GEOLOGICAL SURVEY OF CANADA
COMMISSION GEOLOGIQUE DU CANADA

Open File 665

GENERALIZED POSTING ROUTINE: A COMPUTERIZED SYSTEM FOR THE PLOTTING OF EARTH SCIENCE POINT DATA

D.N. PROUDFOOT
GEOLOGY DEPARTMENT
THE UNIVERSITY OF ALBERTA

T.P. LAM
ALBERTA DEPARTMENT OF ENVIRONMENT

Geological Survey of Canada (Calgary)
3303 - 33 Street N.W.
Calgary, Alberta T2L 2A7

1980

TABLE OF CONTENTS

CHAPTER		PAGE
	Author Contributions and Acknowledgements	V
I	INTRODUCTION	1
	Why was GPR created?	1
	A Brief Overview	1
	Features	2
	Sample Applications	2
	Some Limitations of GPR Capabilities	4
	Pre-run Requirements of GPR	5
	The Run Specification	6
	How Does GPR Operate?	7
II	THE LANGUAGE	8
	Introduction	8
	Commands	9
III	GENERAL COMMANDS	11
	What is a general command	11
IV	FILE SECTION	13
V	MAP SECTION	16
VI	LEGEND SECTION	28
VII	DISPLAY SECTION	30
VIII	LOCATION SYMBOLS	31
IX	SELECTION CAPABILITIES	33
	EXPRESSIONS	33
X	THE VARIABLE DIRECTORY	39
XI	A FAST METHOD TO SPECIFY VARNAMS AND THEIR FORMAT	46

XII ERRORS 48

How GRP Handles Errors 48

How to Correct Errors 48

LIST OF ILLUSTRATIONS

FIGURE 1: A sample posting map displaying unlabelled location symbols.

FIGURE 2: A sample posting map showing labelled location symbols and a legend.

FIGURE 3: A sample posting map which displays a GPR-selected subset of the data using one location symbol.

FIGURE 4: A sample posting map which displays a GPR-selected subset of the data, using two different location symbols.

FIGURE 5: A demonstration of the symbol overplotting feature.

FIGURE 6: A further demonstration of the label capability.

FIGURE 7: The location symbol directory.

FIGURE 8: A sample posting map which displays an example of oriented symbols.

LIST OF TABLES

TABLE I: A sample variable directory.

TABLE II: The GPR Language.

TABLE III: A list of orientation symbols and their uses.

LIST OF APPENDICES

APPENDIX I: A selection of examples, including the commands used to generate figures in the text.

APPENDIX II: System Documentation

Author Contributions and Acknowledgements

The original concept for GPR was developed by D.N. Proudfoot, D.W. Flint, G.D. Williams and J.A. Irvine. Design work was done by D.N. Proudfoot, with programming and systems analysis by Tom Lam; and much of the testing and debugging performed by Curtis Stevens. We are grateful to S. Cheshire for his work on the conversion of early versions of GPR to CDC equipment, as well as to M. Murphy and P.M. Waters for their help in testing GPR. We would also like to thank B.A. Latour, J.A. Irvine and K. Baer of the Geological Survey for their patient support. Most of the research and development for GPR was performed as part of a coal resource evaluation research contract, funded by the Geological Survey of Canada, with support from the University of Alberta for the final version of GPR.

I. INTRODUCTION

A. Why was GPR created?

In our work with the Western Canada Coal Resource Data Base, performed under contract to the Geological Survey of Canada, we found an increasing need for a generalized plotting system for the generation of base maps. It was decided that this system should have the capability to select data using criteria specified by the user and the versatility to operate on most types of geological data. In addition, the map produced had to be a "stand alone" product which was easily understood without additional notes. The system had to be easy to use, requiring very minimal knowledge of computer and plotter facilities. The Generalized Posting Routine (GPR) is an outgrowth of these requirements.

B. A Brief Overview

GPR is a computer software system for the display of spatially distributed point data. The product is a digital plotter map, with each data point location denoted by a symbol and often accompanied by a label. The title, subtitle, legend and annotated, scaled axes are also plotted. In addition, a line printer listing with all (or a subset of) the data for each point plotted, can be produced.

For example, Figure 1 is a simple map which displays the location of all coal exploration boreholes within an area.

C. Features

Features built into GPR include the following:

1. Interactive map specification capability.
2. Free form input of commands and parameters.
3. Syntax check of user input.
4. Facility to apply selection criteria to the data to be plotted.
5. Facility to plot orientation data.
6. Facility to overlay symbols, thus enabling the construction of new symbols.
7. Facility to plot a legend.
8. Facility to edit map specifications.
9. Facility to produce multiple maps from the same or different data sets without leaving control of GPR.[1]

D. Sample Applications

For the purposes of a discussion of GPR's capabilities, it is useful to define several terms. In this context, a data set[2] is a collection of data pertaining to a number of geographic locations. The data

---

1. This feature is dependent on the version of GPR implemented at your installation and is system dependent.
2. Heavy print is used to emphasize computer terms which may be unfamiliar to the reader.

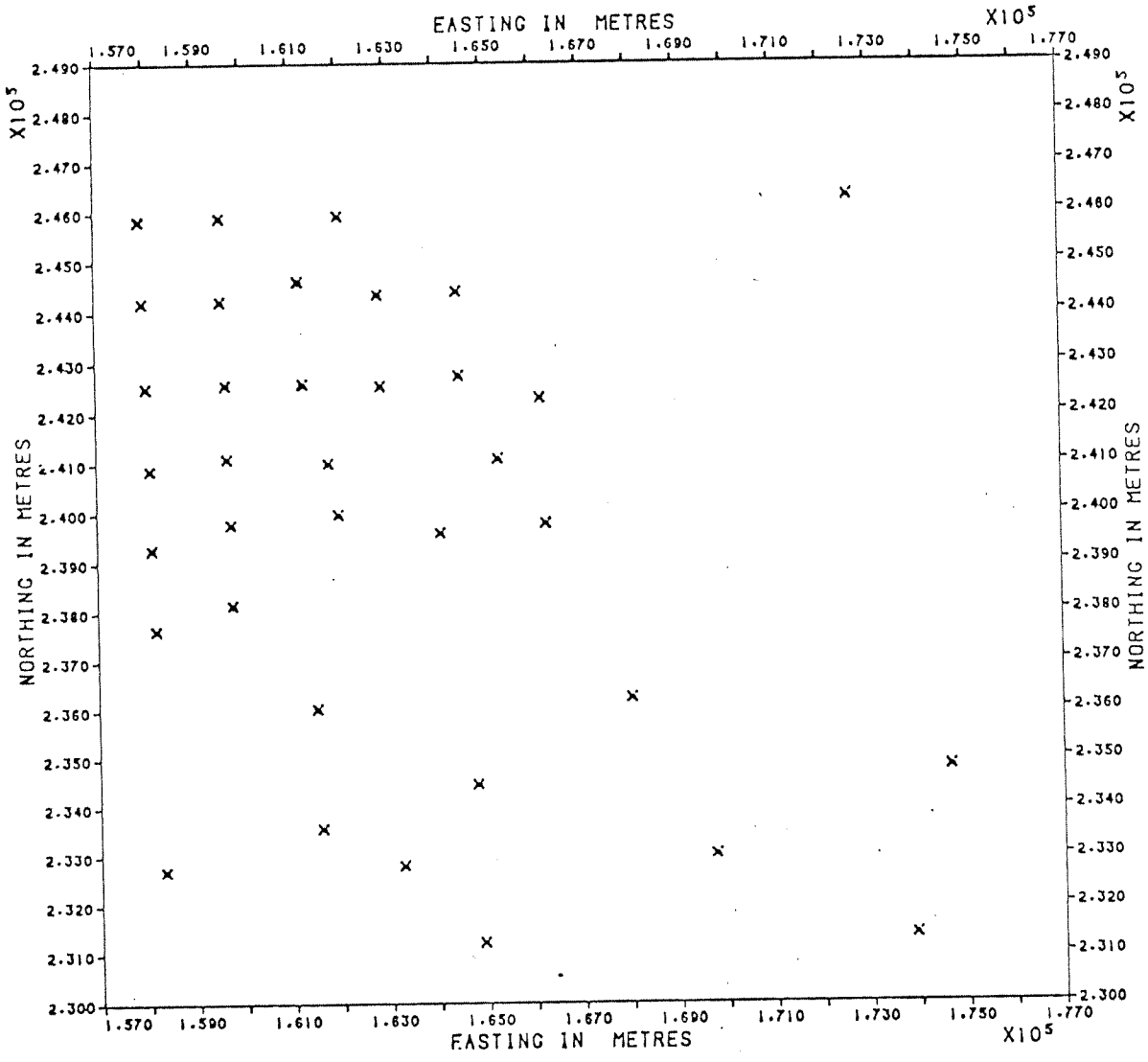


FIGURE 1

THE LOCATION OF COAL EXPLORATION BOREHOLES IN THE AREA

at each location might, for example, describe a geological borehole log. The data belonging to one borehole are contained in a record. Thus, a data set consists of a number of records. A number of variables comprise each record. These variables include the borehole location, expressed as a pair of rectangular coordinates (x and y),[3] and at least one measured or assigned value (z-value).

Given that the user has a data set stored in the computer, what can GPR enable him to do with his data? (Refer to Table I for a list of variables in the sample data file.)

1. First of all, GPR must know where to find the data set, and what is in it. Reference to the location of the data set name must be made external to GPR at run time. Refer to "The Run Specification." The user must specify the variable names and their format within control of GPR. This enables GPR to access the data set directly when it is time to generate a plot.
2. The first map specified by the user may simply show the spatial distribution of all of the data locations in the data set. Figure 1 is a posting map, with a user specified symbol, to mark the

---

3. Preferably in Universal Transverse Mercator Projection, but any other rectangular coordinate system may be used.

location of each data point.

3. The user may also plot a label beside each location symbol. Figure 2 displays seam name and seam thickness in the label beside each borehole location. Note the legend which describes the label in the user's own words.
4. A subset of the borehole records in the data set may be selected and plotted. Figure 3 displays the location of boreholes which have intersected a coal seam greater than or equal to 5 feet in thickness.
5. Several different subsets may be selected and plotted on the same map, each with its own location symbol and label. Figure 4 displays borehole locations which intersect coal seams less than 5 feet in thickness, and also borehole locations which intersect coal seams greater than or equal to 5 feet in thickness. Note the utility of the legend.
6. Orientation data may be plotted at the location of measurement. Figure 8 displays bedding plane strike and dip.

E. Some Limitations of GPR Capabilities

1. It is preferable to use Universal Transverse Mercator Projection coordinates, since map axes are annotated in UTM units. However any rectangular coordinates are acceptable.
2. X and Y coordinates must be in either feet or

