Bureau of Mineral Resources, Geology & Geophysics

R E C O R D

RECORD 1990/79

MAPDAT: A PROGRAM FOR PLOTTING SPATIAL DATA FROM A RELATIONAL DATABASE ONTO MAPS

David Collins

1990/79
c.4

RECORD 1990/79

MAPDAT: A PROGRAM FOR PLOTTING SPATIAL DATA FROM A RELATIONAL DATABASE ONTO MAPS

David Collins

Internal use only

NOTE TO USERS:

This program operates satisfactorily in BMR's current DG/ORACLE environment.

The software will not automatically be updated or modified by the Information Systems Branch to cater for any changes required by upgrades to the DG operating system or the ORACLE RDBMS.

2 October 1990

CONTENTS

CONTENTS (continued)

ILLUSTRATIONS

1. INTRODUCTION

MAPDAT is a program for plotting spatial data held in the ORACLE relational database onto any map within the Australian region at any scale.

The program enables the plotting of sample locations along with information specific to each location. The information can be displayed beside each point or in a list to the side of the map. The symbols can be sized proportionally to the value of a column in a table or a SQL expression. Town locations, survey paths, gridlines, survey areas, coastlines and other geographical lines can be plotted.

MAPDAT also includes a system for defining geological structures, thus any geological structure can be stored in the database and plotted.

The program does not compete with geographical information systems but fills a niche at a much lower level of complexity. As a result of its simplicity a minimum in setting up of data is required and using the program is very straight forward with the user always aware of the database operations being performed.

Part of the MAPDAT program is a system for storing and retrieving line data in the relational database. This system allows line data to be accessed more easily and quickly than from files. Users can store simple line data such as survey and tenement boundaries, as well as complex line data such as geological boundaries, rivers and basin outlines.

Simple line data are held in columns of type LONG within ORACLE, in a text form allowing for easy data entry and modification of data using SQLFORMS. Complex line data are held in columns of type LONG RAW in a binary form to allow for maximum efficiency when retrieving data.

By using MAPDAT, geoscientists can store their field data in a standard relational database management system and produce highly readable maps containing selected information. Besides producing standard sized maps that can be readily overlaid onto published maps, photoscale maps can also be produced to overlay airphotos.

The program has a user-friendly menu interface and sets defaults throughout to keep keyboard input to a minimum. Mapping parameter defaults are set according to Australian mapping standards so that standard maps can be produced by users lacking detailed mapping knowledge. These defaults can be easily overridden if required, for specialist map production. When selecting coastlines, the appropriate resolution coastline is chosen according to the scale being used.

In the 'Data plotting menu' the user pieces together a SQL select statement to select the points to be plotted. Several selections of data can be made and the symbol type, size and/or colour changed between selections.

MAPDAT can be run interactively or in batch.

Typing MAPDAT/HELP on the DG gives a brief explanation of MAPDAT.

2. PREREQUISITES

To run MAPDAT the user needs to know the scale of the map required and its area described in terms of the longitudes and latitudes of the four boundaries of the map. The user must also have an ORACLE username and password that gives access to the data to be plotted and must be able to put together the SQL select statement needed to retrieve the data from the database. Examples of SQL select statements are shown in appendix A.

Before running MAPDAT, it is advisable to use SQLPLUS and try out select statements that will be used within MAPDAT.

3. RUNNING MAPDAT INTERACTIVELY

This section describes the method of gaining entry to MAPDAT, the sequence of screens/menus you will encounter and how to use them to drive the application. It should be used in conjunction with the screen layouts described in appendix D and/or while running the application.

It is assumed that you are logged onto the network and the DG and have the normal DG prompt on the screen.

To run MAPDAT, simply enter:

mapdat

INITIAL INPUT (see screen 1, appendix D)

The initial information given to MAPDAT is used to log the application into the ORACLE database, create a file to contain the plot commands that will be generated and to set default mapping parameters.

ORACLE user ID/password - database userid/password required to get access to the data to be plotted.

Plot filename - the name to give to the device-independant plotfile that is produced by MAPDAT. If the file already exists its contents will be overwritten.

Scale - choose either one of the standard scales or 'another scale' to produce a non-standard scale (e.g. a photoscale). If you choose the latter you are then prompted to type in a scale value.

Maximum and minimum longitudes and latitudes - the longitudes and latitudes of the four boundaries of the map. Each longitude and latitude must be specified by two whole numbers, the first is degrees, the second minutes(1/60 degree). The numbers must be separated by spaces and/or commas. Southern latitudes are negative. Longitudes are east of Greenwich.

