
      DO 40 I = 1, 43
40       B(I) = A(I)
C
```

```
C  BLANK OUT NEW NUMERIC FIELD
      DO 50 I = 44,48
50       B(I) = BLANK
      K2 = K2 + 1
      NP1 = N + 1
C
C  REPOSITION NUMERIC FIELD AND INCLUDE COMMA AT 49
      DO 60 I = 1, NP1
         J1 = 50 - I
         J2 = K2 - I
60       B(J1) = A(J2)
      M1 = M + 52
C
C  REPOSITION ALPHA FIELD AND INCLUDE COMMA AT 52
      DO 70 I = 52, M1
70       B(I) = A(I - 9)
C
C  PLACE IN POSITION 50 AND 51 THE LENGTH OF ALPHA FIELD
      WRITE(8,11) (B(I), I=1,49), M, (B(I), I=52,96)
11    FORMAT(49A1, I2, 45A1)
      GO TO 1
C
100   WRITE (6,12) KOUNT1, KOUNT2
12    FORMAT(//27H NUMBER OF TYPE ONE CARDS =,I6,/
     X27H NUMBER OF TYPE TWO CARDS =,I6)
      STOP
2000  WRITE(6,14) M, KOUNT1
14    FORMAT(//22H ALPHA FIELD TOO BIG =,I6,/
     X22H ERROR IN TYPE 1 CARD , I6)
      STOP
      END
/*
//GO.FT10F001  DD UNIT=TAPE9,VOL=SER=ttt,
//    DISP=(OLD,KEEP),LABEL=(1,NL),DCB=(RECFM=FB,
//    LRECL=96,BLKSIZE=9600,DEN=3)
//GO.FT08F001  DD DSN=OG.ssOG,DISP=(NEW,CATLG),
//    UNIT=3330,VOL=SER=CCDnnn,DCB=(RECFM=FB,
//    LRECL=96,BLKSIZE=3072),SPACE=(3072,(200,50),RLSE)
//
//
```

PROGRAM NAME: CAMFMT

Author: R. W. Coffin

Purpose of the program: **camfmt** creates the 22-byte line plot and 80-byte symbol plot CAM records from the oil field and gas field coordinate data set. Separate files are created for oil and gas. The user can control the format of the symbol plot file. The program can be invoked to produce other symbol plot file formats without reprocessing *TYPE 2* coordinate records.

Data base: PDS
Computer: IBM 370/165
Operating system: MVT
Calling sequence: None
Arguments called: None
Subroutine called: **dms**
Common data referenced: None
Input files:
OG.$\alpha\beta$OG, where $\alpha\beta$ is the State FIPS code on unit 1
Card file–a single control card

Output files: Printer file
Optional:
OG.CAM22αβO – Oil line plot file on unit 8
OG.CAM22αβG – Gas line plot file on unit 9
One of:
OG.CAMSYM.αβOFCNM – Oil symbol plot file with field code and name on unit 10
OG.CAMSYM.αβOFC – Oil symbol plot file with field code only on unit 12
OG.CAMSYM.αβONM – Oil symbol plot file with field name only on unit 11
Corresponding to one of:
OG.CAMSYM.αβGFCNM – Gas symbol plot file with field code and name on unit 20
OG.CAMSYM.αβGFC – Gas symbol plot file with field code only on unit 22
OG.CAMSYM.αβGNM – Gas symbol plot file with field name only on unit 21

Arrays used: None

Called by: None

Error checking and reporting: If there are more than 40 characters to plot, a warning message is written to the printer specifying which *TYPE 1* card is being processed and the number of characters. The program is not aborted.

Summary statistics are printed giving the number of input records, the total number of output binary records for oil and gas, and the total number of symbol plot records.

Constants: *CONVT* Real 1.7453293E-2 = π/180
Used to convert degrees to radians.

Program logic:

1. The control card information is read into *OPT* and *SKIP*, which is used to determine the fields to be included on the symbol plot records and whether to process *TYPE 2* cards.
2. A *TYPE 1* card is read which gives the contents of of the symbol plot records, whether the contents are an oil field or a gas field, and other information.
3. The number of records read and the number of outlines processed are updated.
4. **dms** is called to convert the location of the field center in decimal degrees to degrees, minutes, and seconds – the format required on the symbol plot records.
5. The appropriate line identifier is incremented and file number is set (*FILE* = 10 for oil, and *FILE* = 20 for gas).
6. A symbol plot record is written consisting of a count of the symbols, the characters "+ " (to place a cross on the field center), the field code and (or) field name as specified by *OPT* (see below), and the latitude/longitude coordinates of the characters to be plotted.
   OPT = " " (blank) – Write both field code and name
   OPT = "*NAME*" – Write field name only
   OPT = "*CODE*" – Write field code only
7. If there are more than 40 characters to plot, a warning is sent to printer.
8. The number of lines (input records) for this outline is calculated from the number of coordinate pairs in this outline (found in step 2); there are up to five pairs of latitude/longitude coordinates per line. The number of records read is also incremented by this amount.
9. If *SKIP* is set to 1 (in step 1), then no *TYPE 2* records are processed. (This step is useful if **camfmt** is executed a second time to create different symbol plot record formats.) The next *TYPE 1* card is read.
10. The number of coordinate pairs on the last *TYPE 2* card of this outline is calculated.
11. Within the processing loop, the five pairs of latitude/longitude coordinates are read, the degrees are converted to radians, and the west longitude is made negative.
12. For each pair of coordinates, the appropriate (oil or gas) sequence count is incremented, and an unformatted binary write is performed to the appropriate file.
13. After the last outline is processed, statistics are printed.

```
//  - JOB -
//*  CAMFMT
//STEP  EXEC FORTXCB
//FORT.SYSIN  DD *
C    THIS PROGRAM CONVERTS THE O & G FIELD COORDINATE DATA TO
C    THE 22 BYTE BINARY FORMAT REQUIRED BY CAM.  ONE FILE IS
C    CREATED TO PLOT OIL FIELDS, ANOTHER FOR GAS FIELDS.  ALSO
C    THE 80 BYTE CAM SYMBOL PLOT FILE IS PRODUCED FROM THE
```

```
C    TYPE 1 CARD INFORMATION AND FORMATTED ACCORDING TO
C    THE VALUE ON THE INPUT CARD AS FOLLOWS:
C
C      OPT = CODE          O & G FIELD CODE ONLY IS PLOTTED
C      OPT = NAME          O & G FIELD NAME ONLY IS PLOTTED
C      OPT =    (BLANK)    O & G FIELD CODE AND NAME ARE PLOTTED
C      SKIP  = 1           DO NOT PROCESS TYPE 2 CARDS
C
C    THE VALUE OF OPT IS READ FROM CC 1 - 4 FROM THE SINGLE
C    INPUT CARD TO THE PROGRAM.  SKIP IS READ FROM CC 5.
C
      INTEGER LINIDO/0/, LINIDG/0/, RECD/0/, OUTLN/0/, SKIP,
     X SEQCNT, SEQO/0/, SEQG/0/, FILE, OPT, NAME/4HNAME/, CODE/4HCODE/
      INTEGER*2 FC(6), FNAME(44), RANK/0/, GAS/1HG/, OG
      DATA NAME /4HNAME/, CODE /4HCODE/
      REAL LAT, LONG, LTLG(5,2), LTLGR(5,2)
C
C    CONVERSION FACTOR = PI/180
      CONVT = 1.7453293E-2
10    READ(5,5) OPT, SKIP
5     FORMAT(A4,I1)
1     READ(01,20,END=170) FC, OG, LAT, LONG, N, M, FNAME
20    FORMAT(19X,6A1,1X,1A1,1X,F6.3,1X,F7.3,1X,I5,1X,I2,1X,44A1)
C
C    INCREMENT INPUT AND OUTPUT RECORD COUNTS
      RECD = RECD + 1
      OUTLN = OUTLN + 1
C
C    CONVERT DECIMAL DEGREES TO DEG, MIN, AND SEC
      CALL DMS(LAT, LATD, LATM, LATS)
      CALL DMS(LONG, LONGD, LONGM, LONGS)
C
C
      IF(OG .EQ. GAS) GO TO 30
         LINIDO = LINIDO + 1
         FILE = 10
         GO TO 40
30    LINIDG = LINIDG + 1
         FILE = 20
C
C    WRITE SYMBOL PLOT FILE DEPENDING ON OPTION
40    IF(OPT .EQ. CODE) GO TO 80
      IF(OPT .EQ. NAME) GO TO 60
C
C    WRITE FIELD CODE AND NAME
      NCH = M + 9
      WRITE(FILE,50) NCH, FC, FNAME,
     XLATD, LATM, LATS, LONGD, LONGM, LONGS
50    FORMAT(I2,2H+ ,6A1,1H ,44A1,4X,3I2,1HN,I3,2I2,1HW)
      GO TO 100
C
C    WRITE NAME ONLY
60    NCH = M + 2
```

```
      FILE = FILE + 1
      WRITE(FILE,70) NCH, FNAME,
     XLATD, LATM, LATS, LONGD, LONGM, LONGS
70    FORMAT(I2,2H+ ,44A1,11X,3I2,1HN,I3,2I2,1HW)
      GO TO 100
C
C   WRITE FIELD CODE ONLY
80    NCH = 8
      FILE = FILE + 2
      WRITE(FILE,90) NCH, FC,
     XLATD, LATM, LATS, LONGD, LONGM, LONGS
90    FORMAT(I2,2H+ ,6A1,49X,3I2,1HN,I3,2I2,1HW)
C
C   WRITE WARNING IF THERE ARE MORE THAN 40 CHARACTERS TO PLOT
100   IF(NCH .GT. 40) WRITE(6,110) OUTLN, NCH
110   FORMAT(1H ,'TYPE 1 CARD NO.',I5,' PLOTS',I5,' CHARACTERS')
C
C   CALCULATE # OF TYPE 2 CARDS FOLLOWING THIS TYPE 1 CARD
      LINES = (N+4)/5
      RECD = RECD + LINES
C
C   SHOULD TYPE 2 CARDS BE PROCESSED?
      IF(SKIP .NE. 1) GO TO 120
         DO  115  L = 1, LINES
115         READ(01,5)
            GO TO 1
C
C   CALCULATE # OF COORDINATE PAIRS IN LAST CARD
120   K1 = N-(N/5)*5
      IF(K1 .EQ. 0) K1 = 5
C
C   PROCESS THIS OUTLINE
      DO  160  I = 1, LINES
C
C   READ IN ONE LINE OF COORDINATE PAIRS
         READ(01,130)  ((LTLG(J1,J2), J2 = 1, 2), J1 = 1, 5)
130      FORMAT(16X,5(F6.3,1X,F7.3,1X))
C
C   IF LAST LINE, MAY HAVE FEWER THAN 5 COORDINATE PAIRS
         K = 5
         IF(I .EQ. LINES) K = K1
         DO 150  J = 1, K
C
C           CONVERT TO RADIANS
            LTLGR(J,1) = LTLG(J,1)*CONVT
            LTLGR(J,2) = -LTLG(J,2)*CONVT
C
C           UNFORMATTED WRITE -- 22 BYTE BINARY FORMAT CARDS
            IF(OG .EQ. GAS) GO TO 140
            SEQO = SEQO + 1
            WRITE(8) LINIDO, RANK, LTLGR(J,1), LTLGR(J,2), SEQO
            GO TO 150
140         SEQG = SEQG + 1
            WRITE(9) LINIDG, RANK, LTLGR(J,1), LTLGR(J,2), SEQG
```

```
150    CONTINUE
160    CONTINUE
C
C    DONE WITH THIS OUTLINE -- GET NEXT TYPE 1 CARD
       GO TO 1
C
C    FINISHED PROCESSING -- PRINT STATISTICS
170    SEQCNT = SEQO + SEQG
       WRITE(6,200) RECD, SEQCNT, OUTLN
200    FORMAT(//' NUMBER OF INPUT RECORDS =',I7,//,
      X          ' NUMBER OF OUTPUT BINARY RECORDS =',I7,//,
      X          ' NUMBER OF SYMBOL PLOT RECORDS =',I7)
       STOP
       END
C    THE FOLLOWING SUBROUTINE CONVERTS DECIMAL DEGREES TO DEGREES
C     MINUTES, AND SECONDS
       SUBROUTINE DMS(DEG, D, M, S)
       INTEGER D, M, S
C    WHOLE DEGREES -- FRACTION PART TRUNCATED
       D = DEG
C    GET FRACTION PART
       F1 = DEG-D
C    CALCULATE # OF MINUTES
       TEMP = F1*60.0
C    WHOLE MINUTES -- FRACTION PART TRUNCATED
       M = TEMP
C    CALCULATE # OF SECONDS
       F2 = (TEMP-M)*60.0
C    WHOLE SECONDS -- ROUNDED
       S = INT(F2*2.0)-INT(F2)
       RETURN
       END
/*
//GO.FT01F001  DD DSN=OG.ssOG,DISP=SHR
//GO.FT08F001  DD DSN=OG.CAM22.ssO,DISP=(NEW,CATLG),
//    UNIT=3330,VLL=SER=CCDnnn,DCB=(RECFM=VBS,
//    LRECL=22,BLKSIZE=1650),SPACE=(1650,(200,100),RLSE)
//GO.FT09F001  DD DSN=OG.CAM22.ssG,DISP=(NEW,CATLG),
//    UNIT=3330,VLL=SER=CCDnnn,DCB=(RECFM=VBS,
//    LRECL=22,BLKSIZE=1650),SPACE=(1650,(200,100),RLSE)
//GO.FT10F001  DD DSN=OG.CAMSYM.ssOFCNM,DISP=(NEW,CATLG),
//    UNIT=3330,VOL=SER=CCDnnn,DCB=(RECFM=FB,LRECL=80,
//    BLKSIZE=1280),SPACE=(1280,(20,10),RLSE)
//GO.FT11F001  DD DSN=OG.CAMSYM.ssONM,DISP=(NEW,CATLG),
//    UNIT=3330,VOL=SER=CCDnnn,DCB=(RECFM=FB,LRECL=80,
//    BLKSIZE=1280),SPACE=(1280,(20,10),RLSE)
//GO.FT12F001  DD DSN=OG.CAMSYM.ssOFC,DISP=(NEW,CATLG),
//    UNIT=3330,VOL=SER=CCDnnn,DCB=(RECFM=FB,LRECL=80,
//    BLKSIZE=1280),SPACE=(1280,(20,10),RLSE)
//GO.FT20F001  DD DSN=OG.CAMSYM.ssGFCNM,DISP (NEW,CATLG),
//    UNIT=3330,VOL=SER=CCDnnn,DCB=(RECFM=FB,LRECL=80,
//    BLKSIZE=1280),SPACE=(1280,(20,10),RLSE)
//GO.FT21F001  DD DSN=OG.CAMSYM.ssGNM,DISP=(NEW,CATLG),
//    UNIT=3330,VOL=SER=CCDnnn,DCB=(RECFM=FB,LRECL=80,
```