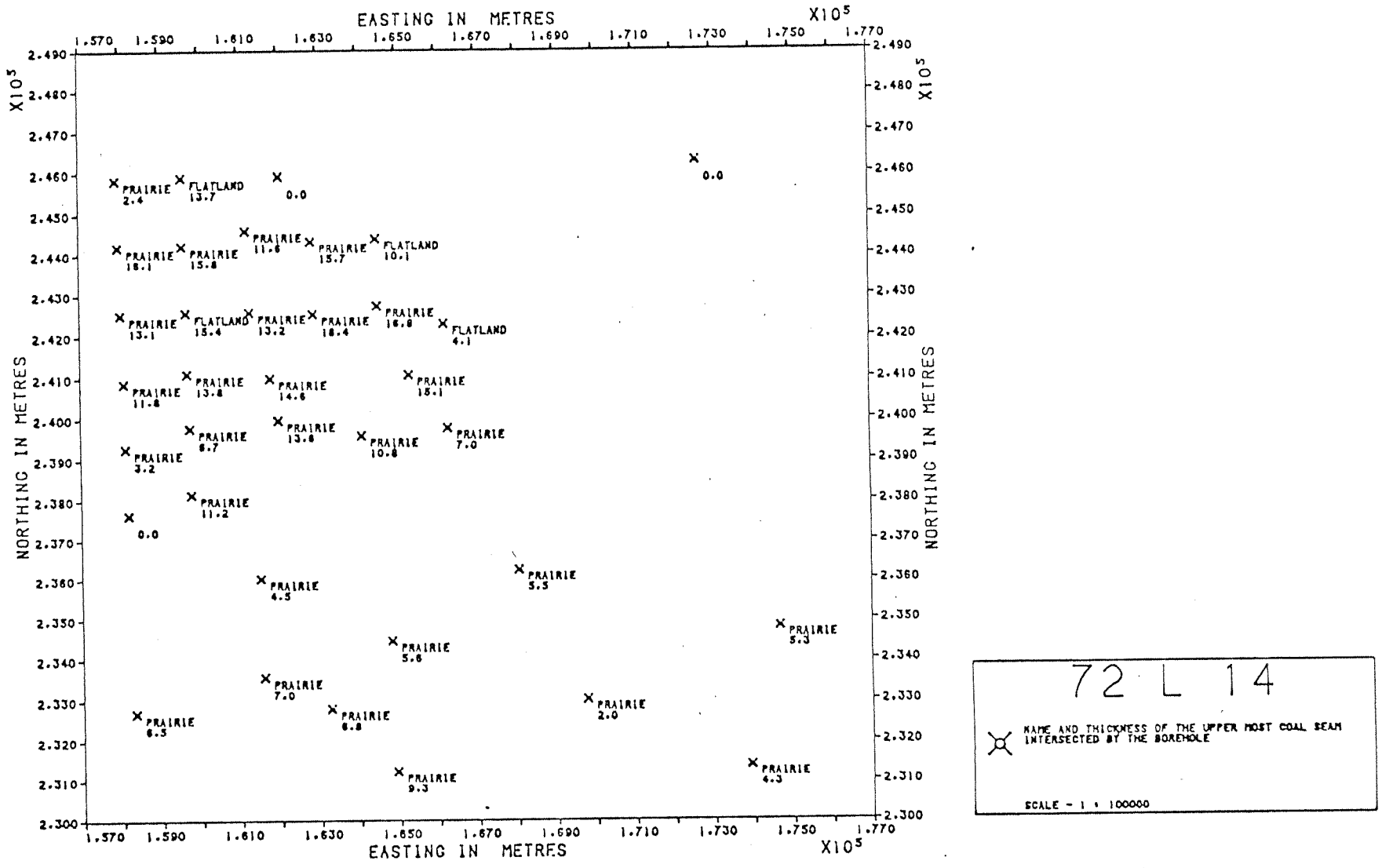
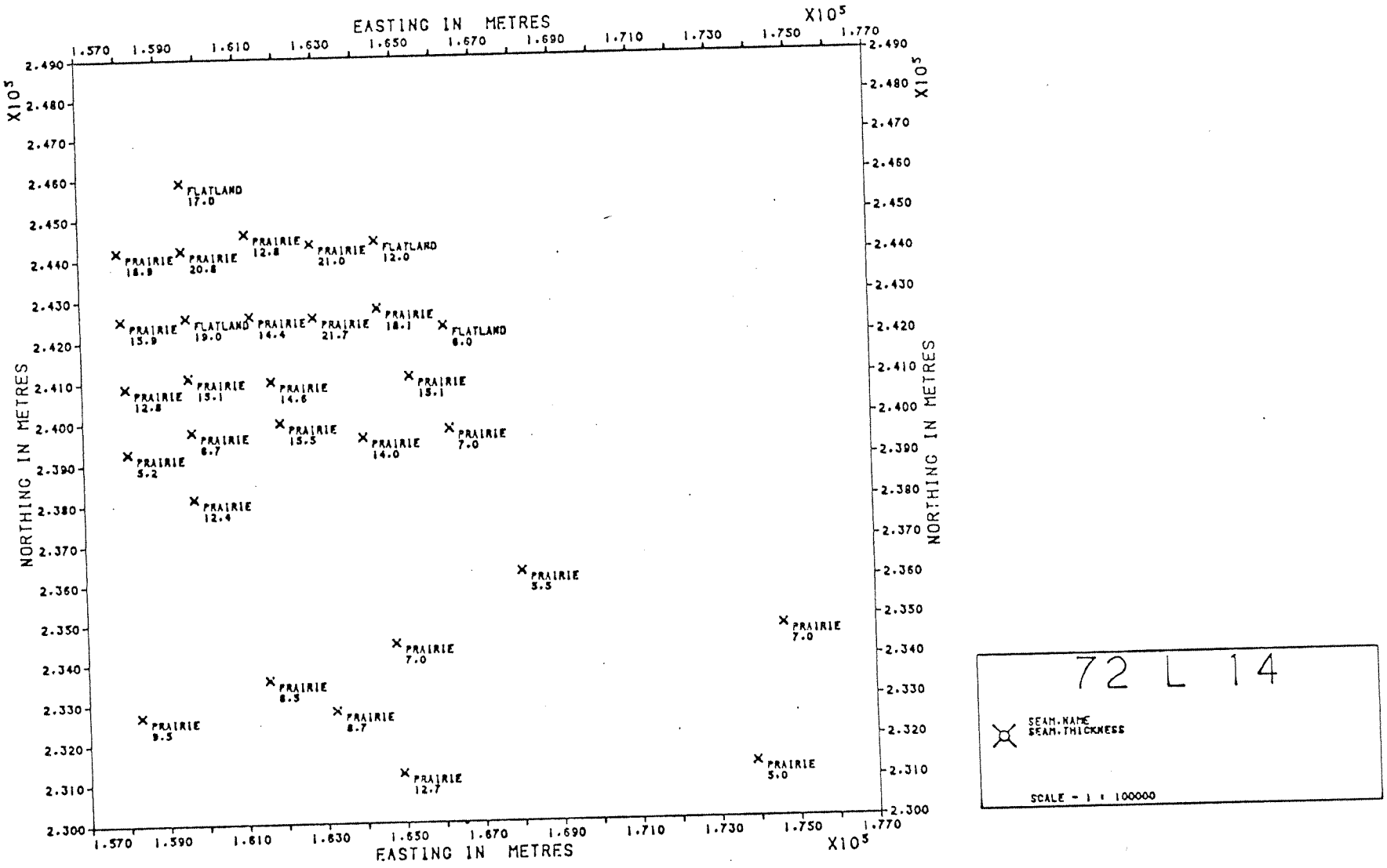