MAP PARAMETERS MENU (see screen 2, appendix D)

On entering this menu, the mapping parameters have already been set to the default values for producing a standard map. (See appendix A for a list of these default values.) This menu allows you to change mapping parameters should you require a non-standard map.

Projection - Universal Transverse Mercator
Lambert Conformal
Simple Conic
Modified Mercator
or Rectangular

Graticule spacing - shown on screen as latitude, longitude

Metric grid spacing - zero means no grid lines

Standard parallels - a standard parallel is the latitude at which a projection surface intersects the surface of the earth. These are only meaningful for the middle three projections shown above.

Finish with menu (and draw map) - when all mapping parameters are correct choose this option. The base map will be drawn and the coastlines menu will appear.

COASTLINES MENU (see screen 3, appendix D)

All coastline data are held as binary data in ORACLE tables. (See appendix C for details on how the data is stored.) When plotted, the lines are clipped along the border of the map. Town names are also held in an ORACLE table.

Coastline, islands and state borders (Australian) - the appropriate resolution is automatically chosen according to the scale of the map to be produced.

Basin and fold belt outlines - major structural unit boundaries on the Australian continent.

Reefs (Australia and Indonesia) - coral reefs up to about 10^{o}N lat.

Offshore boundaries - offshore international and state boundaries

Antarctic coastline - the currently available data set includes ice sheets, glaciers and shoreline but is incomplete in coverage.

Towns - at present only available for maps at 1:100 000 scale and those less detailed than 1:1 000 000 scale.

Finished with coastlines menu - when all required coastlines/towns have been drawn, choose this option. You will then be prompted for particulars about the ORACLE data you want plotted.

NAMING TABLE CONTAINING LATITUDES AND LONGITUDES AND SPECIFYING WHERE CLAUSE (see screen 4, appendix D)

Name table, latitude and longitude columns - the user is asked to name the table or tables that hold the point data that are to be plotted. The latitude and longitude columns are the names of the columns that hold the latitude and longitude for each point. The latitude and longitude columns are not applicable if you are using the simple lines/polygons option, and can therefore be left with their default values.

Where clause - specify a SQL WHERE clause to selectively plot data from a table or tables. This clause, when appended to 'SELECT ',LATITUDE COLUMN,LONGITUDE COLUMN,' FROM ',TABLE NAME(S), should form a valid SQL statement (see Appendix A). There is no need to specify that the latitude column and the longitude column is not null within the where clause, as this is checked by MAPDAT.

DATA PLOTTING MENU (see screen 5, appendix D)

The data plotting menu can be logically divided into two parts; 1) the current SQL statement and symbol settings and 2) the data plotting routines. When any data plotting routine is used, the current SQL statement and the current symbol settings are used within that routine. Thus it is necessary to set the current SQL statement and the symbol settings before using a data plotting routine.

Current SQL statement and symbol settings - the SQL statement that is defined using the following options, and the symbol settings, are used in each of the data plotting routines.

Name table, latitude and longitude columns - the user is asked to name the table or tables that hold the point data that are to be plotted. The latitude and longitude columns are the names of the columns that hold the latitude and longitude for each point. The latitude and longitude columns are not applicable if you are using the simple lines/polygons option, and can therefore be left with their default values.

Where clause - specify a SQL WHERE clause to selectively plot data from a table or tables. This clause, when appended to 'SELECT ', LATITUDE COLUMN,LONGITUDE COLUMN,' FROM ',TABLE NAME(S), should form a valid SQL statement (see Appendix A). There is no need to specify that the latitude column and the longitude column is not null within the where clause, as this is checked by MAPDAT.

Trailing clause - this is by default 'ORDER BY ',LONGITUDE NAME. This is the preferable ordering when using a pen plotter to plot the map, as this ordering minimizes the amount of paper movement during plotting.

Sometimes when selecting from more than one table, points may be duplicated. In this case change the trailing clause to 'GROUP BY ', LONGITUDE COLUMN,LATITUDE COLUMN (eg. order by dlong,dlat). This prevents duplication and retains the ordering by longitude. Sometimes you may want a completely different order (e.g. order by NAME - this will order alphabetically by NAME).

Change symbol type, size, colour, label height - there are 16 types of symbols that can be produced and 8 colours (see appendix A). The symbol height and label height can also be adjusted. These settings are used in all data plotting routines.