```
//    BLKSIZE=1280),SPACE=(1280,(20,10),RLSE)
//GO.FT22F001  DD DSN=OG.CAMSYM.ssGFC,DISP=(NEW,CATLG),
//    UNIT=3330,VOL=SER=CCDnnn,DCB=(RECFM=FB,LRECL=80,
//    BLKSIZE=1280),SPACE=(1280,(20,10),RLSE)
//GO.FT05F001  DD *
CODE1
/*
//
//
```

SUBROUTINE NAME: DMS

Purpose of program: **dms** converts decimal degrees to degrees, minutes and seconds.
Data base: None
Computer: IBM 370/165
Operating system: MVT
Calling sequence: CALL DMS(DEG, D, M, S)
Arguments:
DEG – decimal degrees – REAL
D – whole degrees – INTEGER
M – whole minutes – INTEGER
S – whole seconds – INTEGER
Subroutines called: None
Common data referenced: None

PROGRAM NAME: CAMCTY

Author: R. W. Coffin
Purpose of the Program: **camcty** creates the 22-byte line plot CAM records from the county outline coordinate data sets. No processing of the header cards takes place.
Data base: PDS
Computer: IBM 370/165
Operating system: MVT
Calling sequence: None
Subroutines called: None
Common data referenced: None
Input file:
OG.$\alpha\beta$CR, where $\alpha\beta$ is the State FIPS code
Output files: *OG.CAM22.$\alpha\beta$CR*
Printer file
Arrays used: None
Called by: None
Error checking and reporting: Summary statistics are printed, giving the total number of input records, the number of output binary records, and the number of header cards.

Input files: None
Output files: None
Arrays used: None
Called by: **camfmt, camnames**
Error checking and reporting: None
Constants: None
Program logic:
1. Integer truncation is used to determine the number of degrees. Minutes and seconds are calculated from the fraction part of *DEG*, with seconds rounded to the nearest unit.

Note that this subroutine is called by two programs. Since it is short, a copy of **dms** exists in the source code of each of these programs to avoid the use of linkage editor.

Constants: None
Program logic:
1. The fourth integer field of a header card is read; it gives the number of coordinate pairs in this outline.
2. This number is used to calculate the number of lines (input records) for this outline and the number of coordinate pairs in the last line. (There are up to three pairs of longitude/latitude coordinates per line.)
3. The line identifier is incremented for this outline, and the number of input records processed is updated.
4. Within the processing loop, the three pairs of latitude/longitude coordinates are read, the numbers are converted to real numbers (floating point), and the order is reversed – CAM expects latitude first. The first pair of coordinates in each outline is skipped; this pair is the location of a point within the county borders for plotting the FIPS code.
5. For each pair of coordinates, the sequence count is incremented and an unformatted binary write is performed.
6. After the last outline is processed, statistics are printed.

```
//  - JOB -
//*  CAMCTY
//STEP  EXEC FORTXCG
//FORT.SYSIN  DD *
C    THIS PROGRAM CONVERTS COUNTY OUTLINE COORDINATE DATA TO THE
C    22 BYTE BINARY FORMAT REQUIRED BY CAM.  THE HEADER CARDS
C    WHICH IDENTIFY THE COUNTY, ETC., ARE IGNORED.
C
      INTEGER SEQCNT/0/, LINID/0/, RANK*2/0/, RECD/0/, LGLT(3,2)
      REAL XLTLG(3,2)
1     READ(10,10,END=100) N
10    FORMAT(15X,I5)
C
C    INCREMENT INPUT RECORD COUNT
      RECD = RECD+1
C
C    SET NEW LINE IDENTIFIER FOR THIS OUTLINE
      LINID = LINID+1
C
C    CALCULATE NUMBER OF DATA CARDS FOLLOWING THIS HEADER CARD
      LINES = (N+2)/3
C
C    ...AND ADD THIS # OF CARDS TO INPUT RECORD COUNT
      RECD = RECD+LINES
C
C    CALCULATE NUMBER OF COORDINATE PAIRS ON LAST CARD
      K1 = N-(N/3)*3
      IF(K1 .EQ. 0) K1 = 3
C
C    PROCESS THIS OUTLINE
      DO  20  I = 1, LINES
C
C    READ IN ONE LINE OF LONG/LAT COORDINATE PAIRS
        READ(10,30) ((LGLT(M1,M2), M2 = 1, 2), M1 = 1, 3)
30      FORMAT(6(1X,I11))
C
C    IF LAST LINE, MAY HAVE FEWER THAN 3 COORDINATE PAIRS
        K = 3
        IF(I .EQ. LINES) K = K1
C
C    IF FIRST LINE, SKIP FIRST PAIR OF COORDINATES
        L = 1
        IF(I .EQ. 1) L = 2
        DO  40  J = L, K
          SEQCNT = SEQCNT+1
C
C    CONVERT TO FLOATING POINT, POSITION DECIMAL POINT,
C    AND  REVERSE ORDER OF COORDINATE PAIRS
          XLTLG(J,2) = LGLT(J,1)
          XLTLG(J,1) = LGLT(J,2)
          XLTLG(J,2) = XLTLG(J,2)/1.E9
          XLTLG(J,1) = XLTLG(J,1)/1.E9
C
C    UNFORMATTED WRITE -- 22 BYTE BINARY-FORMAT CARDS
```

```
40          WRITE(8) LINID, RANK, XLTLG(J,1), XLTLG(J,2), SEQCNT
20     CONTINUE
       GO TO 1
C
C    FINISHED PROCESSING--PRINT STATISTICS
100    WRITE(6,200) RECD, SEQCNT, LINID
200    FORMAT(//' NUMBER OF INPUT RECORDS =',I7,//,
      X          ' NUMBER OF OUTPUT BINARY RECORDS =',I7,//,
      X          ' NUMBER OF HEADER CARDS =',I7)
       STOP
       END
/*
//GO.FT10F001  DD DSN=OG.ssCR,DISP=SHR
//GO.FT08F001  DD DSN=OG.CAM22.ssCR,DISP=(NEW,CATLG),
//    UNIT=3330,VOL=SER=CCDnnn,DCB=(RECFM=VBS,
//    LRECL=22,BLKSIZE=1650),SPACE=(1650,(200,100),RLSE)
//
//
```

PROGRAM NAME: CTYNAMES

Author: R. W. Coffin

Purpose of the program: **ctynames** creates a data set of 80-byte records containing the name and length of the name of each county in a State, the latitude/longitude coordinates of the first letter of the county name, and the numeric FIPS code for the county and State. This information can subsequently be used by **camnames** to create a symbol plot file of county name locations and character strings. The raw data is input to **ctynames** in a card deck created from measurements taken in millimeters from USGS 1:500,000-scale base maps. **ctynames** thus reduces the possibility of error in estimating latitude/longitude coordinates directly from the maps. In the conversion routine, the program takes into account the fact that the meridians are noticeably nonparallel at this scale, and thus **ctynames** helps preserve the accuracy of the measurements taken by hand.

Data base: None

Computer: IBM 370/165

Operating system: MVT

Calling sequence: None

Subroutines called: None

Common data referenced: None

Input files: Card deck of measurements taken from USGS base maps of scale 1:500,000

Output files:
OG.$\alpha\beta$CDNAME, where $\alpha\beta$ is the State FIPS code
Printer file

Arrays used: None

Called by: None

Error checking and reporting: If there are more than 40 characters in the county name field, a warning message is sent to the printer specifying the county numeric FIPS code and the number of characters in the name. If there are more than 45 characters, the program is also aborted. If a card is out of order by increasing numeric FIPS code, a message is printed identifying by numeric FIPS code the last good card, and the program is aborted.

Constants: None

Program logic:

1. The first card contains special measurements and other parameters taken from the map (see format 8, appendix A). This card must be present. It includes a field that, if not left blank, will cause the input cards and output records to be sent also to the printer.
2. In the processing loop, a data card is read. This card contains the county name, the county numeric FIPS code, and measurements of the lower left corner of the first letter of the county name (point *P*).

Measurements are taken between this point and an arbitrary reference latitude (longitude). All numeric values are real to facilitate key punching. Note that the formulas can handle positive and negative measurements.

3. A check is made that this FIPS code is greater than the previous one read. If not, a message is written and the program is aborted.
4. Latitude is computed by taking a ratio of the distance (usually in millimeters) between P and the reference latitude to the distance between the latitudes. This ratio is added to the reference latitude giving an answer in degrees north latitude.
5. Longitude is computed in a similar way, giving the result in degrees west longitude. In addition, however, an approximation is made to compensate for the fact that the distance between the meridians is substantially less at the top of the map than at the bottom. Thus, the distance from P to the southern border of the map is also computed and used in a linear approximation of the distance between the meridians at the latitude of P.
6. The length of the county name is determined by looking for the first two adjacent blanks after the name. Note that columns 79 and 80 must, therefore, be left blank on the input data cards.
7. If the county name length is greater than 40 characters, a warning mesage is printed; if it is greater than 45, a message is printed and the program stops.

```
//  - JOB -
//*  CTYNAMES
//STEP  EXEC FORTXCG
//FORT.SYSIN  DD *
C    NOTE THAT ALL MEASUREMENTS ARE IN MILLIMETERS
C    INPUT VARIABLES:
C    STATE       FIPS CODE FOR THIS STATE
C    A           AVG DISTANCE (MM) BETWEEN MERIDIANS MEASURED AT TOP OF MAP
C    B           AVG DISTANCE BETWEEN MERIDIANS MEASURED AT BOTTOM OF MAP
C    C           AVG DISTANCE BETWEEN PARALLELS
C    D           AVG DISTANCE BETWEEN BOTTOM BORDER AND SOUTHERNMOST LATITUDE
C    H           VERTICAL HEIGHT OF MAP FROM BORDER TO BORDER
C    L           SOUTHERNMOST LATITUDE (TO WHICH D IS MEASURED)
C    PRNT        IF NON-BLANK, OUTPUT IS ALSO PRINTED TO PRINTER
C
C    FIPS        COUNTY FIPS CODE
C    LAT         REFERENCE LATITUDE
C    MLAT        MM FROM P TO REFERENCE LATITUDE (POSITIVE IF P IS NORTH)
C    LONG        REFERENCE LONGITUDE
C    MLONG       MM FROM P TO REFERENCE LONGITUDE (POSITIVE IF P IS EAST)
C    CTY         ALPHA ARRAY CONTAINING COUNTY NAME (CC 79 AND 80 MUST BE BLANK)
C
      REAL LAT, LONG, MLAT, MLONG, LT, LG, L
      INTEGER CTY*2(54), CHS, STATE, FIPSC/0/, BLANK*2/1H /, PRNT*2
C
C    READ PGM PARAMETER CARD
      READ(5,10) STATE, A, B, C, D, H, L, PRNT
10    FORMAT(I3,1X,6F5.0,1A1)
      IF(PRNT .NE. BLANK) WRITE(6,5) A, B, C, D, H, L
5     FORMAT(1H ,6F9.1)
C
C    COMPUTE CONSTANT OUTSIDE PROCESSING LOOP
      T1 = (A-B)/H
C
C    READ A DATA CARD
15    READ(5,20,END=300) FIPS, LAT, MLAT, LONG, MLONG, CTY
20    FORMAT(F5.0,F4.0,F6.0,F5.0,F6.0,54A1)
      IFIPS = FIPS
C
C    CHECK ORDER OF CARDS
```