FIGURE 2

COAL SEAMS INTERSECTED BY BOREHOLES WITHIN THE AREA



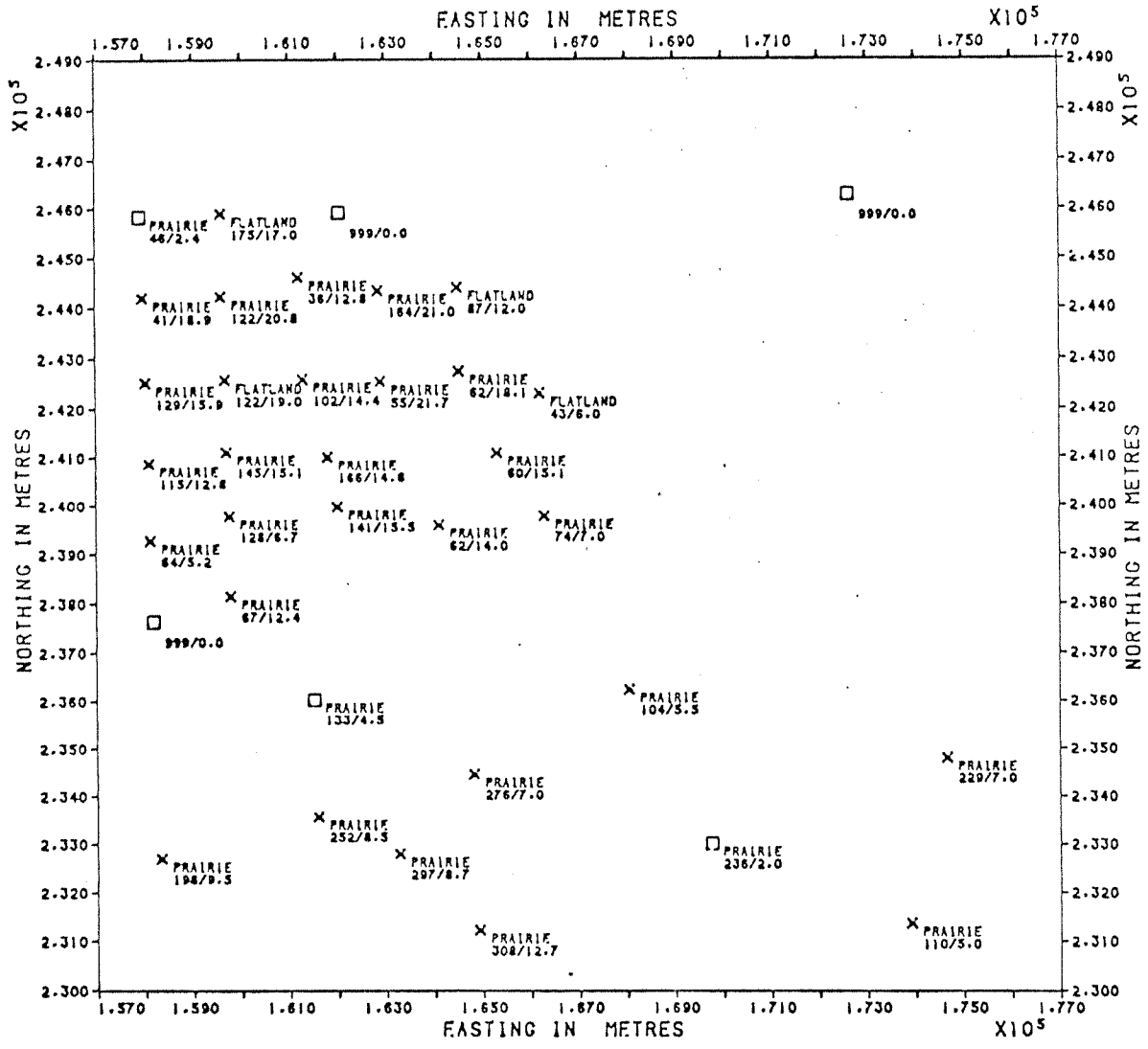
72 L 14

SEAM.NAME
SEAM.THICKNESS

SCALE - 1 : 100000

FIGURE 3

BOREHOLES WHICH INTERSECTED COAL SEAMS >= 5 FT IN THICKNESS



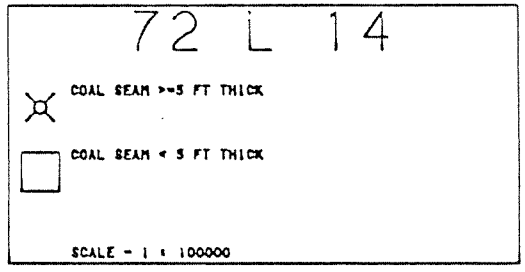


FIGURE 4

COAL SEAMS >=5 OR <5 FT THICK

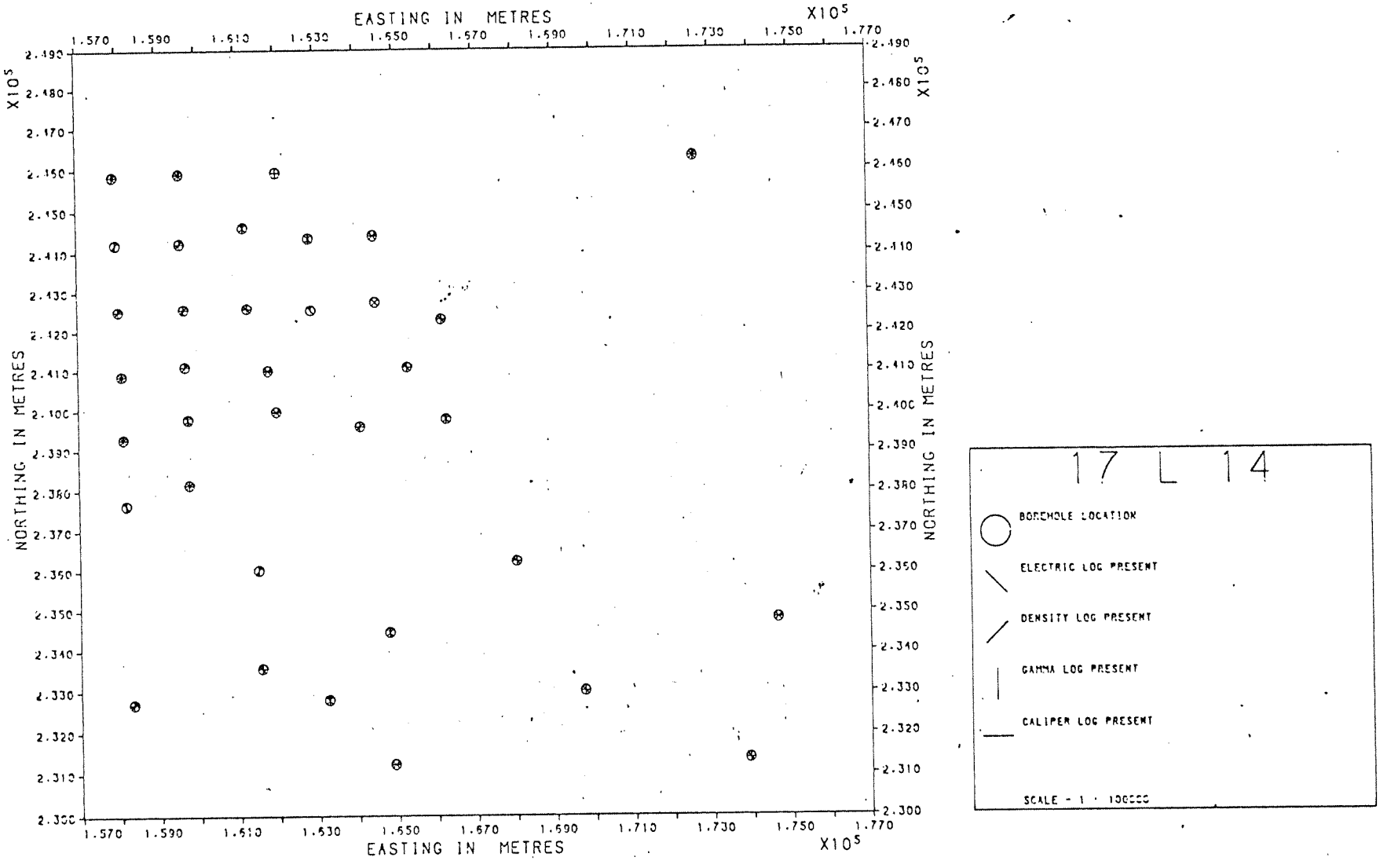
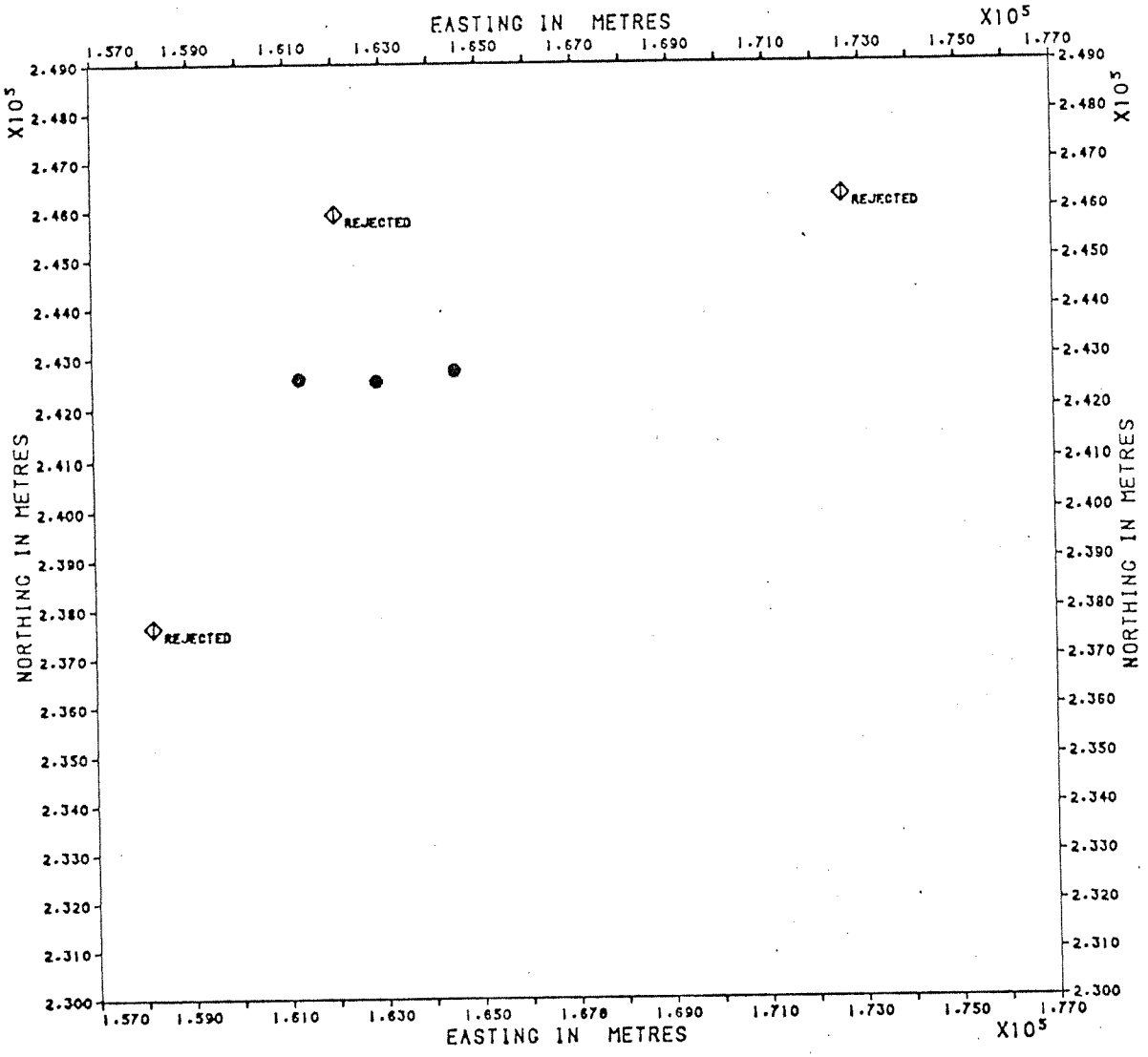


FIGURE 5

GPR LOCATION SYMBOL OVERLAY FACILITY (BUILDSYM)



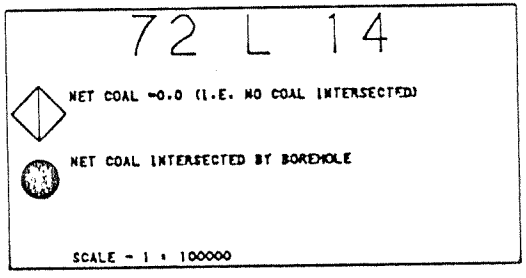
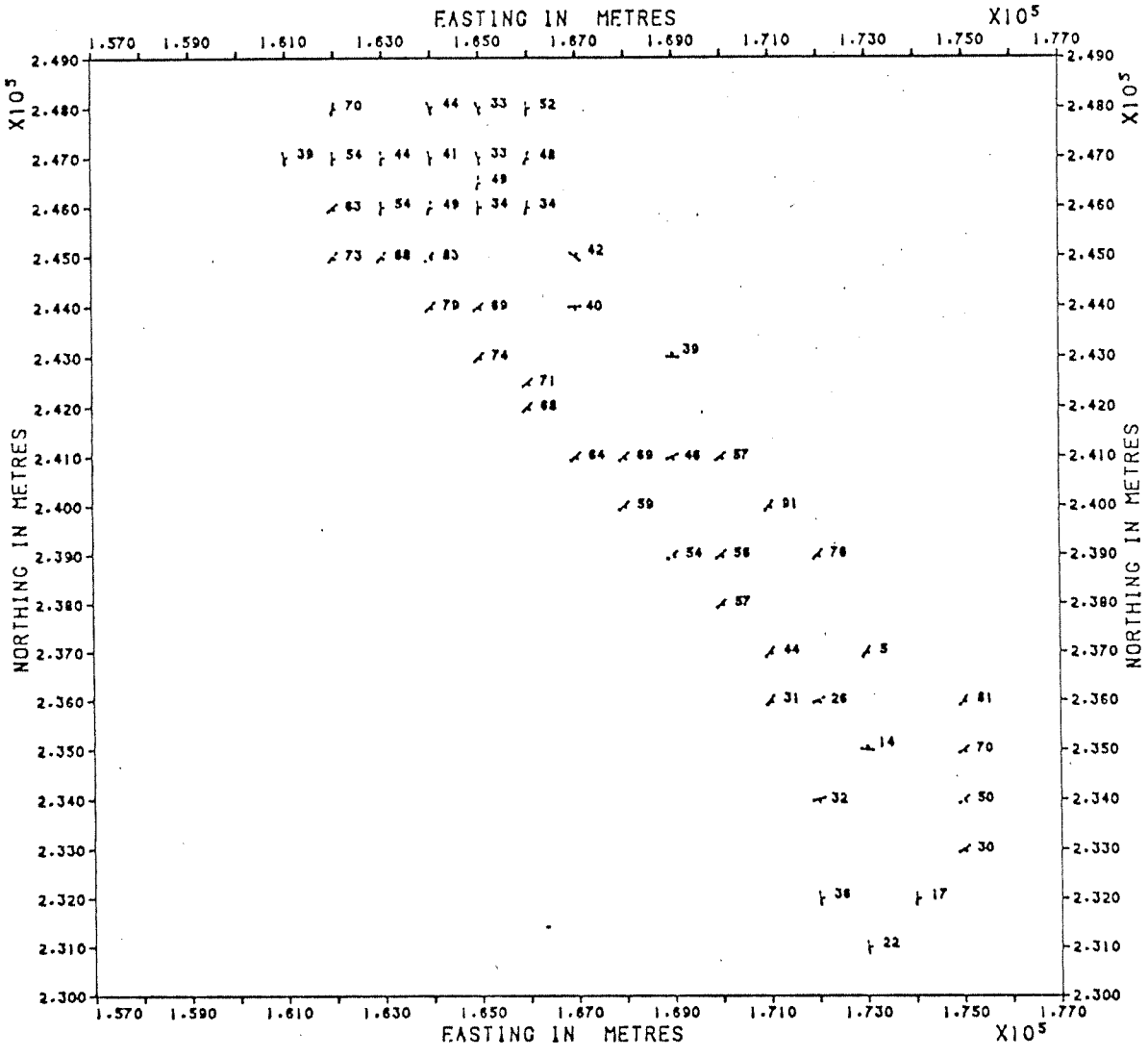


FIGURE 6

A SELECTED AREA



72 L 14

⊥ BEDDING PLANE ORIENTATION

SCALE - 1 : 100000

FIGURE 8

BEDDING PLANE ORIENTATIONS MEASURED IN THE AREA

metres.

3. Single maps which overlap UTM zones, will be distorted unless location coordinates are modified external to GPR. It is suggested that maps be confined to single UTM zones, and that where necessary, a mosaic of GPR maps be constructed.

4. GPR does not warn the user when the physical size of a map exceeds plotter paper size. It indicates to the user what the total map size is, but truncates the upper and right sides of the map.

F. Pre-run Requirements of GPR

Before GPR is run,* the user must have an introductory level of computing knowledge and several types of information must already be defined and stored in the computer.

1. A general familiarity with FORTRAN logic and the FORTRAN format statement.

2. The ability to gain access to your local computer and to be able to build and access files within that computer system.

---

*. The term 'run' is used to describe the initiation and operation of GPR on the computer.

3. Some familiarity with local digital plotting facilities and how to use them.

4. The variable directory, which defines the permissable data item names of your data. Refer to Chapter X for details.

5. The data set which is to be plotted must be stored in the computer. The data must be in fixed format, with all data belonging to one location stored in the same record. You must know the FORTRAN format statement for this data. Note that the format for each variable must agree with that specified in the variable directory.

G. The Run Specification

In order to run GPR, a command must be specified to the computer. This command, which is different for each different computer operating system, specifies the device numbers to be used by GPR for various files. They are as follows:

Unit 9 = A temporary file which is to contain the control data used by the program in unit 10.

Unit 10 = A temporary file which is to contain the generated FORTRAN program.

Unit 15 = a list of variable names and their respective format in the data set to be accessed. Note: Unit 15 is

only required if 'SELF' is used for VARNAM or FORMAT.

Unit 16 = The variable directory.

A maximum computing time limit should also be used. Under the Michigan Terminal System this command is as follows:

$RUN GPROBJECT 9=-data 10=-source 15=variable_list_and_format (FORVAR) 16=-variable directory t=10s

where GPROBJECT is the GPR system and T=10s is a 10 second time limit.

H. How Does GPR Operate?

GPR is a code generating, four step system. That is, the result of a GPR run is a generated FORTRAN source program, which must be compiled in a second step and then run with the data set in a third step to create a plot description.

Step 4 involves the the actual plotting of the plot description file.

II. THE LANGUAGE

A. Introduction

In order to plot a map, GPR must have all pertinent information, which may be categorized as:

1. (i) The data set name.
   (ii) Variable names and their corresponding format.
2. The specifics about the map.

These categories are defined to GPR via four modules and the run specification.

(i) FILE section
(ii) MAP section
(iii) LEGEND section
(iv) DISPLAY section

Each section has a number of commands and parameters, each with its respective sets of values. Refer to Table II.

All communication with GPR is done via free form input, with the exception of access to the plot data set, and the variable directory. Free form allows the input to be unformatted, that is:

(i) input may start in any column.
(ii) one or more blanks may be entered between complete words. (A word is defined as any set of consecutive characters from the set A-Z, 0-

9 and the dot.) All GPR language components are considered as words. (including: commands, section names, command values, operators, operands).

(iii) the maximum input line length is 80 columns, including blanks.

(iv) to continue a line, break a word in column 80, or place the entire word on the next line.

1. Commands

A command is used to specify an action or to describe some aspect of the plotting task.

There are a number of general commands which can be stipulated within the FILE and MAP sections, and other commands which can be used only within either the MAP or the FILE section. Most commands may be specified in any order within a section, with several exceptions, which are marked by 'F' in Table II and explained later in the documentation. Compulsory commands are marked with 'C', and commands for which there is a default if they are omitted are marked with 'D'. Most commands have a range of possible values, one of which must be specified when that command is used. A ccmmand should be specified:

COMMAND = command value

e.g. XYUN = metres.

One command, "LOCSYM", has a number of parameters associated with it. These will be discussed in Chapters V and IX.

The LEGEND and DISPLAY features are completely controlled by GPR so that once the feature is invoked, the user is prompted for the required input.

III. GENERAL COMMANDS

A. What is a general command?

A general command usually initiates an action by GPR and can be used almost anywhere within the file and map sections. Note that all commands are listed alphabetically for the purposes of reference and discussion. They may be entered in any order.