Point data plotting routines

Points (optionally labelled) - this routine selects points using the current SQL statement (see above) and plots the points using the current symbol settings (see Figure 1). These points can have up to three labels, each positioned next to the symbol as follows.

```
  label 1.........
* label 2.........
  label 3.........
```

The user is prompted for each label. To leave a label blank, press the NEWLINE key when prompted for the label.

Using the SQL concatentator more than one piece of information can be presented in a single label.

eg. Column(or expression) for label 1: wellname||'-'||welltype

eg. Column(or expression) for label 2: to_char(feo)||' '||to_char(mgo)

List with numbered points - this routine plots points with sequence numbers next to each point. A list is drawn to the right of the map. The left column contains the sequence numbers. Up to 6 other columns contain information from user-specified ORACLE columns.

eg.	No.	SiO2	MgO	FeO
	1	45.67	5.78	6.39
	2	48.56	6.79	3.78
	.	.	.	.
	.	.	.	.

Sometimes the required list will be too long or contain too many columns (more than 6) to be plotted onto the map. In this case use the following procedure:

1> Run MAPDAT. Select 'Points(optionally labelled)' instead of 'List Points'. When prompted for label 1, enter ROWNUM. This is an ORACLE pseudo-column that will label the symbols with sequence numbers. Take note of the exact WHERE clause used for the routine.