```
      IF(FIPSC .GE. IFIPS) GO TO 100
      FIPSC = IFIPS
C
C   COMPUTE DECIMAL LONGITUDE AND LATITUDE
      T = B + T1*((LAT-L)*C + D + MLAT)
      LG = LONG - MLONG/T
      LT = LAT + MLAT/C
C
C   DETERMINE LENGTH OF COUNTY NAME
      DO  30  I = 1,53
         IF(CTY(I) .NE. BLANK) GO TO 30
         IF(CTY(I+1) .EQ. BLANK) GO TO 40
30    CONTINUE
40    CHS = I-1
      IF(CHS .GT. 40) WRITE(6,50) IFIPS, I
50    FORMAT(1H ,'COUNTY NUMBER',I5,'  HAS',I3,'  CHARACTERS')
      IF(CHS .GT. 45) GO TO 200
      WRITE(8,60) STATE, IFIPS, LT, LG, CHS, (CTY(I), I = 1, 45)
60    FORMAT(2(I4,1H,),2(F9.3,1H,),I4,1H,,45A1)
      IF(PRNT .EQ. BLANK) GO TO 15
C
      WRITE(6,65) FIPS, LAT, MLAT, LONG, MLONG
65    FORMAT(1H0,F5.0,F4.0,F6.0,F5.0,F6.0)
      WRITE(6,70) STATE, IFIPS, LT, LG, CHS, (CTY(I), I = 1, 45)
70    FORMAT(1H ,2(I4,1H,),2(F9.3,1H,),I4,1H,,45A1)
      GO TO 15
C
100   WRITE(6,110) FIPSC
110   FORMAT(1H ,'CARD AFTER COUNTY NUMBER',I4,'  OUT OF ORDER')
200   WRITE(6,210)
210   FORMAT(1H ,'...AND PGM ABORTED')
300   STOP
      END
/*
//GO.FT08F001  DD DSN=OG.ssCDNAME,DISP=(NEW,CATLG),
//   UNIT=3330,VOL=SER=CCDnnn,DCB=(RECFM=FB,
//   LRECL=80,BLKSIZE=1600),SPACE=(1600,(5,5),RLSE)
//GO.FT05F001  DD *
 56 157. 167. 222.  20. 938.  41. X
1.   42. -62.  106. 27.   ALBANY
3.   45. -100. 109. 107.  BIG HORN
5.   44. 53.   106. 31.   CAMPBELL
7.   42. -64.  107. -11.  CARBON
9.   43. -10.  106. 41.   CONVERSE
11.  45. -97.  105. 33.   CROOK
13.  43. -33.  109. 42.   FREMONT
15.  42. 22.   105. 78.   GOSHEN
17.  44. -64.  109. 30.   HOT SPRINGS
19.  44. 3.    107. 29.   JOHNSON
21.  41. 67.   105. 15.   LARAMIE
23.  42. 44.   111. 35.   LINCOLN
25.  43. -8.   107. -15.  NATRONA
27.  43. 11.   105. 51.   NIOBRARA
29.  44. 93.   110. 66.   PARK
31.  42. 28.   105. -21.  PLATTE
33.  45. -51.  108. 130.  SHERIDAN
35.  43. -47.  111. 121.  SUBLETTE
37.  42. -86.  110. 94.   SWEETWATER
39.  44. -15.  111. 47.   TETON
```

```
41.   41. 62.    111. 49.    UINTA
43.   44. -13.   108. 3.     WASHAKIE
45.   44. -37.   105. 30.    WESTON
/*
//
//
```

PROGRAM NAME: CAMNAMES

Author: R. W. Coffin

Purpose of the program: **camnames** creates the 80-byte symbol plot CAM file from the county names and locations data set created by **ctynames**.

Data base: None

Computer: IBM 370/165

Operating system: MVT

Calling sequence: None

Arguments: None

Subroutine called: **dms**

Common data referenced: None

Input files: OG.$\alpha\beta$CDNAME–where $\alpha\beta$ is the State FIPS code

Output files: *OG.CAMSYM.$\alpha\beta$CD*
Printer file

Array used: None

Called by: None

Error checking and reporting: If there are more than 40 characters to plot, a warning message is sent to the printer specifying the county numeric FIPS code and the number of characters.

At the conclusion of the program, the number of records processed is sent to the printer

Constants: None

Program logic:

1. For each card read, **dms** is called to convert decimal degrees to degrees, minutes, and seconds–the format that CAM requires.
2. The symbol plot card is written, with a warning sent to the printer if the county name field contains more than 40 characters.
3. The number of records processed is printed.

```
//  - JOB -
//*  CAMNAMES
//STEP  EXEC FORTXCG
//FORT.SYSIN  DD *
C    THIS PROGRAM CREATES AN 80 BYTE CAM SYMBOL PLOT FILE
C    FROM THE COUNTY NAMES AND LOCATIONS FILE CREATED
C    BY CTYNAMES.
C
      REAL LT, LG
      INTEGER STATE, CHS, CTY*2 (45), RECD /0/
1     READ(10,10,END=100) STATE, IFIPS, LT, LG, CHS, CTY
10    FORMAT(I4,1X,I4,1X,F9.3,1X,F9.3,1X,I4,1X,45A1)
C
C    INCREMENT INPUT  RECORD COUNT
      RECD = RECD + 1
C
C    CONVERT DECIMAL DEGREES TO DEG, MIN, AND SEC
      CALL DMS(LT, LTD, LTM, LTS)
      CALL DMS(LG, LGD, LGM, LGS)
      WRITE(12,20) CHS, CTY, LTD, LTM, LTS, LGD, LGM, LGS
20    FORMAT(I2,45A1,12X,3I2,1HN,I3,2I2,1HW)
C
C    WRITE WARNING IF THERE ARE MORE THAN 40 CHARACTERS
      IF(CHS .GT. 40) WRITE (6, 30) IFIPS, CHS
30    FORMAT(1H ,'COUNTY CARD',I5,' PLOTS',I5,'  CHARACTERS')
      GO TO 1
100   WRITE(6,110) RECD
```

```
110    FORMAT(' NUMBER OF RECORDS PROCESSED = ',I5)
       STOP
       END
C
C    THE FOLLOWING SUBROUTINE CONVERTS DECIMAL DEGREES TO DEGREES,
C     MINUTES, AND SECONDS
       SUBROUTINE DMS(DEG, D, M, S)
       INTEGER D, M, S
C
C    WHOLE DEGREES -- FRACTION PART TRUNCATED
       D = DEG
C
C    GET FRACTION PART
       F1 = DEG - D
C
C    CALCULATE # OF MINUTES
       TEMP = F1*60.0
C
C    WHOLE MINUTES -- FRACTION PART TRUNCATED
       M = TEMP
C
C    CALCULATE # OF SECONDS
       F2 = (TEMP - M) * 60.0
C
C    WHOLE SECONDS -- ROUNDED
       S = INT(F2*2.0) - INT(F2)
       RETURN
       END
/*
//GO.FT10F001  DD DSN=OG.ssCDNAME,DISP=SHR
//GO.FT12F001  DD DSN=OG.CAMSYM.ssCD,DISP=(NEW,CATLG),
//    UNIT=3330,VOL=SER=CCDnnn,DCB=(RECFM=FB,LRECL=80,
//    BLKSIZE=1600),SPACE=(1600,(5,5),RLSE)
//
```

PROGRAM NAME: CAMSELOG

Author: P. A. Fulton

Purpose of the program: **camselog** selects specified petroleum oil fields from the oil and gas outline files so that these fields can be plotted later by CAM. It also creates a file of the remaining oil and gas outlines.

Data base: PDS

Computer: IBM 370/165

Operating system: MVT

Calling sequence: **camselog**

Arguments: None

Subroutines called: None

Common data referenced: None

Input files:

Selected data file on unit 10

OG.$\alpha\beta$OG – coordinate file of oil fields and gas fields on unit 12

Output files:

Coordinate file of selected oil fields and gas fields on unit 14

Coordinate file of remaining oil fields and gas fields on unit 16

Arrays used:

IFIELD(40),*ILFLD*(2),*I1*(3),*I2*(34),*I3*(48),*IA*(96)

Called by: None

Error checking and reporting: The selected data file should be in ascending order by field number. If the numbers are out of order, a message is printed on the output listing and the program continues. If the field number is missing, a message is printed on the output listing and the program continues. If the number of coordinates is incorrect, a message is printed on the output listing and the program halts. If the end of the coordinate file is reached before the selected file is ex-

hausted, a message is written to the output listing and the program halts.

Constants: None

Program logic:

1. The program reads a field number from the selected file. The file of field numbers was retrieved and cataloged from PDS by **ogqueskp** or **ogquesku**. The file was formatted by the COPY command and the following instructions:

```
COPY
' '
FLDCODE 9
' '
FIELD 40
' '
STCODE 6
```

2. The program reads a field number from the coordinate file.
3. When the field numbers from the two files match, the the coordinates for that field are written to unit 14 where they become the selected coordinate file.
4. When the field numbers from the two files do not match, the coordinates for that field are written to unit 16.

```
//  - JOB -
//* CAMSELOG
//STEP  EXEC FORTYCC
//FORT.SYSIN  DD *
C
C     THIS PROGRAM SELECTS CERTAIN OIL AND GAS FIELDS FOR LATER PLOTTING
C
      DIMENSION IFIELD(40), ILFLD(2),I1(3),I2(34),I3(48),IA(96)
C
C    CHECK INPUT FILE OF SELECTED FIELD CODES AND NAMES FOR BLANKS AND DUPLICATES
C
      CALL SELFIL
C
      IC=0
C
C             READ FIELD NUMBER - IDFLD - FROM SELECTED FILE
C
  200 CONTINUE
C
      READ  (10,110,END=235) IDSTAT,IDFLD,IFIELD,ILFLD
  110 FORMAT (I3,1X,I6,1X,40A1,1X,2A1)
C
      IC=IC+1
C
      IF (IDFLD) 201,201,210
C
  201 WRITE (6,111)
  111 FORMAT (' NO FIELD NUMBER')
      WRITE ( 6,110)          IDSTAT,IDFLD,IFIELD,ILFLD
C
      GO TO 200
C
C
C     READ HEADER CARD OF COORDINATE FILE AND GET FIELD NUMBER - IOGFLD
C
  210 CONTINUE
      READ (12,113,END=230) I1,IOGFLD,I2,COUNT,I3
  113 FORMAT (3A1,I6,34A1,F5.0,48A1)
C
C     COMPUTE NUMBER OF CARD IMAGES TO READ FROM COORDINATE FILE - IDO
C          ROUND UP IF NECESSARY
```

```
C
      KOUNT=COUNT
      ADD =COUNT/5.0
      IDD=ADD
      BDD=IDD
      IF (ADD-BDD) 250,212,211
  250 WRITE (6,130) ADD,IDD,BDD
  130 FORMAT (F10.2,I8,F10.2)
      STOP 2
  211 IDD=IDD+1
C
C   DETERMINE IF THIS IS A SELECTED OR NONSELECTED FIELD
C
  212 CONTINUE
      IF (IDFLD-IOGFLD) 214,215,218
C
  214 WRITE (6,114)
  114 FORMAT (*  NOT FOUND*)
      WRITE ( 6,110)          IDSTAT,IDFLD,IFIELD,ILFLD
      BACKSPACE 12
      GO TO 200
C
C       WRITE DATA TO SELECTED COORDINATE FILE
C
  215 CONTINUE
      WRITE(14,115)          I1,IOGFLD,I2,KOUNT,I3
  115 FORMAT (3A1,I6,34A1,I5,48A1)
C
      DO 216 I=1,IDD
      READ  (12,116,END=240) IA
  116 FORMAT (96A1)
      WRITE (14,116) IA
  216 CONTINUE
C
C     CHECK FOR MULTIFLE FIELDS I.E. OIL AND GAS BOTH
C
      READ (12,113,END=230) I1,IOGFLD,I2,COUNT,I3
       IF (IDFLD-IOGFLD) 380,381,203
  380 BACKSPACE 12
      GO TO 200
  381 BACKSPACE 12
       GO TO 210
C
  203 WRITE (6,112)
  112 FORMAT (*  O - G FILE OUT OF ORDER*)
      WRITE ( 6,110)          IDSTAT,IDFLD,IFIELD,ILFLD
      STOP 3
C
C
  240 WRITE (6,119)
  119 FORMAT (*  EOF ON COOR DATA - UNIT 12*)
      GO TO 236
C
C        WRITE DATA TO NONSELECTED COORDINATE FILE
C
  218 CONTINUE
      WRITE(16,115)          I1,IOGFLD,I2,KOUNT,I3
      DO 219 I=1,IDD
      READ  (12,116,END=240) IA
      WRITE (16,116) IA
```

```
  219 CONTINUE
C
      GO TO 210
C
  230 CONTINUE
      WRITE (6,117)
  117 FORMAT (' LAST CARD')
      WRITE ( 6,110)          IDSTAT,IDFLD,IFIELD,ILFLD
      GO TO 236
C
  235 WRITE (6,118) IC
  118 FORMAT (' ALL CARDS PROCESSED',I5)
C
C     WRITE REMAINING UNSELECTED COORDINATE FILE
C
  281 CONTINUE
      READ (12,113,END=236) I1,IOGFLD,I2,COUNT,I3
      KOUNT=COUNT
      ADD =COUNT/5.0
      IDD=ADD
      BDD=IDD
      IF (ADD-BDD) 250,312,311
  311 IDD=IDD+1
C
  312 CONTINUE
      WRITE(16,115)          I1,IOGFLD,I2,KOUNT,I3
      DO 340 I=1,IDD
      READ (12,116) IA
      WRITE (16,116) IA
  340 CONTINUE
      GO TO 281
C
  236 CONTINUE
      END FILE 14
      REWIND 10
      REWIND 12
      REWIND 14
      STOP
      END
      SUBROUTINE SELFIL
C
C  THIS SUBROUTINE REWRITES THE SELECTED INPUT FILE TO ELIMINATE
C   BLANK FIELD CODES AND DUPLICATE FIELD NAMES
      DIMENSION IFIELD(40),ILFLD(2)
C
      IC=0
      IOLD=0
  200 CONTINUE
C
      READ  ( 8,110,END=235) IDSTAT,IDFLD,IFIELD,ILFLD
  110 FORMAT (I3,1X,I6,1Y,40A1,1Y,2A1)
C
      IC=IC+1
C
      IF (IDFLD) 201,201,210
C
  201 CONTINUE
      WRITE ( 6,110)          IDSTAT,IDFLD,IFIELD,ILFLD
      WRITE (6,111)
  111 FORMAT ('+',56X,'NO FIELD NUMBER')
```