1. "CLEAR" - is used to erase the entire contents of a section. It is frequently used to erase major errors, or to respecify the FILE or MAP sections without redefining an entire map during a second or subsequent map specification. "CLEAR" must be specified within a section before "RETURN" is specified. Refer also to the "DELETE" command (Chapter V).
2. "END" - is used to terminate the "VARNAMS" list in the FILE section, the "LIST" list and the "LOCSYM" logical capability feature in the MAP section. If "END" is omitted, GPR will continue to accept input. That is, GPR assumes that the user is continuing to input into the current command/parameter. This omission leads to errors. "END" must be preceeded by at least one blank and no preceding commas are permitted.

3. "FULL" - is specified to enter full prompt mode.[5] In this mode the user is prompted for all compulsory commands, but must provide his own command values and parameters where required. The DISPLAY and LEGEND sections are in full prompt mode. If "FULL" is not specified, then the user must input all commands. GPR will prompt only if compulsory commands are omitted.

4. "RETURN" - is specified to move out of the control of a section. "RETURN" causes GPR to check for the presence of all compulsory commands in the current section. If any are missing, the user is prompted for them in full prompt mode. Otherwise, a summary of the current section is listed, enabling the user to check his input. If an error is encountered, the user should refer to Chapter XII.

5. "STOP" - is used to terminate a GPR session. The user is prompted for this command at various points during a session and can specify "STOP" only at these points.

---

5. The term 'mode' is used here to describe the manner in which GPR operates, ie. either in the full prompt or partial prompt manner.

IV. FILE SECTION

A. "FILE" - is used to specify entry into the FILE section. This section defines to GPR the data to be plotted and its location in storage. All commands in this section are compulsory, with partial prompt as the default mode.

1. "FORMAT" - refers to the format of the data to be plotted. Any one of the following command values may be specified.

1.1. "FIXED" - the data set is stored in fixed format. When FIXED is specified, GPR will prompt with WHICH IS. The user should then enter the FORTRAN format for one data record, starting with the left hand bracket.

e.g. WHICH IS (I7,2X,I1,2X,F5.2)

Note that all alphabetic and alphanumeric data should be referenced in the format statement by A2 and a repetition factor.[6]

e.g. an alphanumeric variable with a value which can be up to 15 characters in length should be formatted by 8A2.

GPR will check each variable name, type and length against the variable directory and will use the directory type and length should there be any discrepencies. No error message

---

6. This is system dependent.

regarding these discrepencies is issued by GPR. Format syntax is checked at run time.

1.2. "SELF" - the format statement for the data to be accessed is stored in a data set called FORVAR (this file name is installation dependent). Refer to Chapter XI.

2. "VARNAMS" - refers to the list of variable names to be read from each record in the data set specified by the "NAME" command.

2.1. The variable names are listed in the order in which they occur in the data record, separated by commas. Each variable name should be entirely on one 80 column line, or, if continued to a second line, should extend to column 80 before the break. VARLIST must be terminated by "END".

Permissible characters are limited to: A-Z, 0-9 and (.) dot. Except the first character which must be from the set A-Z and the reserved words referred to in Chapter X. If a data set has many variables which are not going to be used during a particular GPR session, it is most efficient to specify only those variable names which are to be used immediately.

2.2. "SELF" - may be specified instead of the variable names. This indicates to GPR that the variable name list is in the data set stored on unit 15. Refer to Chapter XI for details.

3. "XYUN" - refers to the type of units in which the X and Y location coordinates operate. Two unit types are permissible, "METRES" and "FEET".

V. MAP SECTION

"MAP" is used to specify entry into the MAP section. This section enables the user to specify necessary map features, location symbols, labels and selection criteria for the data to be plotted. There are both compulsory and optional commands in this section. Partial prompt is the default mode.

A. "BUILDSYM" - is used to specify the type of selection technique to be utilized when the logical capability discussed for "LOCSYM" is employed.

1. "OFF" - is the default. It results in at most one symbol being plotted for each record in the data file (discussed in more detail below).
2. "ON" - results in at least one symbol being plotted for each selected record. It is possible to have overplotting of several symbols at the same location (discussed in detail below).

There are two types of selection techniques available in GPB. Both methods involve a sequential comparison of each data record in the input data set with the criteria associated with each location symbol. If "BUILDSYM = OFF", the comparison progresses from the first location symbol ("LOCSYM") defined, to the last, for each record, until a success occurs. At this point the comparison moves

to the next record beginning over again at the first location symbol. Thus only one location symbol can be plotted for each successful record.

If "BUILDSYM = ON", the comparison progresses from the first location symbol defined to the last, for each record, regardless of whether a success occurs or not. Thus if multiple successes occur for one record, more than one symbol will be plotted at that location. This feature enables the construction of complex symbols from simple symbols. The following example should clarify this explanation.

> Suppose that we are interested in working with borehole information. In particular we want to determine the geographic distribution of coal exploration boreholes to obtain some qualitative concept of sampling density. In addition, we want to determine a qualitative estimate of our degree of confidence in the data measured at each location, based on the presence of a combination of borehole geophysical logs. A map displaying the following symbols should do the job.
>
> 0 - borehole location
>
> \- electric log present
>
> / - density log present
>
> | - gamma log present

_ - caliper log present

How do we specify this task to GPR?

Assume that the FILE section has already been completed and that all compulsory commands in the MAP section have been defined, with the exception of the first location symbol. ("LOCSYM") Then:

```
BUILDSYM = ON
LOCSYM = 29
LOCSYM = 34
  WHEN #1 ELECTRIC_LOG = 'Y'
  LOGIC #1 END
LOCSYM = 33
  WHEN #1 DENSITY_LOG = 'Y'
  LOGIC #1 END
LOCSYM = 35
  WHEN #1 GAMMA_LOG = 'Y'
  LOGIC #1 END
LOCSYM = 36
  WHEN #1 CALIPER_LOG = 'Y'
  LOGIC #1 END
RETURN
```

Figure 5 is the resulting map.

Note how readily the various combinations of symbols provide a quick survey of sampling density and the number of logs present. "BUILDSYM" can be specified anywhere in the MAP section, and will remain "ON"

for subsequent runs within a single GPR session. It is important to note that the "BUILDSYM" feature is considerably more expensive to run when "ON" rather than "OFF".

B. "DELETE" - is used to delete a location symbol including all logic and labels ("LOCSYM" and "LABEL") associated with it. This is most useful when errors have been made during the definition of symbol selection criteria. The form of the command is "DELETE LOCSYM i" where "i" is the sequence number of the location symbol (e.g. each "LOCSYM" is referred to by a number which is the order in which it was entered). "i" has no relationship to the symbol number "nn" discussed under "LOCSYM". This sequence number can be obtained by counting the number of LOCSYMS entered down to the one of interest, including deleted symbols.

C. "DELTATIC" - refers to the spacing between tic marks on the map boundaries. It is specified in the units of the X and Y coordinates. The default value is 1000 units.

D. "LIST" - enables the user to specify all or a subset of the variables from the selected data set for output to a temporary data set. This list must be placed in a more permanent form before a computer session is terminated, otherwise it will be destroyed. Note that the output format is the same as "FORMAT", unless "OUTFORM" is specified. If "LIST=" is not specified, then by default all variable names for selected points are plotted.

1. "LIST = variable names" ("VARNAMS") causes only the named variables for selected records to be output. Each variable must be delimited from the following variable name by a comma, and the entire variable name list should be terminated by "END". This syntax is identical to that for "VARNAMS" in the FILE section.

   e.g. LIST = RECORD_NUM, HOLE_NAME, ELECTRIC_LOG, DENSITY_LOG END

   An "CUTFORM" statement must accompany this use of the "LIST" command.

E. "LOCSYM" - is used to specify the location symbol to be plotted for each selected record. At least one location symbol must be specified. Refer to the location symbol directory (Figure 7). "LOCSYM" can have the following values.

1. "LOCSYM = nn", where nn refers to a symbol in the symbol directory. Refer to Chapter VIII for a detailed discussion of location symbols.
2. "LOCSYM = PREVIOUS", is used to duplicate an immediately preceding location symbol and associated selection criteria. This feature is useful for defining second and subsequent location symbols, when more than one label is used for the same location symbol (refer to Chapter IX for details).

For data which require more than one symbol to adequately display them, the "LOCSYM" command is

equipped with selection capabilities. This facility is available through the use of a number of parameters, which, because they must all be used together, are referred to as a selection parameter set. (Refer to Table II). These parameters and their corresponding values have been assigned levels for ease of use (e.g. level in the sense of a hierarchical level).

Level 1 parameters are optional.

Level 2 parameters are optional, but can only be used after level 1 parameters have been specified for the current symbol.

Level 1 parameters are subdivided into:

1.1 the selection parameter set:

"WHEN	conditions"
"LOGIC	logical
	expressions"
"END"	

For example.

```
LOCSYM = 10
      WHEN #1 SEAM.THICKNESS >= 5
      LOGIC #1
      END
```

Which will select all boreholes with coal seams greater than or equal to 5 feet in thickness and will plot location symbol 10 at the location of each. Refer to numbers 3 and 4 in Appendix I for complete examples.

The use of "WHEN" immediately following a "LOCSYM" specification indicates to GPR that the selection facility is to be employed. The rest of the selection parameter set must follow. The conditions which immediately follow "WHEN" must be in the form:

#n expression

where #n is the sequence number of the condition, which begin at #1 and increase sequentially to a maximum of #10. An expression has the form

operand 1 <operator> operand 2.

The "LOGIC" statement is compulsory, and is used to logically relate conditions. It must be used when only one condition is specified, as in the above example. The "LOGIC" statement must be terminated by "END", in the same manner as the variable list ("VARNAMS").

There are three kinds of exressions available in GPR. (Refer to example 6 in Appendix I.)

Arithmetic expressions in which the operands are numeric operands related by an arithmetic operator such as (TOP.DEPTH-BOTTOM.DEPTH).

Relational expressions in which the operands are compared by a relational operator such as (SEAM.THICKNESS > 5.0)

Logical expressions in which the operands are relational expressions related by a logical operator such as ((SEAM.THICKNESS > 5.0) and

(BOTTOM.DEPTH < 100.))

This limited discussion of the selection capabilities of GPR is intended as a brief outline. It does not describe how to use this facility. A more detailed discussion of the selection capability can be found in Chapter IX.

1.2 "LABEL" - is used to define the label (tag) which is to be plotted beside the location symbol specified in the associated "LOCSYM" command. Refer to number 2 in Appendix I. To specify a label, list the parameters in the order in which they are to be plotted. GPR variable names ("VARNAMS") and literals must be followed by a single delimiter which can be any single character including a blank. This delimiter, which is plotted as part of the label, is itself delimited by single quotation marks. Thus '/' might be used as a delimiter. The resulting label would cause / to be plotted between parameter values. Similarly ' ', would cause a blank to be plotted. There are three parameter value types associated with "LABEL", one or more of which can be used in a given label (refer to Table I).

a. variable names - as defined by the "VARNAMS" command in the FILE section. e.g. for a coal seam with a net coal

thickness of 7.6 feet and a top depth of 29 feet;

LABEL = NET.COAL '/' TOP.DEPTH ' ' would plot as 7.6/29 Refer to number 2 in Appendix I for a complete example.

b. literal character strings - delimited by single quotation marks and followed by a single character delimiter which is itself enclosed in single quotation marks, in the same manner as for variable names. Each literal can be up to a maximum of 30 characters long.

e.g. LABEL = 'REJECTED' '/' NET.COAL ' ' would plot REJECTED/1.5 beside the location symbol of a record selected by the associated "LOCSYM" selection parameter set. Refer to number 6 in Appendix I.

c. NEXTLINE - causes plotting of the label to skip down one line and begin the next character immediately under the first character of the line above.

e.g. LABEL=NET.COAL '/' TOP.DEPTH ' '
NEXTLINE SEAM.THICKNESS ' '
which will plot
7.6/133
8.1

beside the location symbol Refer to number 4 in Appendix I for a complete example.

For added flexibility, the "LABEL" parameter is also equipped with selection capabilities (refer to Table II). This facility is provided via the selection parameter set, exactly as for "LOCSYM". However because "LABEL" cannot be specified without a "LOCSYM" specification, and label selection criteria cannot be defined without a "LABEL", this selection parameter set is considered to be one level lower down in the hierarchy, at level 2. This feature enables different labels to be plotted for the same location symbol.

Note that if "BUILDSYM = ON", care must be taken that label overprinting does not occur. Since more than one label can be plotted at the same location.

F. "NDECP" - specifies the number of digits to be plotted to the right of the decimal point at annotated map boundary tic marks. "NDECP" must be an integer. The default for "NDECP" is 3.

G. "NO LIST" - specifies that no output will be written.

H. "OUTFORM" - specifies the output format for variables specified in the "LIST" command. The format is a FORTRAN IV format statement beginning with the left hand bracket, exactly as for "FORMAT" in the FILE section.

e.g. OUTFORM = (I6,10X,A15,5X,A1)

If "OUTFORM" is not specified, and "LIST=" is, then the

default is the same format as defined in "FORMAT" in the FILE section.

I. "SCALE" - specifies the scale of the map to be plotted. "SCALE" is expressed as a real number greater than zero. e.g. SCALE=250000, indicates that the scale of the map is 1:250000. The default scale is 50000.
Note: When changing the scale, check "DELTATIC" to ensure that a reasonable map boundary tic mark interval is plotted.