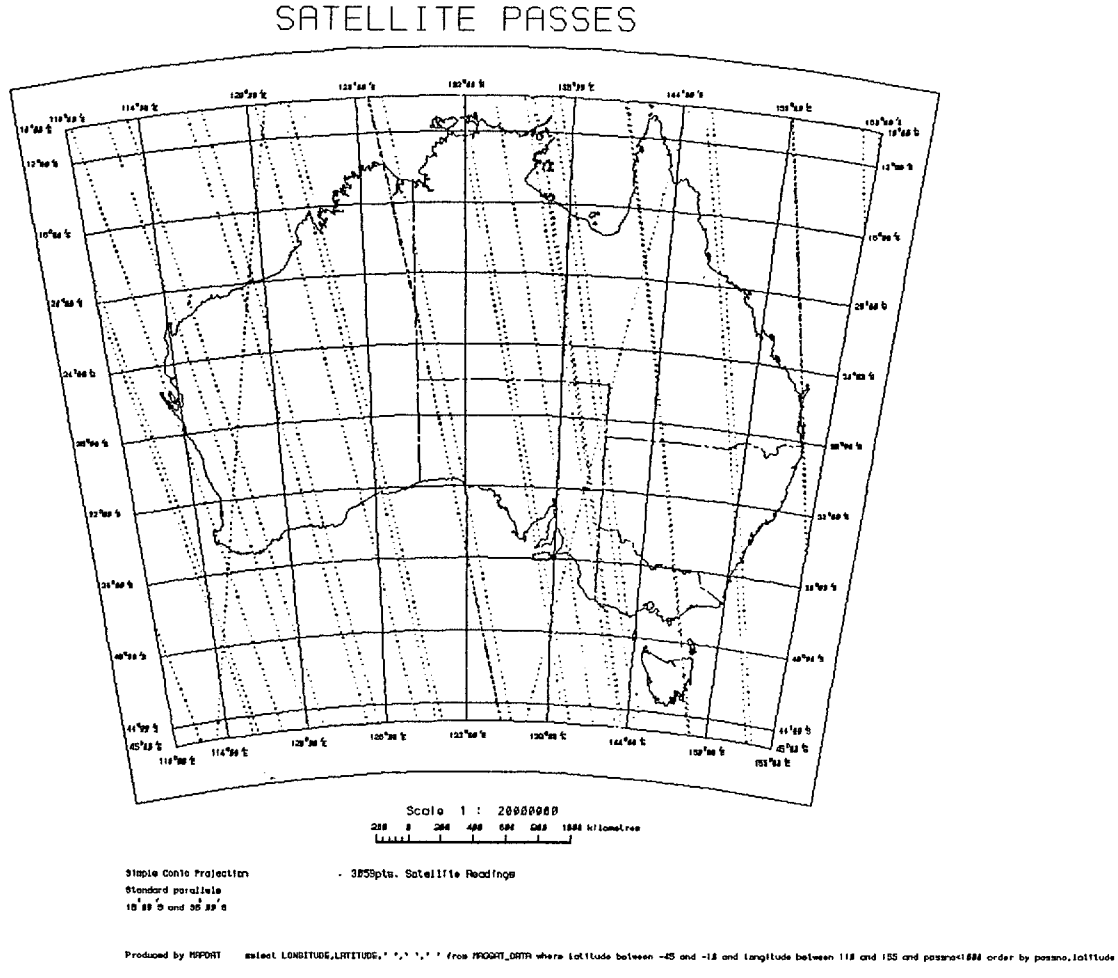


Figure 1. Plot of satellite locations held in ORACLE

2> Within SQLPLUS, enter the following commands.

```
set pagesize ...
set linesize ...
spool filename.lis
select rownum,..(other columns for list).. from ..(tables in 1)
     where ....(the same WHERE clause used in step 1)
spool off
```

Leave SQLPLUS, edit filename.lis and send it to the printer.

The above list will be ordered by ROWNUM. To order by one of the columns in the list, end the select statement with ORDER BY columnname.

Variable sized points (see figure 2) - this routine plots points (using the current SQL statement) without any labels, and indexes the symbol size against an ORACLE column or a user-specified expression. For, example if the column MgO is specified, and if the value of MgO is large at a certain location then the symbol at that point will be large. Similarly, if the value is small the symbol will be small.