```
C
      GO TO 200
C
  210 CONTINUE
      IF (IDFLD-IOLD)  203,204,205
C
  203 CONTINUE
      WRITE ( 6,110)          IDSTAT,IDFLD,IFIELD,ILFLD
      WRITE (6,112)
  112 FORMAT (*+*,56X,*FIELD NOT FOUND*)
       GO TO 200
C
  204 CONTINUE
      WRITE ( 6,110)          IDSTAT,IDFLD,IFIELD,ILFLD
      WRITE (6,114)
  114 FORMAT (*+*,56X,*DUPLICATE FIELD NAME*)
      GO TO 200
C
  205 CONTINUE
      WRITE (10,110)          IDSTAT,IDFLD,IFIELD,ILFLD
      IOLD=IDFLD
      GO TO 200
C
  235 WRITE (6,118) IC
  118 FORMAT (*  ALL INITIAL  CARDS PROCESSED*,I5)
      END FILE 10
      REWIND 8
      REWIND 10
C
      RETURN
      END
/*
//GO.FT08F001 DD DSN=OG.COPYNAY,DISP=SHR
//GO.FT10F001 DD DSN=&&TEMP,DISP=(NEW,DELETE),
//      DCB=(RECFM=FB,LRECL=80,BLKSIZE=800),
//      UNIT=SYSDA,
//      SPACE=(TRK,(2,1),RLSE)
//GO.FT12F001 DD DSN=OG.COOG,DISP=SHR
//GO.FT14F001 DD VOL=SER=CCD943,UNIT=3330,
//      DCB=(RECFM=FB,LRECL=96,BLKSIZE=6432),
//      DSN=OG.SEL.COOG,DISP=(NEW,CATLG,DELETE),
//      SPACE=(TRK,(42,3),RLSE)
//GO.FT16F001 DD VOL=SER=CCD943,UNIT=3330,
//      DCB=(RECFM=FB,LRECL=96,BLKSIZE=6432),
//      DSN=OG.NONSEL.COOG,DISP=(NEW,CATLG,DELETE),
//      SPACE=(TRK,(42,3),RLSE)
/*
//PR  EXEC PGM=IEBPTPCH
//SYSPRINT    DD SYSOUT=A
//SYSUT1      DD UNIT=3330,VOLUME=SER=CCD943,
//          DISP=SHR,DSN=OG.SEL.COOG
//SYSUT2      DD SYSOUT=A
//SYSIN       DD *
   PRINT MAXFLDS=1
   RECORD FIELD=(96)
/*
//
//
```

PROGRAM NAME: CAMPLOT1

Author: R. W. Coffin

Purpose of program: This program plots a map of the oil fields in Oklahoma with the oil field codes plotted adjacent to each field. A grid of parallels and meridians in 1° intervals is overlayed, and a title is included. The plot reproduces as closely as possible (on a first run) the Oklahoma 1:500,000-scale USGS base map.

CAM control words called: PLOTTER, LAMBERT, MAPSAL, CALIB, CETPOT, MAPBOUND, XYLIM, SAVE, OPENBOX, 1, LGRID, BOX, CENTERTK, LINEPT, SYMPT, SPLATE

Notes on selected control words:

PLOTTER

Field 2 – Used to specify pen color.

Field 6 – Specifies the number of hundreths of an inch. For example, a value of 5 means 0.05 inch.

Field 8 – Any segment greater than this length (in inches) will not be plotted.

Note that PLOTTER is called and saved several times in order to change its arguments. Here, the height of the symbols to be plotted changes several times as does the pen color.

LAMBERT

The standard parallels should be specified on the map to be duplicated. The USGS base maps of scale 1:500,000 use 33°N. and 45°N.

MAPSAL

Must be present, even though **calib** is used.

CALIB

Choose point A and point B on the map to maximize the distance between them. For example, use for point A the intersection of the easternmost longitude and the northernmost latitude, and for point B use the intersection of the westernmost longitude and the southernmost latitude. Note that for fields 1–4, values down and left from map center are negative.

CETPOT

Choose the point halfway between the eastern and western borders, and halfway between the northern and southern borders for maps which contain the entire State. The latitude/longitude coordinate of this point are easily determined by taking the ratio of the distance from this point to the nearest latitude (longitude), to the distance between the latitudes (longitudes), giving an answer as a fraction of a degree. Add or substract this number, as appropriate, to or from the latitude (longitude).

MAPBOUND

No plotting takes place for coordinates which lie outside either MAPBOUND or XYLIM. Thus, in order to plot a map with a perfectly rectangular border, specify MAPBOUND latitude/longitude boundaries which lie entirely outside of the plot window (see XYLIM).

XYLIM

Specify the physical limits of plotter (or at least dimensions that are larger than the second XYLIM), then SAVE, then specify the map boundary or window within which to plot.

OPENBOX

Optional; used here to delete plotter lines from title of map.

LGRID

In order to draw grid lines up to the plot window, specify latitude/longitude coordinates which lie outside the plot window. For example, use the same coordinates as in MAPBOUND.

BOX

Fields 1–6 – Use same arguments as for fields 1–6 of CETPOT.

Fields 7–8 – Use same arguments as for the second XYLIM (plot window).

```
//  - JOB -
/*SETUP    ttt/9RN
//*CAMPLOT1
//JOBLIB  DD DSN=SYS1.LOADLIB,DISP=SHR
//STEP1  EXEC PGM=J439,REGION=270K,TIME=12
//FT06F001  DD SYSOUT=A
//FT14F001  DD VOL=SER=ttt,DISP=(NEW,KEEP),
//    UNIT=(TAPE10,,DEFER),LABEL=(,BLP),
//    DCB=(RECFM=VS,LRECL=484,BLKSIZE=488,DEN=2)
//FT10F001  DD DSN=OG.CAM22.OKO,DISP=SHR
//FT12F001  DD DSN=OG.CAMSYM.OKOFC,DISP=SHR
//FT07F001  DD *
PLOTTER 14.,1.,,.01,8.,,,36.
LAMBERT 0.,,,,33.,,,,45.
MAPSAL 500000.
CALIB 14.63,20.45,-11.52,-22.17,37.,,,,-95.,,,,34.,,,,-101.
```

```
CETPOT 35.,21.,40.,-97.,56.,34.
MAPBOUND 33.,20.,,37.,10.,,-103.,10.,,-92.,30.
XYLIM 37.,18.
SAVE
OPENBOX 1.,-26.,-7.5,-17.,-4.5,1.
XYLIM 35.81,15.22
SAVE
1    STATE OF OKLAHOMA
  -24.38    -5.25
PLOTTER 14.,1.,.01,8.,,27.
SAVE
1    OIL FIELDS
  -22.71    -5.93
PLOTTER 14.,1.,.01,8.,,12.
SAVE
1    SCALE  1:500,000
  -22.4      -6.8
LGRID 1.,1.,.1,.1,34.,,,37.,,,-103.,,,-93.
BOX 35.,21.,40.,-97.,56.,34.,35.81,15.22
CENTERTK
PLOTTER 14.,2.,.01,8.
SAVE
LINEPT 10.,2.
PLOTTER 14.,1.,.01,8.,,8.
SAVE
SYMPT 2.,12.
SPLATE
/*
//
//
```

PROGRAM NAME: CAMPLOT2

Author: R. W. Coffin

Purpose of the program: This program plots a map of the gas fields in Oklahoma with the gas field name plotted adjacent to each field. A grid of parallels and meridians in 1° intervals is overlain, and a title is included. The plot scales down by the ratio 8:5, in accordance with the 1:500,000 scale of the USGS base map.

CAM control words called: PLOTTER, LAMBERT, MAPSAL, CETPOT, MAPBOUND, XYLIM, SAVE, OPENBOX, 1, LGRID, BOX, CENTERTK, LINEPT, SYMPT, SPLATE

Notes on selected control words:

PLOTTER

Field 2 – Used to specify pen color.

Field 6 – Specifies the number of hundreths of an inch. For example, a value of 5 means 0.05 inch.

Field 8 – Any segment greater than this length (in inches) will not be plotted.

Note that PLOTTER is called and saved several times in order to change its arguments. Here, the height of the symbols to be plotted changes several times as does the pen color.

LAMBERT

The standard parallels should be specified on the map to be duplicated. The USGS base maps of scale 1:500,000 use 33°N. and 45°N.

CETPOT

Choose the point halfway between the eastern and western borders, and halfway between the northern and southern borders for maps which contain the entire State. The latitude/longitude coordinates of this point are easily determined by taking the ratio of the distance from this point to the nearest latitude (longitude), to the distance between the latitudes (longitudes), giving an answer as a fraction of a degree. Add or subtract this number, as appropriate, to or from the latitude (longitude).

MAPBOUND

No plotting takes place for coordinates which lie outside either MAPBOUND or XYLIM. Thus, in order to plot a map with a perfectly rectangular border, specify MAPBOUND latitude/longitude boundaries which lie entirely outside of the plot window (see XYLIM).

XYLIM

Specify the physical limits of plotter (or at least dimensions that are larger than the second XYLIM), then SAVE, then specify the map boundary or window within which to plot.

OPENBOX

Optional; used here to delete plotter lines from title of map.

LGRID

In order to draw grid lines up to the plot window, specify latitude/longitude coordinates which lie outside the plot window. For example, use the same coordinates as in MAPBOUND. (Not done here.)

BOX

Fields 1–6 – Use same arguments as for fields 1–6 of CETPOT.

Fields 7–8 – Use same arguments as for the second XYLIM (plot window).

```
//  - JOB -
/*SETUP     ttt/9RN
//*CAMPLOT2
//JOBLIB  DD DSN=SYS1.LOADLIB,DISP=SHR
//STEP  EXEC PGM=J439,REGION=270K,TIME=12
//FT06F001  DD SYSOUT=A
//FT14F001  DD VOL=SER=ttt,DISP=(NEW,KEEP),
//    UNIT=(TAPELO,,DEFER),LABEL=(,BLP),
//    DCB=(RECFM=VS,LRECL=484,BLKSIZE=488,DEN=2)
//FT15F001  DD DSN=OG.CAM22.OKG,DISP=SHR
//FT16F001  DD DSN=OG.CAMSYM.OKGNM,DISP=SHR
//FT07F001  DD *
PLOTTER 14.,1.,.01,8.,,25.
LAMBERT 0.,,,33.,,,45.
MAPSAL 800000.
CETPOT 35.,21.,40.,-97.,56.,34.
MAPBOUND 33.,20.,,37.,10.,,-103.,10.,,-92.,30.
XYLIM 25.,18.
SAVE
OPENBOX 1.,-16.25,-4.69,-10.63,-2.81,1.
XYLIM 22.38,9.51
1    STATE OF OKLAHOMA
  -15.24    -3.28
PLOTTER 14.,1.,.01,8.,,17.
SAVE
1    GAS FIELDS
  -14.19    -3.71
PLOTTER 14.,1.,.01,8.,,8.
SAVE
1    SCALE  1:800,000
  -14.0     -4.25
LGRID 1.,1.,.1,.1,34.,,,37.,,,-103.,,,-93.
BOX 35.,21.,40.,-97.,56.,34.,22.38,9.51
CENTERTK
PLOTTER 14.,3.,.01,8.
SAVE
LINEPT 15.,2.
PLOTTER 14.,1.,.01,8.,,5.
SAVE
SYMPT 2.,16.
SPLATE
/*
//
//
```

EXEC_COM NAM: MAP.EC

Author: M. S. Kutsko
Purpose of the program: **map.ec** attaches the necessary files to execute the program **map**.
Data base: PDS
Computer: Honeywell Series 60 (Level 68)
Operating system: Multics
Calling sequence: ec map curdnm αβog_str
Arguments:
curdNM – county/State files in radians
αβog_str – oil/gas data in degrees
Subroutines called: **map**
Common data referenced: None
Input files: None
Output files: None
Arrays used: None
Called by: None
Error checking and reporting: None
Constants: None
Program logic:
1. File name given by the user containing the county/State coordinates in radians is attached to file *in*.
2. File name given by the user containing the oil and gas data in degrees is attached to file *input*.
3. The Dissplа routines are added to the users search rules.
4. The Pl1 program **map** is executed.

LISTING OF **map.ec**

```
io attach in vfile_ &1
io attach input vfile_ &2
asr >iml>dissplа
map
```

PROGRAM NAME: MAP

Author: M. S. Kutsko
Purpose of the program: **map** is used to plot State and county boundaries for a particular State and then plots the oil fields and gas fields located in that State. The plots can be generated on a Tektronix 4014 or 4016 graphics terminal or on a Calcomp 1055 plotter.
Data base: PDS
Computer: Honeywell Series 60 (level 68)
Operating system: Multics
Calling sequence: **map**
Arguments: None
Subroutines called:
Multics system subroutines:
ask_, ask_$ask_int, ask_$ask_line, cu_$af, ioa_, ask_$ask_cir
Calcomp subroutines:
setup_calcomp, calcmp
Tektronix subroutines:
setup_tektronix_tcs, tk30, tk120, tk960
Dissplа subroutines:
curve, dash, donepl, endpl, flatbd, grid, headin, mapgr, newpen, nochek, page, projct, reset, title
Common data referenced: None
Input files:
curdNM – referenced by file *in*
αβog_str – referenced by file *input*
tek_parms – referenced by file *parm*
calcomp_parms – referenced by file *parms*
Output files: None
Arrays used:
headin – Holds titles for the plot
xlong,ylat – Holds county/State longitude and latitude
lat,long – Holds oil/gas latitude and longitude
Called by: **map.ec**
Error checking and reporting:
Messages are written to the interactive user if any of the following errors occur:
- Subscript error in any of the arrays.
- If a user types in anything other than what is asked for in the prompt message.
- If the user asks for a FIPS code less than 1 or greater than 60.
- If the user enters a FIPS code for a State that is not in the *parm* file.

Constants: 57.29577951 – factor to convert radians to degrees.

Program logic:
1. Program acquires the date by using a Multics active function called **date**.
2. The user is asked what kind of plot is wanted, Tektronix or Calcomp.
3. The user is asked for the FIPS code of the State to be plotted.
4. The *parm* file is opened and attached to a particular segment depending whether it is a Calcomp or Tektronix plot.

   The *parm* file was created to alleviate the problem of having to change the parameters of Dissplа calls in the source program that creates the plots, and having to recompile the program every time a new State was to be plotted. The parameters in the *parm* file are those parameters that need to be changed if a new State is to be plotted. There are two files containing these needed parameters, one

for Tektronix plot parameters and one for Calcomp plot parameters. The segment names are:

>udd>Og>PFulton>oilgas>tek_parms
>udd>Og>PFulton>oilgas>calcomp_parms

The following parameter describes the order of the data in the *parm* file and the variables in **map**, that hold this data:

state_parm – The FIPS code to identify the new State.

The following parameters are used in the call to Dissplа for **page**:

pagex – The page size in inches for the x direction of the plot
pagey – The page size in inches for the y direction of the plot

The following parameters are used in the call to Dissplа for **title**:

xaxis – Defines the length of the x axis in inches
yaxis – Defines the length of the y axis in inches

The longitude and latitude limits of the plot being produced are used in the call to Dissplа for **mapgr**:

xorig – value of leftmost longitude
xmax – value of rightmost longitude
yorig – value of bottom latitude
ymax – value of top latitude

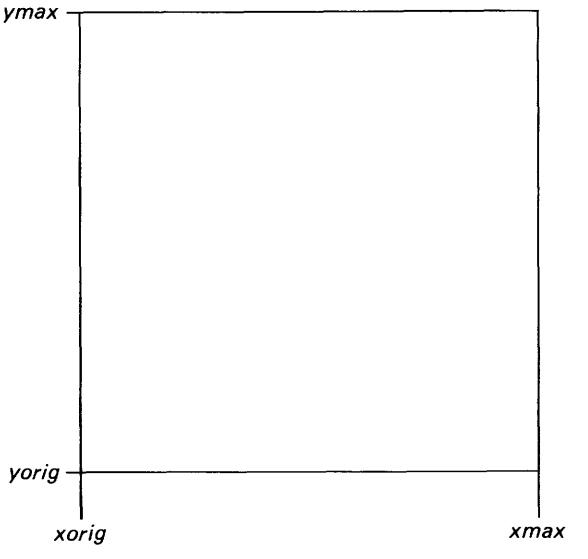
To add a new State to the *parm* file, use the following procedure at Multics command level. Type:

cwd >udd>Og>PFulton>oilgas
qx
r calcomp_parms [or r tek_parms]
a

Then type in the parameters described above in the same order, separated by commas. All parameters must be specified; none can be left out.

Example:
08,51.0,34.0,47.0,30.0,–110.0,–102.0,36.0,42.0
\f
w
q

5. The baud rate is requested and then the corresponding Dissplа routine is called.
6. If the user requests a Calcomp plot, the user is asked for the output medium for the plot (tape or segment). The user is then asked for a tape number or segment name accordingly.
7. The user is asked for the titles for the plot.
8. Dissplа routines are called to initiate the plot.
9. The State and county longitude and latitude are read in radians and are loaded into arrays. The values in the arrays are converted from radians to degrees.
10. The vector or curve is plotted.
11. The oil and gas latitude and longitude is read, loaded into arrays, and then plotted. The program continues reading and plotting the vectors until the file is exhausted.