J. "S.TITLE" (subtitle) - is any character string (including blanks) enclosed in single quotation marks, up to a maximum of 60 characters (excluding the quotation marks which are not plotted). The subtitle is plotted below the map title at the base of the map. It is a compulsory command.
e.g. S.TITLE = 'This is the subtitle'

K. "S.T.H" (subtitle height) - specifies the height of the subtitle in inches. It must be a real number less than or equal to 0.2 inches, which is also the default. If this limit is exceeded, the default is used.

L. "SYM.H" (symbol height) - specifies the location symbol height in inches. It must be a real number less than or equal to 0.1 inches. If this limit is exceeded, the default is used.

M. "TITLE" - is any character string (including blanks) enclosed in single quotation marks, up to a maximum of 50 characters (excluding the quotation marks which are

not plotted). The title is plotted along the base of the map. Care should be taken to ensure that the title does not extend very far beyond the right map margin.

e.g. TITLE = 'This is the title'

N. "T.H" (title height) - specifies the height of the title in inches. It must be a real number less than or equal to 0.6 inches which is also the default. If this limit is exceeded, the default is used.

O. "XMAX" - is the maximum map coordinate in the X-direction, usually the eastern boundary of the map area, expressed in the same units as "XYUN".

P. "XMIN" - is the minimum map coordinate in the X-direction, usually the western boundary of the map area, expressed in the same units as "XYUN".

Q. "YMAX" - is the maximum map coordinate in the Y-direction, usually the northern boundary of the map area, expressed in the same units as "XYUN".

R. "YMIN" - is the minimum map coordinate in the Y-direction, usually the southern boundary of the map area, expressed in the same units as "XYUN".

VI. LEGEND SECTION

"LEGEND" is used to specify entrance into the LEGEND feature. This feature enables the user to explain the location symbols and labels plotted on the map. The four inch wide legend is plotted one inch to the right of the right hand map boundary label. Location symbols plotted in the legend are 0.5 inches high. This feature prompts for all input, and has no correction capacity, other than the respecification of the entire legend. The literal string input via "MAPID" is plotted across the top of the legend, and scale is plotted at the bottom. Refer to numbers 2,3,4,5 & 8 in Appendix I.

Immediately after entry into the LEGEND section, you are prompted by the following:

A. ENTER EXPLANATION FOR LOCATION SYMBOL # nn.

Where "nn" refers to a symbol number in the location symbol directory. The explanation for a "LOCSYM" may be up to 150 characters long, 50 characters per line. The dashed line printed just after the explanation prompt is 50 characters in length and is intended as an aid to legend input, so that the line maximum is not exceeded. For each blank line in the explanation, hit the return key. When the three lines have been entered, a prompt for another location symbol or the footing is printed.

B. FOOTING - is used to provide extra remarks regarding the map. It has a maximum of 150 characters, e.g. 3 lines of

50 characters each, and is entered exactly as for the location symbol explanation.

C. "MAPID" - is used for map identification data. It is any alphanumeric character string up to a maximum of 8 characters, delimited by single quotes. "MAPID" is plotted at the top of the map legend, when the LEGEND feature is used.

VII. DISPLAY SECTION

"DISPLAY" is used to specify entrance into the DISPLAY feature. This feature enables the user to display his map on a digital plotter. Refer to Appendix I for examples. Note that although the DISPLAY section can be entered at any time, the run will terminate abnormally if the "MAP" and "FILE" sections are not complete. Immediately after entry into the DISPLAY section, GPR generates a FORTRAN program which will be used to produce the map. When this program has been generated,a message is printed.

After GPR has been terminated via the "STOP" command, it is necessary to compile the generated program which is stored in device unit 10. This compilation task will have syntax specific to each different computer and its operating system. Under the Michigan Terminal System, this command is as follows:

```
$$RUN -LOAD#+SYMOBJ+*PLOTLIB 5=-DATA 9=-PLOT 11=INPUT.DATA
12=-LIST T=10S
```

Where SYMOBJ is the plot symbol library

*PLOTLIB is the MTS plotting library.

-DATA is the control data for the generated program

-PLOT is the plot description file.

-LIST contains a list of selected data records, if the LIST command is not turned off.

VIII. LOCATION SYMBOLS

There are 40 basic symbols available within GPR; (refer to figure 7).[7] The location of a point on a GPR map is marked by the centre of the location symbol plotted. Nine of the basic symbols (numbers 12-20 inclusive) are reserved for orientation data and numbers 39 and 40 have special functions, which are discussed below. Refer to Table II for the data input required for each symbol.

The variables used for orientation data may have any name, however the position, in the variable name directory, of each type of orientation variable is reserved. Refer to Chapter X for details.

Special function symbols "39" and "40" are used to draw straight line data on a map. Symbol "40" is used to specify that the plotter pen should not be raised when it is moved from the current location to the next location in data file being plotted. This causes a line to be drawn from the last location to the current one. Symbol "39" is used to specify that the plotter pen be raised when it moves to a new location, and that no symbol be plotted there. Care must be taken to organize plot data such that locations are in the order in which lines are to be drawn to join various locations. Experimentation is recommended before this

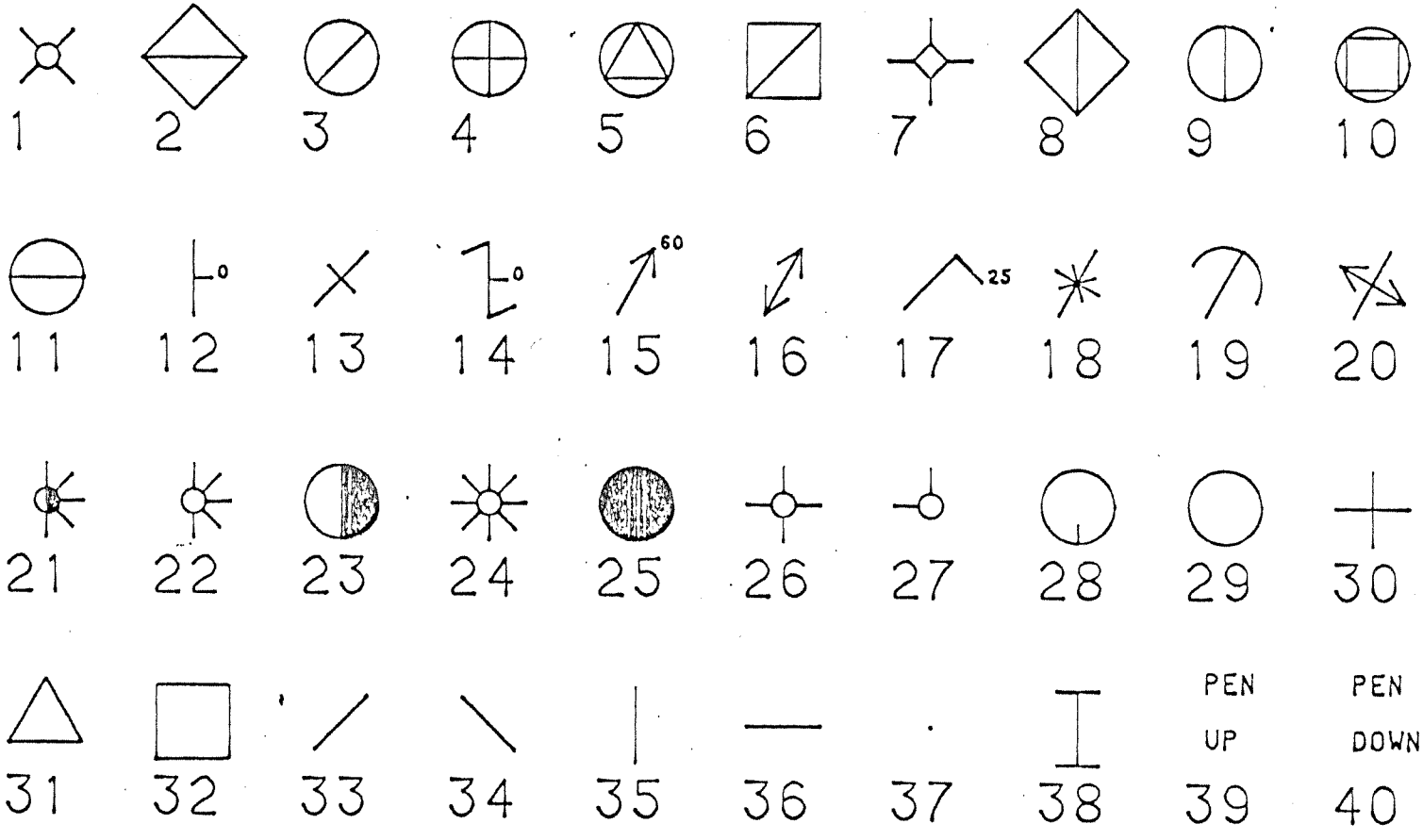
---

7. This is a system dependent feature.

feature is used. Each location symbol number may be raised to the 100's, 200's or 300's, in order to specify three different pen colours or sizes. For example, "105" specifies that symbol "5" be plotted by pen number one on the plotter; "205" specifies that symbol "5" be plotted by pen number two and so on.

If the plotter at your computer installation has only one pen or the pen size and colour are not important, then the location symbol number can be specified as it is shown in the location symbol directory in Table II, e.g. "5".

LOCATION SYMBOLS



IX. SELECTION CAPABILITIES

A. EXPRESSIONS

The selection capability within GPR, can be applied to a location symbol, or to a "LABEL" belonging to a particular location symbol. The selection syntax is compatible with ASCII FORTRAN IV, except for the use of "WHEN", "LOGIC" and "END". This facility is available by means of the selection parameter set.

```
|"WHEN      conditions"          |
|"LOGIC     logical expressions"|
|"END"                           |
```

Refer to Chapter V.E-1.1 for an introduction to the selection parameter set.

Each condition is in the form "#n EXPRESSION", where "#n" is the sequence number of the condition and the expression is of the form operand 1 <operator> operand 2.

For example "#1 TOP_depth - BOTTOM_DEPTH". As noted in Chapter V there are three types of expressions; arithmetic, relational and logical expressions.

1. Arithmetic Expressions

1.1 Arithmetic expressions are of the form

(numeric operand1) arithmetic operator

(numeric operand2)

Valid numeric operands are:

(a) a numeric variable

(b) a numeric constant

(c) an arithmetic expression.

Each numeric operand must be a REAL (F) or INTEGER (I) type as defined in FORTRAN (refer to Chapter X for details). An expression must be less than 73 characters in length.

The result of an arithmetic expression will be INTEGER only if both operands are INTEGER. If a REAL operand and an INTEGER operand are used in an arithmetic expression, the result is a REAL value.

A simple arithmetic expression consists of numeric variables and/or a numeric variable and a numeric constant. For example:

```
NET.COAL/SEAM.THICKNESS      ("/" = divide)
TOP.DEPTH-BOTTOM.DEPTH
NET.COAL * 0.5               ("*" = multiply)
TOP.DEPTH/NET.COAL
```

Refer to number 6 in Appendix I.

Compound arithmetic expressions consist of more than one simple arithmetic expression as an operand.

e.g. TOTAL.DEPTH/NET.COAL/15

Which is the last example above, divided by 15.

In order to ensure that the evaluation of an expression is performed correctly, it is good

practice to use parentheses in a compound expression. The arithmetic expressions are evaluated using FORTRAN IV conventions.

e.g. (TOTAL.DEPTH/NET.COAL)/15.

Refer to figure 6 and number 6 in Appendix I.

For further details refer to any text relating to FORTRAN IV programming.

2. Relational Expressions

Relational expressions are of the form.

(operand1) relational (operator operand2)

2.1 The relational operators are:

=	equal	
¬=	not equal	(varies with computer terminal types)
>	greater than	
<	less than	
>=	greater than or equal	
<=	less than or equal	

These special characters are all standard ASCII characters.

2.2 There are two types of relational expressions.

(a) numeric relational expressions

(b) alphanumeric relational expressions

(a) Numeric Relational Expressions

Numeric relational expressions compare the value of one numeric operand to the value of another numeric operand. Valid numeric

operands are:

(i) numeric variables

(ii) numeric constants

(iii) arithmetic expressions.

A comparison performed between INTEGER values is an INTEGER number comparison. Similarly, a comparison between REAL numbers is a REAL number comparison. However an INTEGER and a REAL value are compared as REAL numbers.

e.g. BOTTOM_DEPTH >=100.

NET_COAL <= SEAM_THICKNESS

(BOTTOM_DEPTH/NET_COAL) <= 15.

(TOP_DEPTH-BOTTOM_DEPTH) >=

(0.8 * SEAM_THICKNESS)

Refer to figure 6 and number 6 in Appendix I.

(b) Alphanumeric Relational Expressions

Alphanumeric expressions are of the form

(alphanumeric operand1) relational operator (alphanumeric operand2)

Valid alphanumeric operands are:

(i) an alphanumeric variable

(ii) an alphanumeric constant called a literal.

A literal is a string of 1 to 30 characters

enclosed in single quotation marks.

for example:

(i) HOLE.NAME = SEAM.NAME.ROOT

(ii) HOLE.NAME = 'PRAIRIE'

Valid operators used within an alphanumeric relational expression are: = , ¬= , >=, <=, > and <.