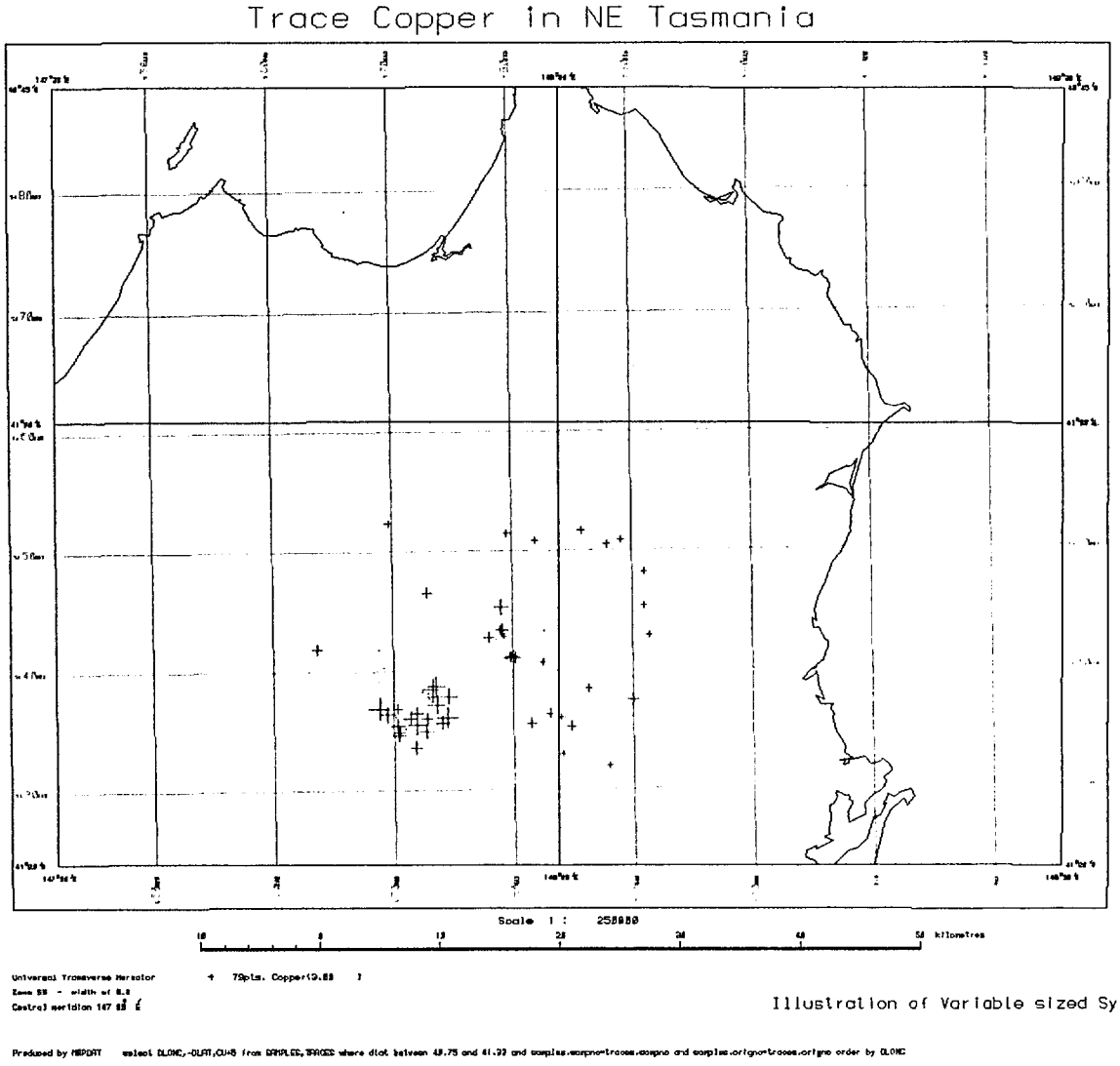


Figure 2. Symbols sized proportionally to the sample copper value

Within the routine the user is asked for the value that will be represented by a symbol 2mm high. The area, not the height, of the symbol will be proportional to the values. If the height of the symbol is to be proportional to the value then square the index column/expression. For example, in the above MgO illustration, the user would specify MgO*MgO as the ORACLE column/expression to index by.

Simple lines/polygons (see figure 3) - coastline data used by MAPDAT are held in binary strings for efficiency reasons, but for simple lines and polygons such as borders of survey areas, gridlines and flight paths it is more convenient to hold the data in text strings. The coordinates for a simple line or polygon must be held in a column of type CHAR or LONG. Each point is represented by, first the longitude (East from Greenwich), then the latitude (Negative for the southern hemisphere). The numbers must be separated by between one and five spaces.

An example of a representation of a rectangle follows:

150.3 -23.4 151.4 -23.4 151.4 -24.7 150.3 -24.7 150.3 -23.4

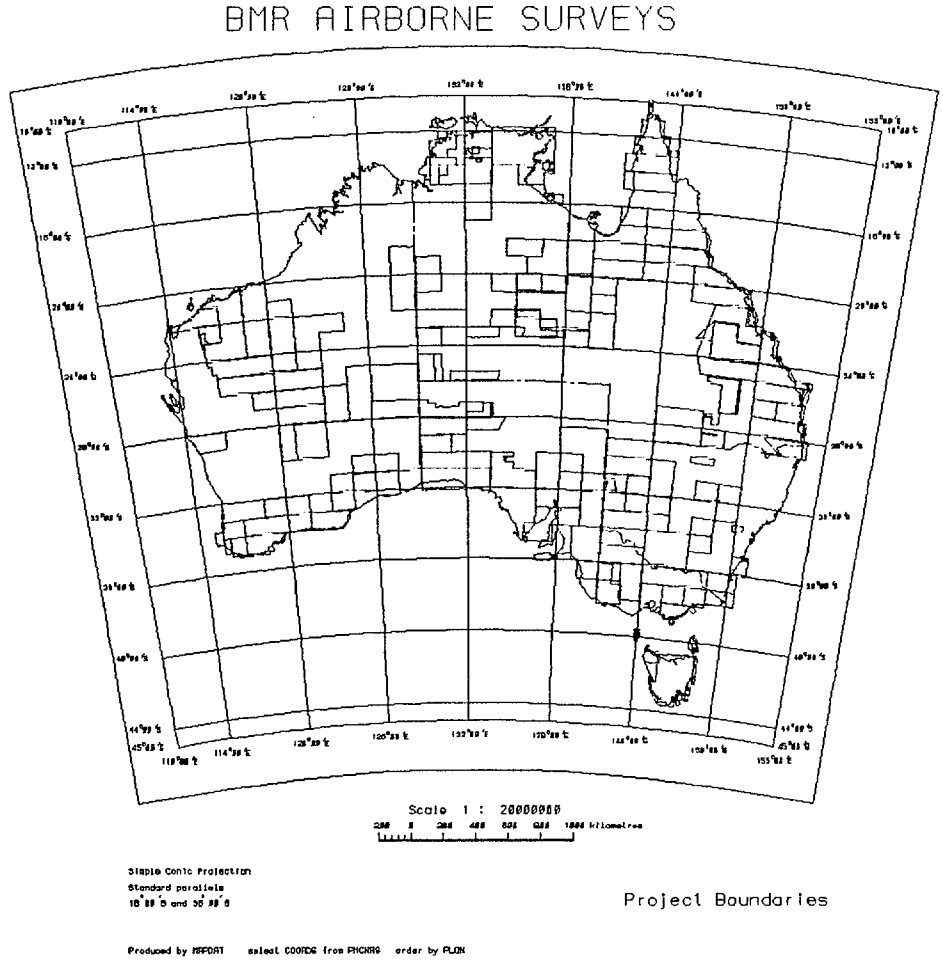


Figure 3. Plot of simple polygons stored in ORACLE

These text strings can be inserted into an ORACLE table using SQLFORMS, EDL, SQLLOADER or SQLPLUS. If no more than 240 characters are needed to represent any of the lines/polygons then these data should be entered in a column of type CHAR. If more than 240 characters are required then the column must be of type LONG. The type LONG does have some disadvantages with respect to type CHAR, one of them being that only one such column can exist in any table. The name of the column holding the text string should be called COORDS.

Some users may wish to store the values of the maximum and minimum longitudes and latitudes on the line/shape in four columns of type NUMBER and select data based on the values of these columns. If this system is adopted, it is preferable to give these values the names: ELON, WLON, TLAT and BLAT.

Note: It is helpful at the plotting stage to order the selection of COORDS by a longitude value (eg. WLON or ELON).

Geological structures (see figure 4) - any geological structure can be plotted, provided the symbol has been defined in the database. It is strongly suggested that only <u>one</u> symbol definition table should exist in the BMR and this be made available to all users. Users who have questions regarding this should contact the Information Systems Branch.

There must exist a table that contains a legend description for each symbol, a column of type LONG that contains a text string of numbers that mathematically defines the symbol shape, and a number that specifies how far from the point of observation the tip of the arrow/dip-mark is. The names of the above 3 columns <u>must</u> be LEGEND, SYMBOL and ENDPT.

A table containing structural measurements must contain the columns AZIMUTH, INCLIN, PITCH and OCTANT. For a plane structure, AZIMUTH contains the dip-direction, INCLIN contains the dip value, and PITCH and OCTANT optionally contain the pitch and pitch direction of a lineation exposed on the plane.

For line structures, AZIMUTH contains the plunge direction and INCLIN contains the plunge value. To identify the types of the structures stored, this table will also require 2 columns containing the type and subtype of the structure. It is preferable to call these columns TYPE and SUBTYPE.

For more details on setting up a structures database see appendix B.

MAP NAME AND DESCRIPTION (see screen 6, appendix D)

Map name - the map name is centred above the map. It should not normally be more than about 50 characters long. The map name cannot be omitted.

Map description - this is written centred to the right on the bottom of the map. It is up to the user to ensure that this is not so long that it overwrites other information preceding it. The map description can be left blank by pressing the NEWLINE key when prompted for it.