```
COMPILATION LISTING OF SEGMENT map
Compiled by: Multics PL/I Compiler, Release 24c, of May 11, 1979
Compiled at: USGS, Reston, VA
Compiled on: 06/30/80  1543.7 edt Mon
     Options: table map

1 (subrg):map:proc;
2      dcl ask_ entry options(variable);
3      dcl ask_$ask_clr entry options(variable);
```

```
4     dcl ask_$ask_int entry options(variable);
5     dcl ask_$ask_line entry options(variable);
6     dcl calcmp entry (fixed bin(35));
7     dcl cu_$af entry (char(*),bit(1),char(*) varying,fixed bin(35));
8     dcl curve entry ((*)float bin,(*)float bin,fixed bin(35),fixed bin(35));
9     dcl dash entry ();
10    dcl donepl entry ();
11    dcl endpl entry (fixed bin(35));
12    dcl flatbd entry ();
13    dcl grid entry (fixed bin(35),fixed bin(35));
14    dcl headin entry (char(*),fixed bin(35),float bin,fixed bin(35));
15    dcl ioa_ entry options(variable);
16    dcl mapgr entry (float bin,float bin,float bin,float bin,float bin,float bin);
17    dcl newpen entry(fixed bin(35));
18    dcl nochek entry ();
19    dcl page entry (float bin,float bin);
20    dcl projct entry (char(*));
21    dcl reset entry (char(*));
22    dcl setup_calcomp entry options(variable);
23    dcl setup_tektronix_tcs entry ();
24    dcl title entry(char(*),fixed bin(35),char(*),fixed bin(35),char(*),fixed bin(35),float bin,float bin);
25    dcl tk30 entry ();
26    dcl tk120 entry ();
27    dcl tk960 entry ();
28   /*************************************************************************/
29    dcl ans char(32);
30    dcl ct  char(256);
31    dcl code fixed bin(35);
32    dcl date_in char(8) varying;
33    dcl endfile condition;
34    dcl field_length fixed bin(35);
35    dcl field_name char(44);
36    dcl fips_code fixed bin(35);
37    dcl heading(3) char(100);
38    dcl head_num fixed bin(35);
39    dcl i fixed bin(35);
40    dcl ii fixed bin(35);
41    dcl in file input;
42    dcl info char(15);
43    dcl input file input;
44    dcl j fixed bin(35);
45    dcl (j1,j2,j3,j4,j5,j6,j7) fixed dec(5);
46    dcl lat(800) float bin;
47    dcl long(800) float bin;
48    dcl n fixed bin(35);
49    dcl npts fixed bin(35);
50    dcl npts_out char(53);
51    dcl oilgas char(1);
52    dcl output char(5);
53    dcl pagex float bin;
54    dcl pagey float bin;
55    dcl parm file;
56    dcl response char(256);
57    dcl state_code char(9);
58    dcl state_parm fixed bin(35);
59    dcl subscriptrange condition;
60    dcl (xaxis,yaxis) float bin;
61    dcl (xflunk,yflunk) float bin;
62    dcl xlong(800) float bin;
63    dcl ylat(800) float bin;
64    dcl (xorig,yorig) float bin;
65    dcl (xmax,ymax) float bin;
66    on endfile(parm) begin;
67        close file(parm);
68        call ioa_ ("There isn't any data at this time for state ^i.",fips_code);
69        go to fips_prompt;
70    end;
71    on subscriptrange begin;
72        call ioa_ ("Subscript error");
73    end;
74    call cu_$af("date","0"b,date_in,code);
75 ct_prompt:
76    call ask_ ("Do you want a tektronix or calcomp plot?^/Type ""t"" for tektronix or ""c"" for calcomp.^/",ct);
77    if ct ^= "c" & ct ^= "t" then do;
78        call ioa_ ("Invalid response ^a.",ct);
79        call ask_$ask_clr;
80        go to ct_prompt;
81    end;
82 fips_prompt:
83    call ask_$ask_int("Enter two digit FIPS code number for the state you want to plot.^/",fips_code);
84    if fips_code <= 0 | fips_code > 60 then do;
85        call ioa_ ("Invalid response ^i.",fips_code);
86        call ioa_ ("FIPS code must be between 1 and 60");
```

```
 87          call ask_$ask_clr;
 88          go to fips_prompt;
 89      end;
 90      if ct = "t" then open file(parm) title("vfile_ tek_parms") input;
 91      else open file(parm) title("vfile_ calcomp_parms") input;
 92  get_parm:
 93      get file(parm) list(state_parm,pagex,pagey,xaxis,yaxis,xorig,xmax,yorig,ymax);
 94      if state_parm ^= fips_code then go to get_parm;
 95      if ct = "t" then do;
 96          call setup_tektronix_tcs;
 97          call ask_ ("Enter baud rate.^/",response);
 98          if response = "300" then call tk30;
 99          else if response = "1200" then call tk120;
100          else if response = "9600" then call tk960;
101          else call tk30;
102      end;
103      else if ct = "c" then do;
104  output_prompt:
105          call ask_ ("Type ""t"" for tape output or ""s"" for segment output.^/",ans);
106          if ans = "t" then output = "-tape";
107          else if ans = "s" then output = "-file";
108          else do;
109              call ioa_ ("Invalid response ^a.",ans);
110              call ask_$ask_clr;
111              go to output_prompt;
112          end;
113          call ask_ ("Enter tape number or segment name.^/",ans);
114          call setup_calcomp (output,"-name",ans);
115          call calcmp(16);
116      end;
117      call ioa_ ("You can enter up to three heading lines for the plot.");
118      call ioa_ ("The fourth or last line will be today's date,^/which is put in automatically by the program.");
119  head_num_prompt:
120      call ask_$ask_int ("Enter the number of heading lines you would like (from 1 to 3).^/",head_num);
121      if head_num < 0 | head_num > 3 then go to head_num_prompt;
122      call ioa_ ("Each heading line must be less than 99 characters.");
123      do i = 1 to head_num;
124  head_prompt:
125          call ask_$ask_line ("Enter heading line number ^i.^/",heading(i),i);
126          if heading(i) = " " then go to head_prompt;
127      end;
128      call flatbd;
129      call page (pagex,pagey);
130      call project ("lambe");
131      if ct = "c" then call newpen(1);
132      call title (" ",1," ",1," ",1,xaxis,yaxis);
133      do i = 1 to head_num;
134          call headin (rtrim(heading(i))||"$",100,2.,head_num + 1);
135      end;
136      call headin (date_in,8,1.,head_num+1);
137      call mapgr (xorig,1.,xmax,yorig,1.,ymax);
138      call dash;
139      call grid (1,1);
140      call reset ("dash");
141      on endfile(in) go to p001;
142      on endfile(input) go to p999;
143  p000:
144      get file(in) edit(j1,j2,j3,npts,j4,j5,j6,j7)(col(1),8(f(5,0)));
145      npts = npts - 1;
146      get file(in) edit (xflunk,yflunk,(xlong(i),ylat(i) do i = 1 to npts))
147      (col(1),6(f(12,9)));
148      do i = 1 to npts;
149          if xlong(i) > 0 then xlong(i) = -xlong(i);
150          xlong(i) = xlong(i) * 57.29577951;
151          ylat(i)  = ylat(i) * 57.29577951;
152      end;
153      call curve(xlong,ylat,npts,0);
154      go to p000;
155  p001:
156      get file(input) edit(info,state_code,oilgas,lat(1),long(1),npts,field_length,field_name)
157      (col(1),a(15),x(1),a(9),x(1),a(1),x(1),f(6,3),x(1),f(7,3),x(1),f(5),x(1),f(2),x(1),a(44));
158      if oilgas = "O" & ct = "c" then call newpen(2);
159      else if oilgas = "G" & ct = "c" then call newpen(3);
160  p01:
161      i = 0;
162      j = 0;
163  p02:
164      i=j+1;
165      j=j+5;
166      if j > npts then j = npts;
167      get file(input) edit((lat(n),long(n) do n = i to j))
168      (col(1),x(16),5(f(6,3),x(1),f(7,3),x(1)));
169      do n = i to j;
```

```
170          long(n) = -long(n);
171      end;
172      if j ^= npts then go to p02;
173      call curve(long,lat,npts,0);
174      go to p001;
175 p999:
176      call endpl(0);
177      call donepl;
178      if ct = "c" then call setup_calcomp("-reset");
179 end map;
```

EXEC_COM NAME: MAPP.EC

Author: M. S. Kutsko
Purpose of the program: **mapp.ec** attaches the necessary files to execute the program **mapp**.
Data base: PDS
Computer: Honeywell Series 60 (level 68)
Operating system: Multics
Calling sequence: ec mapp curdnm
Arguments: ***curdNM***—county/State files in radians
Subroutines called: **mapp**
Common data referenced: None
Input files: None
Output files: None
Arrays used: None
Called by: None
Error checking and reporting: None
Constants: None
Program logic:

1. File name, given by the user, containing the county/State coordinates in radians is attached to file ***in***.
2. The Dissplа routines are added to the user's search rules.
3. The Pl1 program **mapp** is executed.

LISTING OF **mapp.ec**

```
io attach in vfile__ &1
asr >iml>dissplа
mapp$map
```

PROGRAM NAME: MAPP

Author: M. S. Kutsko
Purpose of program: **mapp** is used to plot State and county boundaries for a particular State. The plots can be generated on tektronix 4014 or 4016 graphics terminal or on a Calcomp 1055 plotter.
Data base: PDS
Computer: Honeywell Series 60 (level 68)
Operating system: Multics
Calling sequence: **mapp**
Arguments: None
Subroutines called:
Multics system subroutines:
ask__, ask__$ask__int, ask__$ask__line, cu__$af, ioa__, ask__$ask__clr
Calcomp subroutines:
setup__calcomp, calcmp
Tektronix subroutines:
setup__tektronix__tcs, tk30, tk120, tk960
Dissplа subroutines:
curve, dash, donepl, endpl, flatbd, grid, headin, mapgr, newpen, nochek, page, projct, reset, title
Common data referenced: None
Input files:
curdNM—referenced by file ***in***
tek__parms—referenced by file ***parm***
calcomp__parms—referenced by file ***parm***
Output files: None
Arrays used:
headin—holds title for the plot
xlong,ylat—holds county/State longitude and latitude
Called by: **mapp.ec.**
Error checking and reporting:
Messages are written to the interactive user if any of the following errors occur:

- Subscript error in any of the arrays.
- The user types in anything other than what is asked for in the prompt message.
- The user asks for a FIPS code less than 1 or greater than 60.
- The user enters a FIPS code for a State that is not in the ***parm*** file.

Constants: 57.29577951—factor to convert radians to degrees.
Program logic:

1. Program acquires the date by using a Multics active function called **date.**
2. The user is asked what kind of plot is wanted, Tektronix or Calcomp.
3. The user is asked for the FIPS code of the State to be plotted.
4. The ***parm*** file is opened and attached to a particular segment depending on whether it is a Calcomp or Tektronix plot.
   The ***parm*** file was created to alleviate the problem of having to change the parameters of Dissplа calls in the source program that creates the plots,

and having to recompile the program every time a new State was to be plotted. The parameters in the parm file are those parameters that need to be changed if a new State is to be plotted. There are two files containing these needed parameters, one for Tektronix plot parameters and one for Calcomp plot parameters. The segment names are:

>udd>Og>PFulton>oilgas>tek_parms
>udd>Og>PFulton>oilgas>calcomp_parms

The following parameter describes the order of the data in the parm file and the variables in **map** that hold this data:

state_parm – The FIPS code to identify the new State

The following parameters are used in the call to Disspla for **page**:

pagex – The page size in inches for the x direction of the plot
pagey – The page size in inches for the y direction of the plot

The following parameters are used in the call to Disspla for **title**:

xaxis – Defines the length of the x axis in inches
yaxis – Defines the length of the y axis in inches

The longitude and latitude limits of the plot being produced are used in the call to Disspla for **mapgr**:

xorig – value of leftmost longitude
xmax – value of rightmost longitude
yorig – value of bottom latitude
ymax – value of top latitude

To add a new State to the *parm* file, use the following procedure at Multics command level. Type:

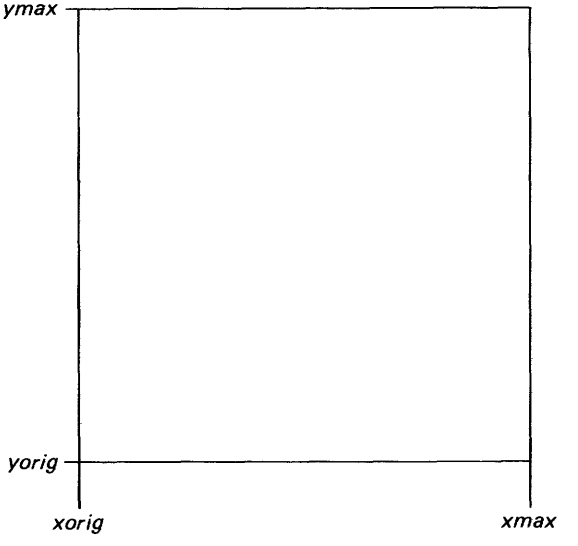
```
cwd >udd>Og>PFulton>oilgas
qx
r calcomp_parms [or r tek_parms]
a
```

Then type in the parameters described above in the same order, separated by commas. All parameters must be specified; none can be left out.

Example:

```
08,51.0,34.0,47.0,30.0,-110.0,-102.0,36.0,42.0
\f
w
q
```

5. The baud rate is requested, and then the corresponding Disspla routine is called.
6. If the user requests a Calcomp plot, he is asked for the output medium for the plot (tape or segment). The user is then asked for a tape number or segment name accordingly.
7. The user is asked for the titles for the plot.
8. Disspla routines are called to initiate the plot.
9. The State and county longitude and latitude are read in radians and are loaded into arrays. The values in the arrays are converted from radians to degrees.
10. The vector or curve is plotted.




```
COMPILATION LISTING OF SEGMENT mapp
Compiled by: Multics PL/I Compiler, Release 24c, of May 11, 1979
Compiled at: USGS, Reston, VA
Compiled on: 12/12/80  1523.8 est Fri
    Options: table map

1 (subrg):map:proc;
2     dcl ask_ entry options(variable);
3     dcl ask_$ask_clr entry options(variable);
4     dcl ask_$ask_int entry options(variable);
5     dcl ask_$ask_line entry options(variable);
6     dcl calcmp entry (fixed bin(35));
```

```
 7     dcl cu_$af entry (char(*),bit(1),char(*) varying,fixed bin(35));
 8     dcl curve entry ((*)float bin,(*)float bin,fixed bin(35),fixed bin(35));
 9     dcl dash entry ();
10     dcl donepl entry ();
11     dcl endpl entry (fixed bin(35));
12     dcl flatbd entry ();
13     dcl grid entry (fixed bin(35),fixed bin(35));
14     dcl headin entry (char(*),fixed bin(35),float bin,fixed bin(35));
15     dcl ioa_ entry options(variable);
16     dcl mapgr entry (float bin,float bin,float bin,float bin,float bin,float bin);
17     dcl newpen entry(fixed bin(35));
18     dcl nochek entry ();
19     dcl page entry (float bin,float bin);
20     dcl project entry (char(*));
21     dcl reset entry (char(*));
22     dcl setup_calcomp entry options(variable);
23     dcl setup_tektronix_tcs entry ();
24     dcl title entry(char(*),fixed bin(35),char(*),fixed bin(35),char(*),fixed bin(35),float bin,float bin);
25     dcl tk30 entry ();
26     dcl tk120 entry ();
27     dcl tk960 entry ();
28   /*************************************************************************/
29     dcl ans char(32);
30     dcl ct  char(256);
31     dcl code fixed bin(35);
32     dcl date_in char(8) varying;
33     dcl endfile condition;
34     dcl field_npts char(53);
35     dcl fips_code fixed bin(35);
36     dcl heading(3) char(100);
37     dcl head_num fixed bin(35);
38     dcl i fixed bin(35);
39     dcl ii fixed bin(35);
40     dcl in file input;
41     dcl info char(15);
42     dcl j fixed bin(35);
43     dcl (j1,j2,j3,j4,j5,j6,j7) fixed dec(5);
44     dcl n fixed bin(35);
45     dcl npts fixed bin(35);
46     dcl npts_out char(53);
47     dcl oilgas char(1);
48     dcl output char(5);
49     dcl pagex float bin;
50     dcl pagey float bin;
51     dcl parm file;
52     dcl response char(256);
53     dcl state_code char(9);
54     dcl state_parm fixed bin(35);
55     dcl subscriptrange condition;
56     dcl (xaxis,yaxis) float bin;
57     dcl (xflunk,yflunk) float bin;
58     dcl xlong(800) float bin;
59     dcl ylat(800) float bin;
60     dcl (xorig,yorig) float bin;
61     dcl (xmax,ymax) float bin;
62     on endfile(parm) begin;
63         close file(parm);
64         call ioa_ ("There isn't any data at this time for state ^i.",fips_code);
65         go to fips_prompt;
66     end;
67     on subscriptrange begin;
68         call ioa_ ("Subscript error");
69     end;
70     call cu_$af("date","0"b,date_in,code);
71 ct_prompt:
72     call ask_ ("Do you want a tektronix or calcomp plot?^/Type ""t"" for tektronix or ""c"" for calcomp.^/",ct);
73     if ct ^= "c" & ct ^= "t" then do;
74         call ioa_ ("Invalid response ^a.",ct);
75         call ask_$ask_clr;
76         go to ct_prompt;
77     end;
78 fips_prompt:
79     call ask_$ask_int("Enter two digit FIPS code number for the state you want to plot.^/",fips_code);
80     if fips_code <= 0 | fips_code > 60 then do;
81         call ioa_ ("Invalid response ^i.",fips_code);
82         call ioa_ ("FIPS code must be between 1 and 60");
83         call ask_$ask_clr;
84         go to fips_prompt;
85     end;
86     if ct = "t" then open file(parm) title("vfile_ tek_parms") input;
87     else open file(parm) title("vfile_ calcomp_parms") input;
88 get_parm:
89     get file(parm) list(state_parm,pagex,pagey,xaxis,yaxis,xorig,xmax,yorig,ymax);
90     if state_parm ^= fips_code then go to get_parm;
91     if ct = "t" then do;
92         call setup_tektronix_tcs;
93         call ask_ ("Enter baud rate.^/",response);
```

```
94          if response = "300" then call tk30;
95          else if response = "1200" then call tk120;
96          else if response = "9600" then call tk960;
97          else call tk30;
98      end;
99      else if ct = "c" then do;
100 output_prompt:
101         call ask_ ("Type ""t"" for tape output or ""s"" for segment output.^/",ans);
102         if ans = "t" then output = "-tape";
103         else if ans = "s" then output = "-file";
104         else do;
105             call ioa_ ("Invalid response ^a.",ans);
106             call ask_$ask_clr;
107             go to output_prompt;
108         end;
109         call ask_ ("Enter tape number or segment name.^/",ans);
110         call setup_calcomp (output,"-name",ans);
111         call calcmp(16);
112     end;
113     call ioa_ ("You can enter up to three heading lines for the plot.");
114     call ioa_ ("The fourth or last line will be today's date,^/which is put in automatically by the program.");
115 head_num_prompt:
116     call ask_$ask_int ("Enter the number of heading lines you would like (from 1 to 3).^/",head_num);
117     if head_num < 0 | head_num > 3 then go to head_num_promot;
118     call ioa_ ("Each heading line must be less than 99 characters.");
119     do i = 1 to head_num;
120 head_prompt:
121         call ask_$ask_line ("Enter heading line number ^i.^/",heading(i),i);
122         if heading(i) = " " then go to head_prompt;
123     end;
124     call flatbd;
125     call page (pagex,pagey);
126     call projct ("cylin");
127     if ct = "c" then call newpen(1);
128     call title (" ",1," ",1," ",1,xaxis,yaxis);
129     do i = 1 to head_num;
130         call headin (rtrim(heading(i))||"$",100,2.,head_num + 1);
131     end;
132     call headin (date_in,8,1.,head_num+1);
133     call mapgr (xorig,1.,xmax,yorig,1.,ymax);
134     call dash;
135     call grid (1,1);
136     call reset ("dash");
137     on endfile(in) go to p999;
138 p000:
139     get file(in) edit(j1,j2,j3,npts,j4,j5,j6,j7)(col(1),8(f(5,0)));
140     npts = npts - 1;
141     get file(in) edit (xflunk,yflunk,(xlong(i),ylat(i) do i = 1 to npts))
142     (col(1),6(f(12,9)));
143     do i = 1 to npts;
144         if xlong(i) > 0 then xlong(i) = -xlong(i);      /*  Pat this line is the fix for - long  */
145         xlong(i) = xlong(i) * 57.29577951;
146         ylat(i)  = ylat(i) * 57.29577951;
147     end;
148     call curve(xlong,ylat,npts,0);
149     go to p000;
150 p999:
151     call endpl(0);
152     call donepl;
153     if ct = "c" then call setup_calcomp("-reset");
154 end map;
```

EXEC_COM NAME: MAPOG.EC

Author: M. S. Kutsko
Purpose of the program: **mapog.ec** attaches the necessary files to execute the program **mapog**.
Data base: PDS
Computer: Honeywell Series 60 (level 68)
Operating system: Multics
Calling sequence: ec mapog curdnm αβog_str
Arguments:
curdNM – county/State files in radians
αβog_str – oil/gas data in degrees
Subroutines called: **mapog**
Common data referenced: None
Input files: None
Output files: None
Arrays used: None
Called by: None
Error checking and reporting: None
Constants: None
Program logic:
1. File name given by the user containing the county/State coordinates in radians is attached to file *in*.
2. File name given by the user containing the oil/gas data in degrees is attached to file *input*.
3. The Dissplа routines are added to the user's search rules.
4. The Pl1 program **mapog** is executed.

LISTING OF **mapog.ec**

```
io attach in vfile_ &1
io attach input vfile_ &2
asr >iml>dissplа
mapog
```

PROGRAM NAME: MAPOG

Author: M. S. Kutsko
Purpose of program: **mapog** is used to plot State and county boundaries for a particular State and then plots either all the oil fields or all the gas fields located in that State. The plots can be generated on a Tektronix 4014 or 4016 graphics terminal or on a Calcomp 1055 plotter.
Data base: PDS
Computer: Honeywell Series 60 (level 68)
Operating system: Multics
Calling sequence: **mapog**
Arguments: None
Subroutines called:
Multics system subroutines:
ask_, ask_$ask_int, ask_$ask_line, cu_$af, ioa_, ask_$ask_clr
Calcomp subroutines:
setup_calcomp, calcmp
Tektronix subroutines:
setup_tektronix_tcs, tk30, tk120, tk960
Dissplа subroutines:
curve, dash, donepl, endpl, flatbd, grid, headin, mapgr, newpen, nochek, page, projct, reset, titie
Common data referenced: None
Input files:
curdNM – referenced by file *in*
αβog_str – referenced by file *input*
tek_parms – referenced by file *parm*
calcomp_parms – referenced by file *parm*
Output files: None
Arrays used:
headin – holds titles for the plot
xlong,ylat – holds county/State longitude and latitude
lat,long – holds oil/gas latitude and longitude
Called by: **mapog.ec.**
Error checking and reporting:
Messages are written to the interactive user if any of the following errors occur:
- Subscript error in any of the arrays.
- If a user types in anything other than what is asked for in the prompt messages.
- If the user asks for a FIPS code less than 1 or greater than 60.
- If the user enters a FIPS code for a State that is not in the *parm* file.

Constants: 57.29577951 – factor to convert radians to degrees.

Program logic:
1. Program acquires the date by using a Multics active function called **date**.
2. The user is asked what kind of plot is wanted, Tektronix or Calcomp.
3. The user is asked for the FIPS code of the State to be plotted.
4. The *parm* file is opened and attached to a particular segment depending on whether it is a Calcomp or Tektronix plot.
   The *parm* file was created to alleviate the problem of having to change the parameters of Dissplа calls in the source program that creates the plots, and having to recompile the program every time a new State was to be plotted. The parameters in the parm file are those parameters that need to be changed if a new State is to be plotted. There are two files containing these needed parameters, one

for Tektronix plot parameters and one for Calcomp plot parameters. The segment names are:

>udd>Og>PFulton>oilgas>tek_parms
>udd>Og>PFulton>oilgas>calcomp_parms

The following parameter describes the order of the data in the parm file and the variables in **map** that hold this data:

state_parm – The FIPS code to identify the new State.

The following parameters are used in the call to Dissplа for **page**:

pagex – The page size in inches for the x direction of the plot
pagey – The page size in inches for the y direction of the plot

The following parameters are used in the call to Dissplа for **title**:

xaxis – Defines the length of the x axis in inches
yaxis – Defines the length of the y axis in inches

The longitude and latitude limits of the plot being produced are used in the call to Dissplа for **mapgr**:

xorig – value of leftmost longitude
xmax – value of rightmost longitude
yorig – value of bottom latitude
ymax – value of top latitude

To add a new State to the *parm* file, use the following procedure at Multics command level. Type:

```
cwd >udd>Og>PFulton>oilgas
qx
r calcomp_parms [or r tek_parms]
a
```

Then type in the parameters described above in the same order, separated by commas. All parameters must be specified; none can be left out.

Example:

```
08,51.0,34.0,47.0,30.0,-110.0,-102.0,36.0,42.0
\f
w

q
```

5. The baud rate is requested, and then the corresponding Dissplа routine is called.

6. If the user requests a Calcomp plot, the user is asked for the output medium for the plot (tape or segment). The user is then asked for a tape number or segment name accordingly.

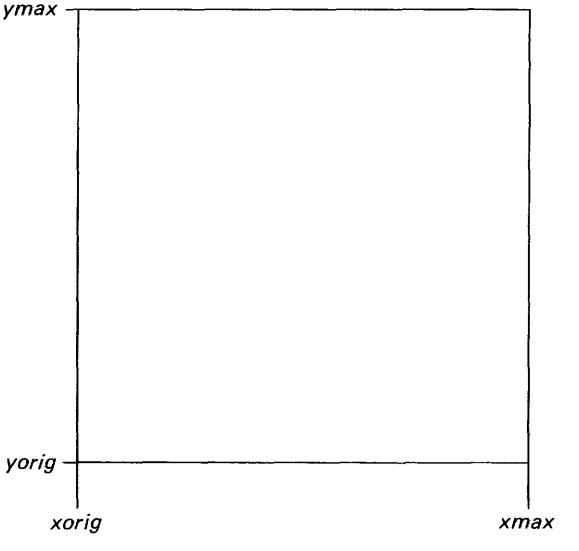
7. The user is asked for the titles for the plot and whether oil fields or gas fields are to be plotted.

8. Dissplа routines are called to initiate the plot.

9. The State and county longitude and latitude are read in radians and are loaded into arrays. The values in the arrays are converted from radians to degrees.

10. The vector or curve is plotted.

11. The oil and gas data is read. Depending on what the user wants to plot, either oil fields or gas fields, the appropriate latitude and longitude is read, loaded into arrays, and then plotted. The State code is also plotted to designate the field. The program continues reading and plotting the vectors until the file is exhausted.