Refer to figure 6 and number 6 in Appendix I.

3. Logical Expressions

Logical expressions are of the form:

(relational expression) logical operator (relational expression).

Permissible logical operators are:

AND

OR

Relational expressions are defined in section 2 of this chapter.

Logical expression may be defined to GPR in two ways.

3.1 "LOGIC" statement.

Relational expressions are first defined for a "LOCSYM" or "LABEL". in the form:

When #1 relational expression

#2 relational expression

to a maximum of 30 conditions

which is followed by a "LOGIC" parameter of

the form.

LOGIC #1 AND #2 END

or LOGIC #1 OR #2 END

The "LOGIC" statement is used as a means to link a set of conditions. It is intended primarily to simplify complex logical expressions. The "LOGIC" statement must be terminated by "END", in the same way as the variable name list (VARNAMS). Refer to figure 6 and number 6 in Appendix I.

3.2 Alternatively, one condition can contain several relational expressions (up to a maximum of 5), related by logical operators (refer to any FORTRAN IV programming text for details).

However, even when there is only one condition, the "LOGIC" statement must be used.

e.g. LOGIC #1 END

X. THE VARIABLE DIRECTORY

The variable directory contains all the possible variable names to be used within GPR. In order to use GPR, a variable directory must be constructed. It is used to verify all variable names, and their corresponding type and length during syntax checking. This directory does not have to be specific to one particular set of data but rather it should be of general use. The directory is readily updated by appending the new variables to the end of the directory and incrementing the number of variables in the directory (entry 1 - see below), by the number added. For each variable name, the type and length are specified. The type of a variable (analagous to FORTRAN IV data type), denotes the kind of data which exists as a value for a given variable. Valid variable types are:

INTEGER (I) for integer numbers

REAL (F) for real numbers

ALPHANUMERIC (A) for alphabetic or alphanumeric character strings.

The single character enclosed in parentheses beside each type is the code for that type, which should be entered where appropriate in the variable directory without the parentheses. Each entry is input on a new line. GPR does not check any of the syntax in this directory. Therefore care should be used to avoid spurious results at run time.

Refer to Table I for an example of a variable directory. The format of the variable directory is:

Entry 1: columns 1 to 3

the number of variable names in the directory right justified.

Each subsequent entry has the format:

columns 1 to 30

variable name, left justified.

column 31 variable type, A, I, or F.

columns 32-33

-length of the variable. If the variable type is F-type, one space must be added for each of the decimal point and the numeric sign, when present, for F-type.

The specification of variable value length should be the maximum expected in the data. If this maximum is exceeded, errors will occur during input. Entries 2 to 9 inclusive in the directory are reserved for special kinds of variables, and therefore must be left blank if not used.

Entry 2 STRIKE - Type and length I03.

e.g. the strike of a planar feature, expressed as an integer, in degrees. Any variable name can be used. That is, STRIKE is not a reserved word, but the data value must have an orientation in the range 0 to 360 degrees.

This variable must be used in conjunction with entry 3 (DIP) or entry 3 and entry 4 (DIP

QUADRANT).

Entry 3 - DIP - Type and length I02.

e.g. the dip of a planar feature, expressed as an integer, in degrees. Negative dips are not accepted.

This variable must be used in conjunction with either entry 2 (STRIKE) or entry 5 (DIP DIRECTION).

Entry 4 - DIP QUADRANT - Type and LENGTH A01

e.g. the quadrant towards which the direction of dip points. Permissible quadrants are N,E,W and S, expressed as single characters. Where N= north, etc. This variable must be used in conjunction with entry 2 (STRIKE) and entry 3 (DIP).

Entry 5 - DIP DIRECTION - Type and length I03.

e.g. the direction of dip expressed in degrees azimuth. Range: 0 to 360 degrees. This variable must be used with entry 3 (DIP).

Entry 6 - TREND - Type and length I03.

The direction of trend of a linear feature. It has a range of 0 to 360 degrees and must be used in conjunction with entry 7 (PLUNGE).

Entry 7 - PLUNGE - Type and length I02.

The vertical angle between a linear feature and the horizontal. It must be used in conjunction with entry 6 (TREND).

Entry 8 - The 'X' location coordinate of a data point.

This variable is compulsory. Type and length are user specified, e.g. UTM.EASTING.

Entry 9 - The 'Y' location coordinate of a data point. This variable is compulsory. Type and length are user specified, e.g. UTM.NORTHING.

Variable names must be less than or equal to 30 characters in length. Permissible characters include; A-Z, 0-9 and (.) dot. The first character of a variable name must be from A-Z. All commands, command values, parameters and parameter values are reserved words, and must not be used as variable names.

TABLE I: A sample variable directory.

87

STRIKE	I03
DIP	I02
DIP_QUAD	A01
DIP_DIRECTION	I03
TREND	I03
PLUNGE	I02
UTM_EASTING	I06
UTM_NORTHING	I07
SEAM_NAME_ROOT	A08
SEAM_NAME_MODIFIER	A02
TOP_DEPTH	I04
TOP_ELEVATION	I04
SEAM_THICKNESS	F04

BOTTOM_DEPTH	I04
BOTTOM_ELEVATION	I04
NET_COAL	F06
TOTAL DEPTH	I03
URN	I05
REFERENCE_NO	A05
NTS	A03
HOLE_NAME	A08
HOLE_NUMBER	I03
LSD	I02
LSD_MODIFIER	I02
SEC	I02
TWP	I02
RGE	I02
MER	I01
LATITUDE	F08
LONGITUDE	F09
UTM_ZONE	I02
LOCATION_UNCERTAINTY	A06
ELEVATION	I04
BOTTOM_HOLE_ELEV	I04
TYPE_OF_DRILL	A01
CUTTINGS	A01
SIDEWALL_SAMPLES	A01
CORE	A01
MUD_COND	I04
WATER_COND	I04

COND_UNITS	A12
ELECTRIC_LOG	A01
DENSITY_LOG	A01
GAMMA_LOG	A01
CALIPER_LOG	A01
COAL	A01
NO_OF_SEAMS	I02
URANIUM	A01
DEEPEST_UNIT	A02
TOPHOLE_UNIT	A02
SOURCE_OF_DATA	A20
UNIT_NAME	A02
UNIT_RANK	A01
UNIT_MODIFIER	A01
UNIT_AGE	I03
UNIT_TOP_DEPTH	F05
UNIT_TOP_ELEV	F06
UNIT_BASE_DEPTH	I03
UNIT_BASE_ELEV	I04
UNIT_THICKNESS	I03
UNIT_PENETRATED	A01
MAJOR_LITHOLOGY	A02
MAJOR_LITHOLOGY_PERCENT	I03
MINOR_LITHOLOGY	A02
MINOR_LITHOLOGY_PERCENT	I03
COAL_BEARING	A01
SEAM_NUMBER	I02

STRAT_UNIT	A02
OVERLYING_INTERSEAM_INT	I03
MAJOR_OVERLYING_LITH	A02
MINOR_OVERLYING_LITH	A02
ROOF_LITHOLOGY	A02
FLOOR_LITHOLOGY	A02
SAMPLED	A03
DENSITY_DETAIL	A01
CALIPER_DETAIL	A01
MIN_OVERBDN_RATIO	F05
YEAR	I02
MONTH	I02
DAY	I02
HOLEID	A12
ELEVN	F05
ELEVUNIT	A01
DEPTH	F04
DEPTHUNT	A01
NUMCOALS	I03
MAXCOAL	F04

XI. A FAST METHOD TO SPECIFY VARNAMS AND THEIR FORMAT

For the user with data sets which are to be accessed over a long period of time, and/or the user with data which have a large number of variables and a correspondingly long format statement, this is a useful feature. The variable name list ("VARNAMS" - FILE section) and "FORMAT" (FILE section) are stored in a permanent data set. This descriptive data set is automatically accessed by GPR when the user specifies "SELF" in the "VARNAMS" and "FORMAT" commands in the FILE section. The variables specified within this file are checked against the variable directory for syntax.

This descriptive data set must have the following form.

Record (line) 1 - The FORTRAN IV format statement including opening and closing parentheses.

e.g. (I6, 2X,I7,3X,F5.1).

This line must not exceed 80 characters.

Record (line) 2 - A list of the variable names which are to be used in the data set. Each variable name should be separated from the next by a comma, and the entire list terminated by "END". This variable list can extend over more than one record (line). However, if a variable name is extended over more than one line, it must extend to column 80 before the break.

e.g. (I5,I4,1X,I3,1X,I3,1X,I3)

UTM_EASTING,UTM_NORTHING,DIP,STRIKE,URN END

Care should be taken to ensure that an end-of-file is only entered at the end of the FORVAR file. Otherwise GPR may abort abnormally.

XII. ERRORS

1. How GRP Handles Errors

GPR checks syntax character by character and word by word. That is, data is read in character by character until the end of a word is delimited. A word is defined as any set of consecutive characters from the set A-Z, 0-9 and the dot. Each word is compared with the table of permissible words, as outlined in Table I and the variable directory. Particular attention is given to the language hierarchy. If an error is detected, an error diagnostic is issued, the current input record (line) is ignored and a repeat of the current command or parameter and its value is expected. If you are prompted with a command or a parameter after an error is detected, enter only the expected value.

2. How to Correct Errors

If you detect an error during the specification of a particular map, simply re-enter, when prompted, the section in which the error was made and re-specify the command / parameter in partial prompt mode. Refer also to the "DELETE" and "CLEAR" commands. If errors are found in the LEGEND feature, the entire LEGEND must be respecified.

TABLE II: The GPR Language.

SECTION		COMMANDS	COMMAND VALUES	LEVEL 1 PARAMETERS	LEVEL 1 VALUES	LEVEL 2 PARAMETERS	LEVEL 2 VALUES
		CLEAR					
	F	END					
		PULL					
	F	RETURN					
	F	STOP					
FILE							
	C	FORMAT	fixed				
			self				
	C	NAME	filename				
	C	VARNAMS	list ,self				
	C	XYUN	metres				
			feet				
MAP							
	D	BUILDSYM	on,off				
		DELETE	locsym i				
	D	DELTATIC					
	D	LIST	varnams				
	C	LOCSYM	nnn				
			previous				
				WHEN LOGIC	conditions logical expressions		
				END			
				LABEL	varnams literals nextline	WHEN LOGIC	conditions logical expressions
						END	
	D	NDECP	integer				
		NOLIST					
	D	OUTFORM	fixed				
	D	SCALE	real no.				
		S.TITLE	' '				
	D	S.T.H(subtitle height)	real no.				
	D	SYM.H(symbol height)	real no.				
		TITLE	' '				
	D	T.H(title height)	real no.				
	C	XMAX					
	C	XMIN					
	C	YMAX					
	C	YMIN					

LEGEND	EXPLANATION FOOTING MAPID					
DISPLAY						

[ ] = selection parameter set

F commands which may only be used at fixed points.

C compulsory commands

D commands which when omitted, have a default

TABLE III

SYMBOL NUMBER	INPUT REQUIREMENT
12	STRIKE and DIP - by convention the direction of dip is clockwise of the strike azimuth.
or	STRIKE, DIP and DIP QUADRANT - where dip quadrant can be one of N, S, E, W and the dip is 0-90 degrees.
or	DIP and DIP DIRECTION
13	STRIKE and DIP - where dip must be assigned a value of 0.
14	STRIKE and DIP - where dip is equal to 90 degrees (i.e. vertical).
15	TREND and PLUNGE - where trend is in the range 0-360 degrees.
16	TREND and PLUNGE, where plunge is equal to 0.
17	STRIKE and DIP - by convention the direction of dip is clockwise of the strike azimuth.
18	TREND and PLUNGE
19	TREND and PLUNGE - where plunge is equal to zero.
20	TREND and PLUNGE

APPENDIX I

A. Commands use to generate figures in the text.

Number 1 (Figure 1)

```
FILE
FULL
FIXED
(F6.0,A1,F6.0,7A2,A1,I3,7A2,A1,2I3,2F4.1,3A1)
UTM_EASTING,ELECTRIC_LOG,UTM_NORTHING,HOLE_NAME,HOLE_NUMBER,
SEAM_NAME_ROOT,TOP_DEPTH,BOTTOM_DEPTH,NET_COAL,SEAM_THICKNESS,
DENSITY_LOG,GAMMA_LOG,CALIPER_LOG END
METRES
RETURN
MAP
TITLE='FIGURE 1'
S_TITLE='THE LOCATION OF COAL EXPLORATION BOREHOLES IN THE AREA'
XMIN=157000.
XMAX=177000.
YMIN=230000.
YMAX=249000.
SCALE=100000.
LOCSYM=101
RETURN
DISPLAY
STOP
```