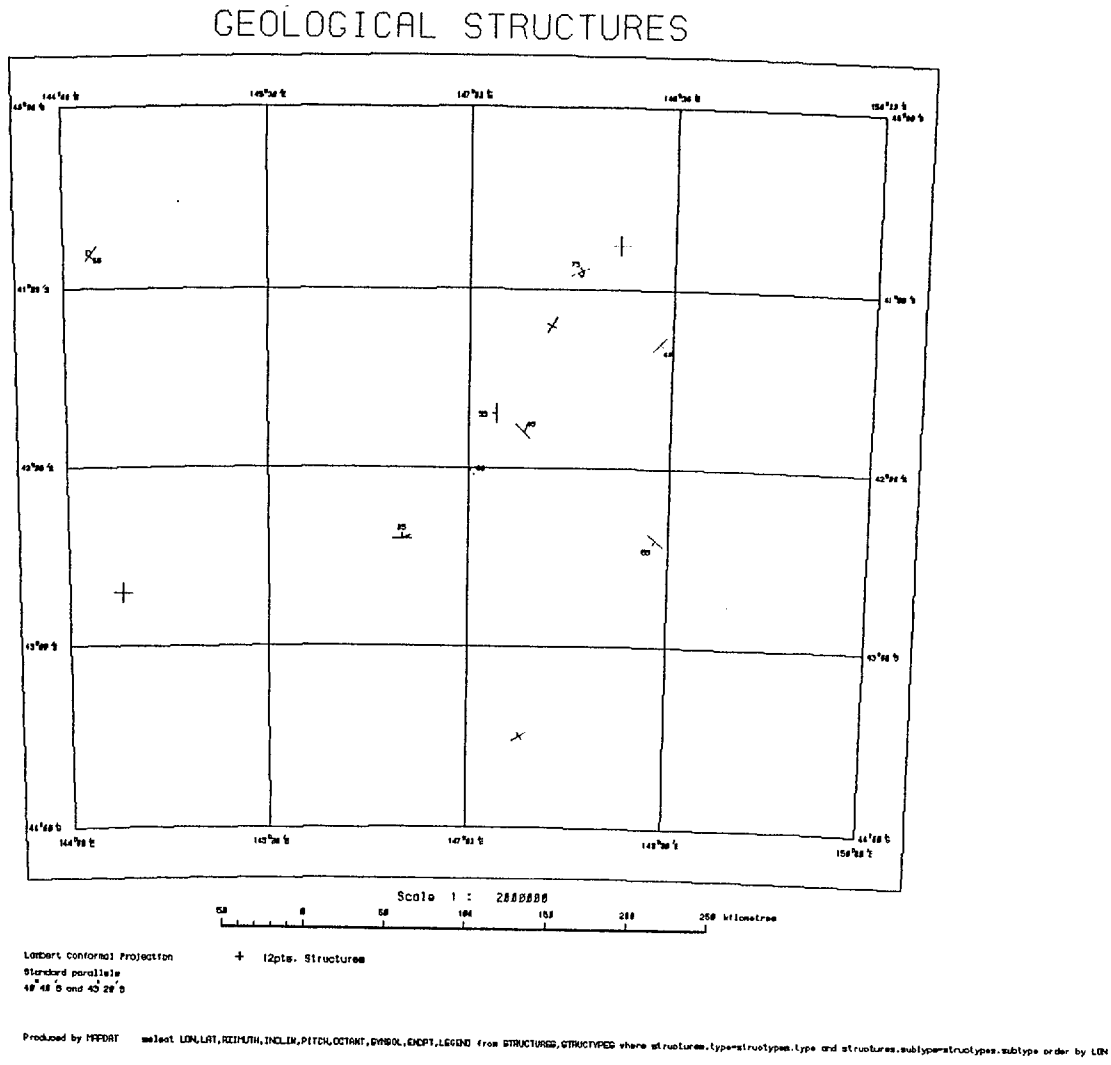


Figure 4. Plot of geological symbols defined in the database

4. RUNNING MAPDAT IN BATCH

MAPDAT can be run in batch, by typing:

mapdat controlfile

or **mapdat/queue=fast/cpu=0:4:59** controlfile

where controlfile is the name of a text file containing the responses that would be typed if MAPDAT were run interactively.

The first command line runs MAPDAT in the ordinary batch queue, thus there is no time limit on the job. The latter command line runs MAPDAT in the fast queue, but there is a CPU time limit of 5 minutes. For uncomplicated maps this time limit will usually not be reached.

To create a control file, it is easiest to copy an existing control file and then edit the copy. An example of a control file follows:

```
geochem
password
brind.plot
3
148 30 149 00
-35 00 -35 30
f                       ! finish menu 1 (don't change default map
                        parameters)
t                       ! towns on map
f                       ! finish with coastlines menu
samples,majors
dlong
-dlat
mapno=8627 and samples.sampno=majors.sampno and
samples.origno=majors.origno

c               ! change symbol                 Note: Comments can
                ! symbol type - as is           only be put after menu
                ! symbol size - as is           selections and after
                ! label size  - as is           first 6 characters of
2               ! symbol/label color - blue     symbol selections.
1               !List with points
sio2

al2o3

na2o

k2o

cao

mgo+fe2o3tot

Major Elements
f
BRINDABELLA SHEET GEOCHEMISTRY
List of Brindabella Values
```

APPENDIX A

SQL syntax, default parameters, and symbols

The SQL select statement

All retrieval of information from the ORACLE database using MAPDAT is done with a SQL SELECT statement. You can select data from one table or many tables. Following is an example of each of these cases.

Select from one table:

Table - SAMPLES

Columns -	rockno	dlat	dlong	state	..	..
	8765	34.3	146.4	NSW	..	..
	2345	22.45	144.5	QLD	..	..
	.	.	.	.	.	.

If we want the rock numbers, and the location of all samples in NSW, we use the select statement:

select rockno, dlat, dlong from samples where state='NSW'

Select from two tables:

Table 1 - STRUCTURES

Columns -	azimuth	incline	locno	..	..
	150	65	1324	..	..
	137	45	1478	..	..
	.	.	.	.	.

Table 2 - LOCALITIES

Columns -	dlong	dlat	mapno	locno	..
	121.3	28.7	3140	1324	.
	121.2	28.85	3140	1478	.
	.	.	.	.	.