```
(subrg):mapog:proc;
    dcl ask_ entry options(variable);
    dcl ask_$ask_clr entry options(variable);
    dcl ask_$ask_int entry options(variable);
    dcl ask_$ask_line entry options(variable);
    dcl calcmp entry (fixed bin(35));
    dcl cu_$af entry (char(*),bit(1),char(*) varying,fixed bin(35));
    dcl curve entry ((*)float bin,(*)float bin,fixed bin(35),fixed bin(35));
    dcl dash entry ();
    dcl donepl entry ();
    dcl endpl entry (fixed bin(35));
    dcl flatbd entry ();
    dcl grid entry (fixed bin(35),fixed bin(35));
    dcl headin entry (char(*),fixed bin(35),float bin,fixed bin(35));
    dcl height entry (float bin);
    dcl ioa_ entry options(variable);
    dcl mapgr entry (float bin,float bin,float bin,float bin,float bin,float bin);
    dcl newpen entry(fixed bin(35));
    dcl nochek entry ();
    dcl page entry (float bin,float bin);
    dcl projct entry (char(*));
    dcl reset entry (char(*));
    dcl rlmess entry (char(*),fixed bin(35),float bin,float bin);
    dcl setup_calcomp entry options(variable);
    dcl setup_tektronix_tcs entry ();
    dcl title entry(char(*),fixed bin(35),char(*),fixed bin(35),char(*),fixed bin(35),float bin,float bin);
    dcl tk30 entry ();
    dcl tk120 entry ();
    dcl tk960 entry ();
  /*****************************************************************************/
    dcl ans char(32);
    dcl ct  char(256);
    dcl code fixed bin(35);
    dcl date_in char(8) varying;
    dcl endfile condition;
    dcl field_length fixed bin(35);
    dcl field_name char(44);
    dcl fips_code fixed bin(35);
    dcl heading(3) char(60);
    dcl head_num fixed bin(35);
    dcl i fixed bin(35);
    dcl ii fixed bin(35);
    dcl in file input;
    dcl info char(15);
    dcl input file input;
    dcl j fixed bin(35);
    dcl (j1,j2,j3,j4,j5,j6,j7) fixed dec(5);
    dcl lat(800) float bin;
    dcl long(800) float bin;
    dcl n fixed bin(35);
    dcl npts fixed bin(35);
    dcl npts_out char(53);
    dcl og char(256);
    dcl oilgas char(1);
    dcl output char(5);
    dcl pagex float bin;
    dcl pagey float bin;
    dcl parm file;
    dcl response char(256);
    dcl state_code char(6);
    dcl state_parm fixed bin(35);
    dcl subscriptrange condition;
    dcl (xaxis,yaxis) float bin;
    dcl (xflunk,yflunk) float bin;
    dcl xlong(800) float bin;
    dcl ylat(800) float bin;
    dcl (xorig,yorig) float bin;
    dcl (xmax,ymax) float bin;
    on endfile(parm) begin;
        close file(parm);
        call ioa_ ("There isn't any data at this time for state ^i.",fips_code);
        go to fips_prompt;
    end;
    on subscriptrange begin;
        call ioa_ ("Subscript error");
    end;
    call cu_$af("date","0"b,date_in,code);
ct_prompt:
    call ask_ ("Do you want a tektronix or calcomp plot?^/Type ""t"" for tektronix or ""c"" for calcomp.^/",ct);
    if ct ^= "c" & ct ^= "t" then do;
        call ioa_ ("Invalid response ^a.",ct);
        call ask_$ask_clr;
        go to ct_prompt;
    end;
```

```
fips_prompt:
    call ask_$ask_int("Enter two digit FIPS code number for the state you want to plot.^/",fips_code);
    if fips_code <= 0 | fips_code > 60 then do;
        call ioa_ ("Invalid response ^i.",fips_code);
        call ioa_ ("FIPS code must be between 1 and 60");
        call ask_$ask_clr;
        go to fips_prompt;
    end;
    if ct = "t" then open file(parm) title("vfile_ tek_parms") input;
    else open file(parm) title("vfile_ calcomp_parms") input;
get_parm:
    get file(parm) list(state_parm,pagex,pagey,xaxis,yaxis,xorig,xmax,yorig,ymax);
    if state_parm ^= fips_code then go to get_parm;
    if ct = "t" then do;
        call setup_tektronix_tcs;
        call ask_ ("Enter baud rate.^/",response);
        if response = "300" then call tk30;
        else if response = "1200" then call tk120;
        else if response = "9600" then call tk960;
        else call tk30;
    end;
    else if ct = "c" then do;
output_prompt:
        call ask_ ("Type ""t"" for tape output or ""s"" for segment output.^/",ans);
        if ans = "t" then output = "-tape";
        else if ans = "s" then output = "-file";
        else do;
            call ioa_ ("Invalid response ^a.",ans);
            call ask_$ask_clr;
            go to output_prompt;
        end;
        call ask_ ("Enter tape number or segment name.^/",ans);
        call setup_calcomp (output,"-name",ans);
        call calcmp(16);
    end;
    call ioa_ ("You can enter up to three heading lines for the plot.");
    call ioa_ ("The fourth or last line will be today's date,^/which is put in automatically by the program.");
head_num_prompt:
    call ask_$ask_int ("Enter the number of heading lines you would like (from 1 to 3).^/",head_num);
    if head_num < 0 | head_num > 3 then go to head_num_prompt;
    call ioa_ ("Each heading line must be less than 59 characters.");
    do i = 1 to head_num;
head_prompt:
        call ask_$ask_line ("Enter heading line number ^i.^/",heading(i),i);
        if heading(i) = " " then go to head_prompt;
    end;
og_prompt:
    call ask_ ("Do you want to plot oil or gas fields?^/Type ""o"" for oil or ""g"" for gas.^/",og);
    if og ^= "o" & og ^= "g" then do;
        call ioa_ ("invalid response ^a.",og);
        call ask_$ask_clr;
        go to og_prompt;
    end;
    if og = "o" then og = "0";
    else og = "G";
    call flatbd;
    call page (pagex,pagey);
    call projct ("lambe");
    if ct = "c" then call newpen(1);
    call title (" ",1," ",1," ",1,xaxis,yaxis);
    do i = 1 to head_num;
        call headin (rtrim(heading(i))||"$",60,2.,head_num + 1);
    end;
    call headin (date_in,8,1.,head_num+1);
    call mapqr (xorig,1.,xmax,yorig,1.,ymax);
    call dash;
    call grid (1,1);
    call reset ("dash");
    on endfile(in) begin;
       call newpen(2);
       call height(0.05);
       go to p001;
    end;
    on endfile(input) go to p999;
p000:
    get file(in) edit(j1,j2,j3,npts,j4,j5,j6,j7)(col(1),8(f(5,0)));
    npts = npts - 1;
    get file(in) edit (xflunk,yflunk,(xlong(i),ylat(i) do i = 1 to npts))
    (col(1),6(f(12,9)));
    do i = 1 to npts;
        if xlong(i) > 0 then xlong(i) = -xlong(i);
        xlong(i) = xlong(i) * 57.29577951;
        ylat(i)  = ylat(i) * 57.29577951;
    end;
```

```
    call curve(xlong,ylat,npts,0);
    go to p000;
p001:
    get file(input) edit(info,state_code,oilgas,lat(1),long(1),npts,field_length,field_name)
    (col(1),a(15),x(4),a(6),x(1),a(1),x(1),f(6,3),x(1),f(7,3),x(1),f(5),x(1),f(2),x(1),a(44));
    if og = oilgas then call rlmess(state_code,6,-long(1),lat(1));
p01:
    i = 0;
    j = 0;
p02:
    i=j+1;
    j=j+5;
    if j > npts then j = npts;
    get file(input) edit((lat(n),long(n) do n = i to j))
    (col(1),x(16),5(f(6,3),x(1),f(7,3),x(1)));
    do n = i to j;
        long(n) = -long(n);
    end;
    if j ^= npts then go to p02;
    if og = oilgas then
    call curve(long,lat,npts,0);
    go to p001;
p999:
    call endpl(0);
    call donepl;
    if ct = "c" then call setup_calcomp("-reset");
end mapog;
```

EXEC_COM NAME: MAPIT2.EC

Author: M. S. Kutsko
Purpose of the program: **mapit2.ec** attaches the necessary files to execute the program **mapit2**.
Data base: PDS
Computer: Honeywell Series 60 (level 68)
Operating system: Multics
Calling sequence: ec mapit2 curdnm αβog_str
Arguments:
curdNM – county/State files in radians
αβog_str – oil/gas data in degrees
Subroutine called: **mapit2**
Common data referenced: None
Input files: None
Output files: None
Arrays used: None
Called by: None
Error checking and reporting: None
Constants: None
Program logic:

1. File name given by the user containing the county/State coordinates in radians is attached to file ***in***.
2. File name given by the user containing the oil/gas data in degrees is attached to file ***input***.
3. File name ***data1*** containing oil/gas information about Fort Chaffee area is attached to file ***oilgasin***.
4. File name ***year7074*** containing oil and gas information for a certain span of years about the Fort Chaffee area is attached to file ***oilgasyr***.
5. File name ***mess***, which would contain any error messages produced from **mapit2**, is attached to file ***mess***.
6. The Dissppla routines are added to the users search rules.
7. The Pl1 program **mapit2** is executed.

LISTING OF **mapit2.ec**

```
io attach in vfile_ &1
io attach input vfile_ &2
io attach oilgasin vfile_ data1
io attach oilgasyr vfile_ year7074
io attach mes vfile_ mess
asr >iml>dissppla
mapit2$map
```

PROGRAM NAME: MAPIT2

Author: M. S. Kutsko

Purpose of program: **mapit2** is used to plot State and county boundaries for Arkansas and then plots the oil fields and gas fields located in the Fort Chaffee area. The plots can be generated on a Tektronix 4014 or 4016 graphics terminal or on a Calcomp 1055 plotter.

Data base: PDS

Computer: Honeywell Series 60 (level 68)

Operating system: Multics

Calling sequence: **mapit2**

Arguments: None

Subroutines called:

Multics system subroutines:
ask_, ask_$ask_int, ask_$ask_line, cu_$af, ioa_, ask_$ask_clr

Calcomp subroutines:
setup_calcomp, calcmp

Tektronix subroutines:
setup_tektronix_tcs, tk30, tk120, tk960 nhance

Dissplà subroutines:
curve, dash, donepl, endpl, flatbd, grid, headin, mapgr, newpen, nochek, page, projct, reset, title rlmess

Common data referenced: None

Input files:
curdNM – referenced by file *in*
αβog_str – referenced by file *input*
tek_parms – referenced by file *parm*
calcomp_parms – referenced by file *parm*
data1 – referenced by file *oilgasin*
year7074 – referenced by file *oilgasyr*

Output files: *mess* – error message file

Arrays used:
headin – Holds titles for the plot
xlong,ylat – Holds county/State longitude and latitude
lat,long – Holds oil/gas latitude and longitude

Called by: **mapit2.ec.**

Error checking and reporting:

Messages are written to the interactive user if any of the following errors occur:
- Subscript error in any of the arrays.
- The user types in anything other than what is asked for in the prompt messages.
- The user asks for a FIPS code less than 1 or greater than 60.
- The user enters a FIPS code for a State that is not in the *parm* file.

Messages written to *mess* file if any of the following errors occur:
- If field *id* is blank in *oilgasin* file.
- If no record was found in the oil/gas data.

Constants: 57.29577951 – factor to convert radians to degrees.

Program logic:

1. Program acquires the date by using a Multics active function called **date**.
2. The user is asked what kind of plot is wanted, Tektronix or Calcomp.
3. The user is asked for the FIPS code of the State to be plotted.
4. The *parm* file is opened and attached to a particular segment depending on whether it is a Calcomp or Tektronix plot.

   The *parm* file was created to alleviate the problem of having to change the parameters of Dissplà calls in the source program that creates the plots, and having to recompile the program every time a new State was to be plotted. The parameters in the *parm* file are those parameters that need to be changed if a new State is to be plotted. There are two files containing these needed parameters, one for Tektronix plot parameters and one for Calcomp plot parameters. The segment names are:

   >udd>Og>PFulton>oilgas>tek_parms
   >udd>Og>PFulton>oilgas>calcomp_parms

   The following parameter describes the order of the data in the parm file and the variables in **map** that hold this data:

   state_parm – The FIPS code to identify the new State.

   The following parameters are used in the call to Dissplà for **page**:

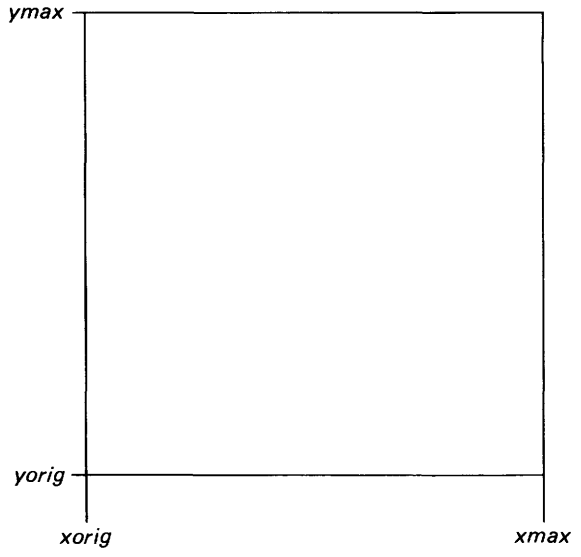
   pagex – The page size in inches for the x direction of the plot
   pagey – The page size in inches for the y direction of the plot

   The following parameters are used in the call to Dissplà for **title**:

   xaxis – Defines the length of the x axis in inches
   yaxis – Defines the length of the y axis in inches

   The longitude and latitude limits of the plot being produced are used in the call to Dissplà for **mapgr**:

   xorig – value of leftmost longitude
   xmax – value of rightmost longitude
   yorig – value of bottom latitude
   ymax – value of top latitude



To add a new State to the parm file, use the following procedure at Multics command level. Type:

```
cwd >udd>Og>PFulton>oilgas
qx
r calcomp_parms [or r tek_parms]
a
```

Then type in the parameters described above in the same order, separated by commas. All parameters must be specified; none can be left out.

Example:

```
08,51.0,34.0,47.0,30.0,-110.0,-102.0,36.0,42.0
\f
w
q
```

5. The baud rate is requested, and then the corresponding Disspla routine is called. **nhance** was used in this version for better resolution of the plot.
6. If the user requests a Calcomp plot, the user is asked for the output medium for the plot (tape or segment). The user is then asked for a tape number or segment name accordingly.
7. The user is asked for the titles for the plot.
8. Disspla routines are called to initiate the plot.
9. The State and county longitude and latitude are read in radians and are loaded into arrays. The values in the arrays are converted from radians to degrees. ***xorig, xmax, yorig, ymax*** were set to specific values in this version to zero in on the Fort Chaffee area of Arkansas.
10. The vector or curve is plotted.
11. The ***oilgasyr*** file is read in to plot only a select few oil fields/gas fields in the Fort Chaffee area. This file contains records for a certain range of years that the oil field/gas field was discovered. Once a record is read in, it is compared to the ***oilgasin*** file to see if an oil field/gas field in the Fort Chaffee area exists. Once this is done, the oil and gas latitude and longitude are read, loaded into arrays, and then plotted. The name of the oil field/gas field is also plotted and oil fields/gas fields are plotted in different colors and are shaded. This process continues until all fields are plotted.

```
(subrg):map:proc;
    dcl ask_ entry options(variable);
    dcl ask_$ask_clr entry options(variable);
    dcl ask_$ask_int entry options(variable);
    dcl ask_$ask_line entry options(variable);
    dcl calcmp entry (fixed bin(35));
    dcl cu_$af entry (char(*),bit(1),char(*) varying,fixed bin(35));
    dcl curve entry ((*)float bin,(*)float bin,fixed bin(35),fixed bin(35));
    dcl dash entry ();
    dcl donepl entry ();
    dcl endpl entry (fixed bin(35));
    dcl flatbd entry ();
    dcl grid entry (fixed bin(35),fixed bin(35));
    dcl headin entry (char(*),fixed bin(35),float bin,fixed bin(35));
    dcl ioa_ entry options(variable);
    dcl mapgr entry (float bin,float bin,float bin,float bin,float bin);
    dcl newpen entry(fixed bin(35));
    dcl nochek entry ();
    dcl page entry (float bin,float bin);
    dcl projct entry (char(*));
    dcl reset entry (char(*));
    dcl rlmess entry (char(*),fixed bin(35),float bin, float bin);
    dcl setup_calcomp entry options(variable);
    dcl setup_tektronix_tcs entry ();
    dcl shade entry ((*)float bin,(*)float bin,fixed bin(35),float bin,float bin,fixed bin(35),(*)fixed bin(35),fixed bin(35));
    dcl title entry(char(*),fixed bin(35),char(*),fixed bin(35),char(*),fixed bin(35),float bin,float bin);
    dcl tk30 entry ();
    dcl tk120 entry ();
    dcl nhance entry (fixed bin(35));
    dcl tk960 entry ();
  /*************************************************************************/
```

```
    dcl ans char(32);
    dcl ct  char(256);
    dcl code fixed bin(35);
    dcl date_in char(8) varying;
    dcl depth char(5);
    dcl endfile condition;
    dcl field_check char(9) init(" ");
    dcl field_ck_yr char(9);
    dcl field_name char(45);
    dcl field_nm char(40);
    dcl field_nm_yr char(40);
    dcl field_npts char(53);
    dcl fips_code fixed bin(35);
    dcl heading(3) char(100);
    dcl head_num fixed bin(35);
    dcl i fixed bin(35);
    dcl ii fixed bin(35);
    dcl in file input;
    dcl info char(15);
    dcl input file input;
    dcl j fixed bin(35);
    dcl (j1,j2,j3,j4,j5,j6,j7) fixed dec(5);
    dcl last_field_ck char(9) init(" ");
    dcl lat(800) float bin;
    dcl long(800) float bin;
    dcl n fixed bin(35);
    dcl name_sw bit(1) init("0"b);
    dcl npts fixed bin(35);
    dcl npts_out char(53);
    dcl mess file;
    dcl oilgas char(1);
    dcl oilgasin file;
    dcl oilgasyr file input;
    dcl output char(5);
    dcl pagex float bin;
    dcl pagey float bin;
    dcl parm file;
    dcl response char(256);
    dcl series_nm char(15);
    dcl shade_sw bit(1) init("0"b);
    dcl state_code char(9);
    dcl state_in char(2);
    dcl state_parm fixed bin(35);
    dcl subscriptrange condition;
    dcl (xaxis,yaxis) float bin;
    dcl (xflunk,yflunk) float bin;
    dcl xlong(800) float bin;
    dcl ylat(800) float bin;
    dcl work(500) fixed bin(35);
    dcl year_abandoned char(4);
    dcl year_discovered char(4);
    dcl (xorig,yorig) float bin;
    dcl (xmax,ymax) float bin;
    on endfile(parm) begin;
        close file(parm);
        call ioa_ ("There isn't any data at this time for state ^i.",fips_code);
        go to fips_prompt;
    end;
    on endfile(oilgasyr) begin;
        close file(oilgasyr);
        goto head;
    end;
    on subscriptrange begin;
        call ioa_ ("Subscript error");
    end;
    call cu_$af("date","0"b,date_in,code);
ct_prompt:
    call ask_ ("Do you want a tektronix or calcomp plot?^/Type ""t"" for tektronix or ""c"" for calcomp.^/",ct);
    if ct ^= "c" & ct ^= "t" then do;
        call ioa_ ("Invalid response ^a.",ct);
        call ask_$ask_clr;
        go to ct_prompt;
    end;
fips_prompt:
    call ask_$ask_int("Enter two digit FIPS code number for the state you want to plot.^/",fips_code);
    if fips_code <= 0 | fips_code > 60 then do;
        call ioa_ ("Invalid response ^i.",fips_code);
        call ioa_ ("FIPS code must be between 1 and 60");
        call ask_$ask_clr;
        go to fips_prompt;
    end;
    if ct = "t" then open file(parm) title("vfile_ tek_parms") input;
    else open file(parm) title("vfile_ calcomp_parms") input;
get_parm:
    get file(parm) list(state_parm,pagex,pagey,xaxis,yaxis,xorig,xmax,yorig,ymax);
    if state_parm ^= fips_code then go to get_parm;
    if ct = "t" then do;
        call setup_tektronix_tcs;
        call ask_ ("Enter baud rate.^/",response);
```

```
        if response = "300" then call tk30;
        else if response = "1200" then call nhance(120);
        else if response = "9600" then call nhance(960);
        else call tk30;
    end;
    else if ct = "c" then do;
output_prompt:
        call ask_ ("Type ""t"" for tape output or ""s"" for segment output.^/",ans);
        if ans = "t" then output = "-tape";
        else if ans = "s" then output = "-file";
        else do;
            call ioa_ ("Invalid response ^a.",ans);
            call ask_$ask_clr;
            go to output_prompt;
        end;
        call ask_ ("Enter tape number or segment name.^/",ans);
        call setup_calcomp (output,"-name",ans);
        call calcmp(16);
    end;
    call ioa_ ("You can enter up to three heading lines for the plot.");
    call ioa_ ("The fourth or last line will be today's date,^/which is put in automatically by the program.");
head_num_prompt:
    call ask_$ask_int ("Enter the number of heading lines you would like (from 1 to 3).^/",head_num);
    if head_num < 0 | head_num > 3 then go to head_num_prompt;
    call ioa_ ("Each heading line must be less than 99 characters.");
    do i = 1 to head_num;
head_prompt:
        call ask_$ask_line ("Enter heading line number ^i.^/",heading(i),i);
        if heading(i) = " " then go to head_prompt;
    end;
    call flatbd;
    call nochek;
    call page (pagex,pagey);
    call projct ("lambe");
    if ct = "c" then call newpen(1);
    call title (" ",1," ",1," ",1,xaxis,yaxis);
    do i = 1 to head_num;
        call headin (rtrim(heading(i))||"$",100,2.,head_num + 1);
    end;
    call headin (date_in,8,1.,head_num+1);
    xorig=-95.0;
    xmax=-93.0;
    yorig=35.0;
    ymax=36.0;
    call mapgr (xorig,1.,xmax,yorig,1.,ymax);
    call dash;
    call grid(1,1);
    call reset ("dash");
    on endfile(oilgasin) go to oilgas1;
    on endfile(in) go to oilgas1;
    on endfile(input) go to p999;
p000:
    get file(in) edit(j1,j2,j3,npts,j4,j5,j6,j7)(col(1),8(f(5,0)));
    npts = npts - 1;
    get file(in) edit (xflunk,yflunk,(xlong(i),ylat(i) do i = 1 to npts))
    (col(1),6(f(12,9)));
    do i = 1 to npts;
        xlong(i) = xlong(i) * 57.29577951;
        ylat(i)  = ylat(i) * 57.29577951;
    end;
    call curve(xlong,ylat,npts,0);
    go to p000;
oilgas1:
    get file(input) edit(info) (col(1),a(9));
p001:
    get file(oilgasin) edit(field_check,field_nm,state_in)
    (col(2),a(9),x(1),a(40),x(1),a(2));
    if  field_check = " " then do;
        put skip file(mess) edit (field_check,field_nm,state_in,"field id is blank")
        (col(2),a,col(12),a,col(53),a,col(56),a);
        go to p001;
    end;
    if field_check = last_field_ck then go to p001;
    last_field_ck = field_check;
    do while (info < field_check);
        get file(input) edit(info) (col(1),a(9));
    end;
    if info ^= field_check then do;
       put skip file(mess) edit(field_check,field_nm,state_in,"no record found in oil/gas data")
       (col(2),a,col(12),a,col(53),a,col(56),a);
       go to p001;
    end;
    open file(oilgasyr) input;
year:
    get file(oilgasyr) edit(field_ck_yr,field_nm_yr,year_discovered,
    year_abandoned,depth,series_nm)(col(2),a(9),col(11),a(40),
    col(51),a(4),col(55),a(4),col(60),a(5),col(66),a(15));
    if field_ck_yr = " " then go to year;
    if field_check ^= field_ck_yr then go to year;
    close file(oilgasyr);
    shade_sw = "1"b;
```

```
head:
    get file(input) edit(info,state_code,oilgas,lat(1),long(1),field_npts)
    (a(6),x(1),a(9),x(1),a(1),x(1),f(6,3),x(1),f(7,3),x(1),a(53));
    if ct = "c" then call newpen(1);
    i=index(field_npts,",");
    field_name = substr(field_npts,1,i-1);
    if ^name_sw then
    call rlmess (rtrim(field_name)||"$",100,-long(1),lat(1));
    if name_sw then name_sw = "0"b;
    npts_out = substr(field_npts,i+1);
    ii = index(npts_out,",");
    npts = substr(npts_out,1,ii-1);
p01:
    i = 0;
    j = 0;
p02:
    i=j+1;
    j=j+5;
    if j > npts then j = npts;
    get file(input) edit(((lat(n),long(n) do n = i to j))
    (col(1),x(16),5(f(6,3),x(1),f(7,3),x(1)));
    do n = i to j;
        long(n) = -long(n);
    end;
    if j ^= npts then go to p02;
    if oilgas = "0" & ct = "c" then call newpen(2);
    else if oilgas = "G" & ct = "c" then call newpen(3);
    call curve(long,lat,npts,0);
    if shade_sw then do;

        if oilgas = "G" then call shade(long,lat,npts,45.,0.07,1,work,500);
        if oilgas = "0" then call shade(long,lat,npts,135.,0.07,1,work,500);
    end;
    get file(input) edit(info)(col(1),a(9));
    if info = field_check then do;
        name_sw = "1"b;
        go to head;
    end;
    shade_sw = "0"b;
    go to p001;
p999:
    call endpl(0);
    call donepl;
    if ct = "c" then call setup_calcomp("-reset");
end map;
```