Number 2 (Figure 2)

```
FILE
FULL
FIXED
(F6.0,A1,F6.0,7A2,A1,I3,7A2,A1,2I3,2F4.1,3A1)
UTM_EASTING,ELECTRIC_LOG,UTM_NORTHING,HOLE_NAME,HOLE_NUMBER,
SEAM_NAME_ROOT,TOP_DEPTH,BOTTOM_DEPTH,NET_COAL,SEAM_THICKNESS,
DENSITY_LOG,GAMMA_LOG,CALIPER_LOG END
METRES
RETURN
MAP
TITLE='FIGURE 2'
S_TITLE='COAL SEAMS INTERSECTED BY BOREHOLES WITHIN THE AREA'
XMIN=157000.
XMAX=177000.
YMIN=230000.
YMAX=249000.
SCALE=100000.
LOCSYM=10 1
LABEL=SEAM_NAME_ROOT ' ' NEXTLINE SEAM_THICKNESS ' '
RETURN
LEGEND
NAME AND THICKNESS OF THE UPPERMOST COAL SEAM
INTERSECTED BY THE BOREHOLE


'72 L 14'
DISPLAY
STOP
```

Number 3 (Figure 3)

```
FILE
FULL
FIXED
(F6.0,A1,F6.0,7A2,A1,I3,7A2,A1,2I3,2F4.1,3A1)
UTM_EASTING,ELECTRIC_LOG,UTM_NORTHING,HOLE_NAME,HOLE_NUMBER,
SEAM_NAME_ROOT,TOP_DEPTH,BOTTOM_DEPTH,NET_COAL,SEAM_THICKNESS,
DENSITY_LOG,GAMMA_LOG,CALIPER_LOG END
METRES
RETURN
MAP
TITLE='FIGURE 3'
S_TITLE='BOREHOLES WHICH INTERSECTED COAL SEAM >= 5 FT IN THICKNESS'
XMIN=157000.
XMAX=177000.
YMIN=230000.
YMAX=249000.
SCALE=100000.
LOCSYM=101
WHEN #1 SEAM_THICKNESS >=5
LOGIC #1 END
LABEL= SEAM_NAME_ROOT ' ' NEXTLINE SEAM_THICKNESS ' '
RETURN
LEGEND
SEAM_NAME
SEAM_THICKNESS


'72 L 14'
DISPLAY
STOP
```

Number 4 (Figure 4)

```
FILE
FULL
FIXED
(F6.0,A1,F6.0,7A2,A1,I3,7A2,A1,2I3,2F4.1,3A1)
UTM_EASTING,ELECTRIC_LOG,UTM_NORTHING,HOLE_NAME,HOLE_NUMBER,
SEAM_NAME_ROOT,TOP_DEPTH,BOTTOM_DEPTH,NET_COAL,SEAM_THICKNESS,
DENSITY_LOG,GAMMA_LOG,CALIPER_LOG END
METRES
RETURN
MAP
TITLE='FIGURE 4'
S_TITLE='COAL SEAMS >= 5 OR <5 FT THICK'
XMIN=157000.
XMAX=177000.
YMIN=230000.
YMAX=249000.
SCALE=100000.
LOCSYM=101
WHEN #1 SEAM_THICKNESS >=5 LOGIC #1 END
LABEL=SEAM_NAME_ROOT ' ' NEXTLINE TOP_DEPTH '/' SEAM_THICKNESS ' '
LOCSYM=132 WHEN #1 SEAM_THICKNESS < 5 LOGIC #1 END
LABEL=SEAM_NAME_ROOT ' ' NEXTLINE TOP_DEPTH '/' SEAM_THICKNESS ' '
RETURN
LEGEND
COAL SEAM >=5 FT THICK

COAL SEAM < 5 FT THICK


'72 L 14'
DISPLAY
STOP
```

Number 5 (Figure 5)

```
FILE
FULL
FIXED
(F6.0,A1,F6.0,7A2,A1,I3,7A2,A1,2I3,2F4.1,3A1)
UTM_EASTING,ELECTRIC_LOG,UTM_NORTHING,HOLE_NAME,HOLE_NUMBER,
SEAM_NAME_ROOT,TOP_DEPTH,BOTTOM_DEPTH,NET_COAL,SEAM_THICKNESS,
DENSITY_LOG,GAMMA_LOG,CALIPER_LOG END
METRES
RETURN
MAP
TITLE='FIGURE 5'
S_TITLE='GPR LOCATION SYMBOL OVERLAY FACILITY (BUILDSYM)'
BUILDSYM=ON
XMIN=157000.
XMAX=177000.
YMIN=230000.
YMAX=249000.
SCALE=100000.
LOCSYM=129
LOCSYM=134 WHEN #1 ELECTRIC_LOG='Y' LOGIC #1 END
LOCSYM=133 WHEN #1 DENSITY_LOG='Y' LOGIC #1 END
LOCSYM=135 WHEN #1 GAMMA_LOG='Y' LOGIC #1 END
LOCSYM=136 WHEN #1 CALIPER_LOG='Y' LOGIC #1 END
RETURN
LEGEND
BOREHOLE LOCATION


ELECTRIC LOG PRESENT


DENSITY LOG PRESENT


GAMMA LOG PRESENT


CALIPER LOG PRESENT



'17 L 14'
DISPLAY
STOP
```

Number 6 (Figure 6)

```
FILE
FULL
FIXED
(F6.0,A1,F6.0,7A2,A1,I3,7A2,A1,2I3,2F4.1,3A1)
UTM.EASTING,ELECTRIC.LOG,UTM.NORTHING,HOLE.NAME,HOLE.NUMBER,
SEAM.NAME.ROOT,TOP.DEPTH,BOTTOM.DEPTH,NET.COAL,SEAM.THICKNESS,
DENSITY.LOG,GAMMA.LOG,CALIPER.LOG END
METRES
RETURN
MAP
TITLE='FIGURE 6'
S.TITLE='A SELECTED AREA'
XMIN=157000.
XMAX=177000.
YMIN=230000.
YMAX=249000.
SCALE=100000.
LOCSYM=108 WHEN #1 NET.COAL<=0.0 LOGIC #1 END
LABEL='REJECTED' ' '
LOCSYM=125
WHEN #1 (TOP.DEPTH/NET.COAL) <= 10
#2 UTM.NORTHING <=243000 AND UTM.NORTHING >= 240000
#3 (UTM.EASTING >= 161000) AND (UTM.EASTING <= 165000)
LOGIC #1 AND #2 AND #3 END
LABEL= NET.COAL ' '
WHEN #1 SEAM.NAME.ROOT='PRAIRIE'
LOGIC #1 END
RETURN
LEGEND
NET COAL =0.0 (I.E. NO COAL INTERSECTED)


NET COAL INTERSECTED BY BOREHOLE


'72 L 14'
DISPLAY
STOP
```

Number 7 (Figure 8)

```
FILE
FULL
FIXED
(F6.0,1X,F6.0,1X,I3,1X,I3,1X,I3,2X,A1)
UTM_EASTING,UTM_NORTHING,URN,STRIKE,DIP,DIP-QUAD END
METRES
RETURN
MAP
TITLE='FIGURE 8'
S.TITLE='BEDDING PLANE ORIENTATIONS MEASURED IN THE AREA'
XMIN=157000.
XMAX=177000.
YMIN=230000.
YMAX=249000.
SCALE=100000.
LOCSYM=12
RETURN
LEGEND
BEDDING PLANE ORIENTATION


'72 L 14'
DISPLAY
STOP
```

B. Data used to generate figures in the text.

ORIENT										
166000	242500	1	50	71	E	N	S	W	0	90
165000	243000	2	45	74	E	N	S	W	0	90
165000	244000	3	52	69	E	N	S	W	0	90
164000	244000	4	46	79	E	N	S	W	0	90
164000	245000	5	44	83	E	N	S	W	0	90
163000	245000	6	40	68	E	N	S	W	0	90
162000	245000	7	42	73	E	N	S	W	0	90
163000	246000	8	180	54	E	N	S	W	0	90
164000	246000	9	195	49	E	N	S	W	0	90
165000	246000	10	180	34	E	N	S	W	0	90
165000	246500	11	187	49	E	N	S	W	0	90
165000	247000	12	165	33	E	N	S	W	0	90
163000	247000	13	177	44	E	N	S	W	0	90
166000	247000	14	198	48	E	N	S	W	0	90
166000	246000	15	190	34	E	N	S	W	0	90
167000	245000	16	126	42	E	N	S	W	0	90
167000	244000	17	87	40	E	N	S	W	0	90
162000	246000	18	48	63	E	N	S	W	0	90
162000	248000	19	192	70	E	N	S	W	0	90
161000	247000	20	168	39	E	N	S	W	0	90
162000	247000	21	180	54	E	N	S	W	0	90
164000	248000	22	161	44	E	N	S	W	0	90
164000	247000	23	171	41	E	N	S	W	0	90
165000	248000	24	160	33	E	N	S	W	0	90
166000	248000	25	178	52	E	N	S	W	0	90
166000	242000	26	45	68	E	N	S	W	0	90
167000	241000	27	48	64	E	N	S	W	0	90
168000	241000	28	50	69	E	N	S	W	0	90
168000	240000	29	48	59	E	N	S	W	0	90
170000	239000	30	51	56	E	N	S	W	0	90
169000	243000	31	92	39	E	N	S	W	0	90
169000	241000	32	66	46	E	N	S	W	0	90
170000	241000	33	51	57	E	N	S	W	0	90
171000	240000	34	39	91	E	N	S	W	0	90
172000	239000	35	46	76	E	N	S	W	0	90
173000	237000	36	33	5	E	N	S	W	0	90
172000	236000	37	64	26	E	N	S	W	0	90
171000	236000	38	37	31	E	N	S	W	0	90
171000	237000	39	38	44	E	N	S	W	0	90
170000	238000	40	45	57	E	N	S	W	0	90
169000	239000	41	52	54	E	N	S	W	0	90
173000	235000	42	95	14	E	N	S	W	0	90
172000	234000	43	77	32	E	N	S	W	0	90
175000	233000	44	56	30	E	N	S	W	0	90
175000	234000	45	52	50	E	N	S	W	0	90
175000	235000	46	53	70	E	N	S	W	0	90
175000	236000	47	37	81	E	N	S	W	0	90
172000	232000	48	173	36	E	N	S	W	0	90
173000	231000	49	173	22	E	N	S	W	0	90

```
    PRAIRIE
165310Y241080PRAIRIE          14PRAIRIE          60 80 15.1 15.1NYY
173910Y231370PRAIRIE          24PRAIRIE         110115  4.3  5.0NYY
161575Y233575PRAIRIE          63PRAIRIE         252264  7.0  8.5NYY
161800Y241000M AND M          13PRAIRIE         166185 14.6 14.6YNY
158320N232700PRAIRIE          61PRAIRIE         198208  6.5  9.5YYY
159700N241105PRAIRIE         106PRAIRIE         145163 13.8 15.1YYY
159600N244225PRAIRIE         107PRAIRIE         122143 15.8 20.8YYY
158030N242520PRAIRIE         116PRAIRIE         129145 13.1 15.9YYY
157980N244200PRAIRIE         118PRAIRIE          41 86 16.1 18.9YYN
162060N245925PRAIRIE         119                999999  0.0  0.0NYY
169750Y233020PRAIRIE         126PRAIRIE         236238  2.0  2.0NYY
166290Y239775PRAIRIE         128PRAIRIE          74 81  7.0  7.0YYN
164810Y234460PRAIRIE         130PRAIRIE         276283  5.6  7.0YYN
163260Y232800PRAIRIE         131PRAIRIE         297308  6.8  8.7YYN
161510N236030PRAIRIE         132PRAIRIE         133138  4.5  4.5YYN
164530Y242740PRAIRIE         133PRAIRIE          62 85 16.8 18.1YNN
162860Y244340PRAIRIE         136PRAIRIE         164185 15.7 21.0YYN
161210Y244610PRAIRIE         137PRAIRIE          36 49 11.6 12.8YYN
161290Y242590PRAIRIE         139PRAIRIE         102120 13.2 14.4NYY
157925Y245840PRAIRIE         140PRAIRIE          46 48  2.4  2.4YYY
158100Y240870PRAIRIE         142PRAIRIE         115132 11.8 12.8YYY
158125Y239275PRAIRIE         145PRAIRIE          64 69  3.2  5.2YYY
164920Y231230PRAIRIE          25PRAIRIE         308324  9.3 12.7YNY
162900Y242540PRAIRIE R       124PRAIRIE          55 94 18.4 21.7NYN
159780Y238150PRAIRIE         148PRAIRIE          67120 11.2 12.4YYY
162000Y239970PRAIRIE         150PRAIRIE         141160 13.6 15.5YNY
174660Y234800PRAIRIE         127PRAIRIE         229236  5.3  7.0YNY
164100N239590PRAIRIE R       196PRAIRIE          62 76 10.8 14.0YYY
159760Y239775PRAIRIE R       195PRAIRIE         128135  6.7  6.7YYN
168040Y236240PRAIRIE         129PRAIRIE         104110  5.5  5.5NYY
158180Y237640PRAIRIE ABC A    94                999999  0.0  0.0NYN
172630Y246275PRAIRIE          19                999999  0.0  0.0YYY
159675N242575PRAIRIE R       112FLATLAND        122141 15.4 19.0YYY
159600Y245900PRAIRIE         138FLATLAND        175192 13.7 17.0YYY
164500Y244400PRAIRIE         135FLATLAND         87 99 10.1 12.0YNY
166200Y242300PRAIRIE R       125FLATLAND         43 49  4.1  6.0NYY
```

APPENDIX II: System Documentation

I- INTRODUCTION

The package called Generalized Posting Routine (GPR) is designed as a graphic system for posting and verifying both oriented and non-oriented geological and geographical data.

The version supplied is operating on a Hewlett Packard 3000 Series II computer at the Institute of Sedimentary and Petroleum Geology, Calgary, Alberta. Initial development work was performed on the University of Alberta, Computing Services, Amdahl 470 V/6 in Edmonton.

II. SYSTEM OVERVIEW

The necessity of a user specification feature in the initial design phase resulted in the incorporation of the dynamic logical expression WHEN and LOGIC, and initiated two possible approaches to design and implementation. The first was to design GPR as an interpreter in which all of the user-defined logical expressions were stored and converted and in which, for each input data record, this interpreter was invoked (comparable for example, to the APL interpreter); the other approach was to design GPR as a translator in which the user's commands and parameters were translated into a program in one of the available computer languages. The program thus generated could then be executed.

In terms of CPU time, development time and cost, the former approach is inferior to that of the latter. Therefore, the second method was employed. For the purposes of portability the code generated is in FORTRAN. Technically speaking, then, GPR is a translator.

GPR is logically divisible into two parts: the parser and the code generator.

The parser is composed of a lexical analyser (SCAN) and a syntax analyser (FILE and MAPDEF). While in this phase,

all the user's commands and parameters are checked and converted into internal form.

The code generator is similarly composed of two routines (MAPGEN and GENCOD) which are used to translate all the information (already in internal form) into a FORTRAN program. The generated program can then be compiled by the FORTRAN compiler and the resulting object module executed via a system linkage editor.

III. SYSTEM DESCRIPTION

GPR is modularized in order to reduce the total amount of computer memory required at any given time during program execution: i.e., GPR can be executed by using a type of system-overlaying technique to reduce the amount of computer storage required; for example, by using the Dynamic Loading Facility on most IBM machines or the overlay facility on most CDC machines.

GPR is divided into six logical main modules (MAIN, FILE, MAPDEF, MAPGEN, LEGEND and SYM); each module performs a set of distinct and unique functions and is composed of one or more subroutines. Communication between modules and subroutines is provided by the FORTRAN common block facility. All of the data and commonly used information are stored in eleven common blocks (INPUT, SCANS, FILES, CLIST, LEGD, RPRGFX, MAPS, CNT, SYMBOS, PARFUL and SPC).

Only the main module and the eleven common blocks must constantly reside in the computer memory during program execution. The rest of the modules may be loaded (overlaid) dynamically and selectively into the memory, dependent upon the user's particular input commands and operating parameters.

IV. PROGRAM DESCRIPTION

A. MAIN Module

The MAIN module is composed of a driver program (DRIVER) and eight subroutines (CHECK, INITX, REPORT, SCAN, GETCHA, DECODE, CLASS, and BLOCK DATA). These routines are used primarily by all other modules and subroutines. They must be loaded together with all the common data blocks in memory at all time during execution.

1. DRIVER

Usage:

Function: The DRIVER is used to initialize all the global variables (or program control keys). It checks the syntax of the definition commands ('FILE', 'MAP', 'LEGEND', 'DISPLAY', and 'STOP'). Next it executes these commands (either selectively or consecutively) by transferring control to one of the modules FILE, MAPDEF, LEGEND, MAPGEN and the operating system through the use of a FORTRAN CALL statement or through a special system instruction which invokes a system dynamic loading facility.

Subroutines required: INITX, REPORT, SCAN, FILE,

2. CHECK

Usage: CALL CHECK (IC, IND, A, N, M, IMAX)

Function: This routine checks the validity of a variable name against those pre-defined in the variable names directory called PORVAR (see the User's Manual for a detailed description of the structure of the variable directory).

Parameters: IC - if IC=0, then the variable is valid.

IND - length of the variable.

A - array contains the pre-defined variables.

N - number of variables.

M - maximum length of a variable.

Subroutines required: None.

3. REPORT

Usage: CALL REPORT (I,J)

Function: To print an error message or prompt for input data.

Parameters: I - the position of the message to be printed.

J - type of message. If J=0, then the message to be printed is a prompt.

Subroutines required: None.

4. SCAN

Usage: CALL SCAN

Function: This subroutine reads all of the user's input from logical I/O unit 5 and stores it in the program variable CARD. It identifies each string of characters (according to the rules outlined in the user's manual) as a variable name, a string of characters enclosed by single quotations (e.g. 'xxx'), an integer number, a real number, a reserved word (that is, a command), a special operator (e.g. <=) or a string of unrecognized characters. It then returns this information to the calling program via the common block INPUT in accordance with the following rules:

(a) If TOKEN=1, then the input was a variable name; the name is stored in the variable IMAGE of length equal to PLACE.

(b) If TOKEN=2, then the input was a string of characters enclosed by single quotations. The characters are stored in the variable IMAGE of length equal to PLACE.

(c) If TOKEN=3, the input was an integer and the value is stored in the variable NUMBER.

(d) If TOKEN=4, then input was a real number; the value is stored in R.

(e) If 50 ≤ TOKEN ≤ 101, then input was a reserved word.

(f) If TOKEN identifies run from 120 to 134 in the following order respectively:

+ - / # ( ) , ' < > = > >= <= >=.

(g) If TOKEN=0, then input was a string of unrecognized characters.

Note: The variables CARD, TOKEN, IMAGE, NUMBER, R and PLACE are all stored in the common block INPUT (refer to the COMMON BLOCK DESCRIPTION for a detailed explanation)

Subroutines required: GETCHA, REPORT.

5. GETCHA

Usage: CALL GETCHA

Function: This subroutine reads the user's input from logical I/O UNIT 5 and stores it in the array IMAGE of the common block INPUT. GETCHA also updates the POINTER 'p' of the array 'IMAGE'.

Subroutines required: None.

6. DECODE

Usage: CALL DECODE

Function: This subroutine converts the string of characters from the array IMAGE into an integer or a real number. Upon return to the calling program it stores the value in the variable NUMBER (or R) of the common block INPUT. The maximum length of a number (real or integer) must be less than or equal to (9) nine digits including the decimal.

Subroutines required: None.

7. CLASS

Usage: CLASS (IDUMMY)

Function: CLASS is a Fortran integer function. It classifies the terminal character set into 9 different classes which are:

CLASS	CHARACTER SET
1	A-Z
2	1-9
3	0 (Zero)
4	+,-,*,/,#,(,).
5	<,>,=,¬
6	Blank Character
7	Others

8 ' (Quotation Mark)

9 - (Dot)

Parameter: IDUMMY is a dummy variable.

Subroutines required: None.

8. BLOCK DATA

Function: This is used to initialize some of the commonly used data.

B. FILE Module

The FILE is comprised of two subroutines, FILE and STREND. It is used to obtain such information as file name, variable names and input record format. The common blocks used by this module are FILES, SPC, LIST and CLIST which are described in Chapter III.

1. FILE

Usage: (i) CALL FILE

(ii) Uses a system dependent dynamic loading instruction.

Function: This subroutine (module) checks the syntax of all FILE definition commands and their related parameter values. If no errors are found, then the parameter values are saved in the common blocks FILES, LIST, SPC and CLIST.

Subroutines required: SCAN, REPORT, CHECK, STREND.

2. STREND

Usage: CALL STREND

Function: This subroutine determines the validity and existence of the orientation type variables Strike, Trend and Plunge. Upon return to the calling program, it stores the results in the common block SPC.

Subroutines required: None.

C. MAPDEF Module

This module is composed of two suboutines, MAPDEF and CONLOG. MAPDEF checks all of the MAP section commands and of the general commands.

1. MAPDEF

Usage: (a) CALL MAPDEF

(b) Use a special, system dependent, dynamic loading instruction.

Function: MAPDEF checks the syntax of all the MAP section commands, MAPID, TITLE, SUBTIT, T.H, S.T.H, XMAX, XMIN, YMAX, YMIN, DELTATIC, LOCSYM, WHEN, LOGIC and LABEL, and all of the general commands. Upon return to the calling program, it stores all of the valid commands and parameter values in the common block MAPS.

Subroutines required: REPORT, SCAN, CHECK, CONLOG.

2. CONLOG

Usage: CALL CONLOG (S1, S2, S3, S4, S5, NOLOG, RC)

Parameters: S1 - S1 is an integer array declared as INTEGER S1(30,6). It is the same as the array LAB1 and LOC1 which are defined in the common block MAPS.

S2 - S2 is an integer array declared

as INTEGER S2(30,80). It is the same as the array LOC2 and LAB2 which are defined in the common block MAPS.

S3 - S3 is an integer array declared as INTEGER S3(30,4). It is the same as the array LOC3 and LAB3 defined in the common block MAPS.

S4 - S4 is an integer array declared as INTEGER S4(30,4). It is the same as the array LOC4 and LAB 4 defined in the common block MAPS.

S5 - S5 is an integer array declared as INTEGER S5(30,30). It is the same as the array LOC5 and LAB5 defined in the common block MAPS.

NOLOC- NOLOC is the location symbol number that is chosen by the user.

RC - RC is a return code. If RC=0, then the user defined logic is valid, otherwise, the logic is invalid.

Function: CONLOG checks all of the user defined logical expressions. If no errors are found, then all of the parameter values

are saved in the array S1, S2, S3, S4
and S5.

Subroutines required: REPORT, SCAN, CHECK.

D. MAPGEN Module

This module is composed of two subroutines, MAPGEN and GENCOD. MAPGEN translates all the user's commands and parameter values (in internal form) to a Fortran main-line program. The generated program may then be compiled and executed to produce the required posting map and option printouts.

1. MAPGEN

Usage: (1) Use a special system dependent statement.

(2) CALL MAPGEN (IC)

Function: This subroutine is used to generate a Fortran program which will perform the plotting of the input data by using that information obtained by the modules FILE and MAPDEF. The information used by this module are stored in the common blocks FILE, SPC and MAPS. It generates all the Fortran declarative statements and CALLS to the plotting library (both user's defined and system plotting subroutines).

Parameters: IC is a return code. If IC=0, then the generated program has no errors; otherwise, the generated program will not be successfully compiled.

Subroutines required: GENCOD.

2. GENCOD

Usage: CALL GENCOD (S1, S2, S3, S4, ISTNO, I, KT, LSTNO)

Function: This subroutine is used to generate Fortran statements to perform the user's defined logical expression WHEN and LOGIC.

Parameters: S1, S2, S3, S4 and S5 are either LAB1, LAB2, LAB3, LAB4 and LAB5 or LOC1, LOC2, LOC3, LOC4 and LOC5, respectively. All of these arrays are defined in the common block MAPS.

Subroutines required: None.

E. LEGEND Module

This module is used to accept three lines of explanation for each location symbol and footing. The length of each line is 50 characters. Upon return to the calling program, it stores all information into the common block LEGD.

F. SYM Module

This module is composed of twenty six subroutines, AXISLD, CHGPEN, AXISZ, SYM, CIRCLE, SQUARE, TRIANG, CUBE, HCIRCLE, LINCLC, SHADED, SCIRCL, PLAGNO, LINANG, ARROW, RGTLIN, CRSLIN, DARRCN, TOPLIN, ARRLIN, DOULIN, ZLINE, DCROSS, SRTLIN and DIRECT. This module is used to perform all the plotting functions defined in the user's manual. In addition, it utilizes seven system dependent subroutines DEVWIN, VRTWIN, PLOTS, PLOT, SYMBOL, NUMBER and AX2EP the basic function of these two subroutines is to set up a mapping system, so that the SYM module can use user's coordinate system(metres and feet) rather than the actual device coordinate system (inches). For more information, see the publication University of Alberta MTS Digittal Plotting system, by P. I. Buttuls.

PLOTS - initializes the plot file.

PLOT - moves the pen in a straight line,with the pen up or down.

SYMBOL and NUMBER - draw alphabetic and numeric characters,respectively.

V. COMMON BLOCKS DESCRIPTION

A. INPUT

This common block is used to store user's terminal input (line or card image). User's input card (line) image is read and stored in the variable CARD, each identifiable string of characters (e.g. an integer number) is then stored in the variable IMAGE.

B. CLIST

CLIST is used to store user's variable names directory.

C. FILE

FILE is used to store all the valid parameter values that the user defined in the FILE section.

D. MAPS

MAPS is used to store all valid MAP section parameter values.

E. LEGD

LEGD is used to store all valid LEGEND section parameter values, i.e. description of symbols and footing.

VI. LOGICAL I/O UNITS

The following is a description of all logical input/output units used by GPR:

A. Input units

Logical unit five (5) is used by GPR to read user's commands and parameter values.

Logical unit sixteen (16) must be assigned to a user's predefined variable name directory.

Logical unit eleven (11) is used by the generated program to read user's input data file.

Logical unit fifteen (15) must be assigned to a FORVAR file, when that option is used.

B. Output units

Logical unit six (6) is used by GPR to prompt for all input data and report any errors found.

Logical unit twelve (12) is used by the generated program to write all the data records selected.

Logical unit ten (10) is used by GPR to output the generated Fortran program(s).

Logical unit nine (9) is used to generate control data for input to the generated Fortran program. This generated control data set must be assigned to logical unit five (5) when executing the generated program.