If we want the locations of all the places in the Leonora sheet (mapno 3140) that have an azimuth and incline recorded, we use the select statement:

select dlong,dlat from localities,structures
where mapno=3140 and structures.locno=localities.locno and azimuth is not null and incline is not null

When using MAPDAT you need to know the full select statement for the data you want to plot. It is often beneficial to go into SQLPLUS before-hand to try out different select statements.

Examples of SQL functions and operators that may be used in MAPDAT

Operators

+ - * / - plus, minus, multiply and divide

|| character concatentation e.g. 'FRED '||SURNAME (where surname is the name of a column of type character)

IN set operator e.g. .. where state in ('NSW','ACT')

BETWEEN range specifier e.g. .. where A between 1 and 9

IS NULL/IS NOT NULL e.g. .. where FEO is not null

Arithmetic functions

CEIL(number) - round number up to an integer

FLOOR(number) - round number down to an integer

MOD(m,n) - m mod n

POWER(m,n) - m to power of n

TRUNC(m,n) - truncate n to m dec. places (m can be -ve)

ROUND(n,m) - round n to m dec. places (m can be -ve)

SQRT(number) - square root (if n<0 it returns a null)

Character functions

LOWER(char)/UPPER(char) - convert to upper/lower case

SUBSTR(char,m,n) - substring of char, starting at mth character, n characters long

INITCAP(char) - capitalise first letter of each word

Conversions

TO_CHAR(number) - convert number to character string
e.g. to_char(FEO)||' '||to_char(MGO)

Standard parameters for Australian maps - MAPDAT defaults

Scale	Map Size		Graticule Space		Grid
	Lat	Long	Lat	Long	metres
Universal Transverse Mercator					
1 : 25 000	7'30"	7'30" or 15'	1'	1'	500
1 : 50 000	15'	15'	5'	5'	1 000
1 : 100 000	30'	30'	10'	10'	1 000
1 : 250 000	1°	1°30'	30'	30'	10 000
1 : 500 000	2°	3°	1°	1°30'	no grid

Lambert Conformal

1 : 1 000 000	4°	6°	1°	$1^{\circ}30'$

standard parallels - 1/6 (ie. 40') up from bottom
- 1/6 down from top

Simple Conic (exception: 1:5 000 000 metallogenic map is Lambert conformal)

1 : 5 000 000 1 : 10 000 000	-10° to -45° 110° to 155°		4°	6°

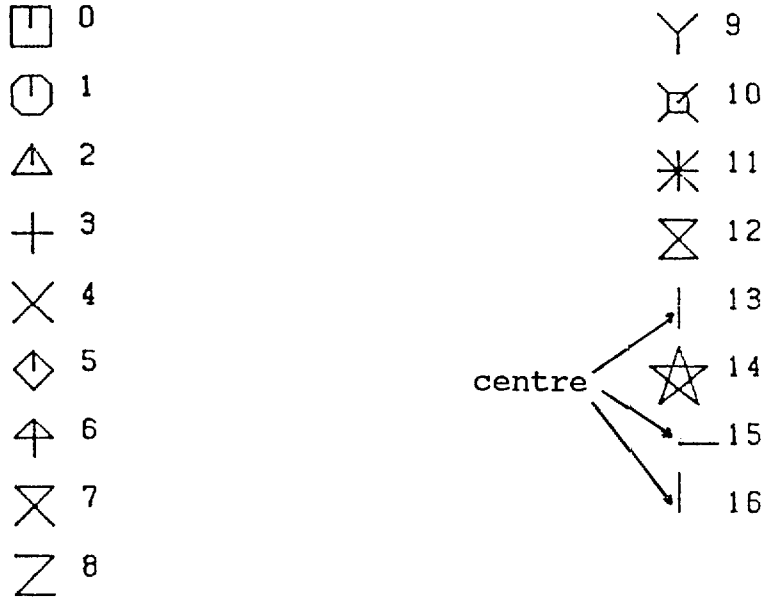
standard parallels at 18° and 36°

Pen Colour Choice for Symbols/Labels on Map

1. Black - used for graticule, border and map information
2. Blue
3. Red
4. Green
5. Brown
6. Orange - this is used for gridlines - it is not recommended as a symbol/label colour on a map with gridlines
7. Violet
8. Magenta

Symbol Numbers

There are 17 types of symbols to choose from when plotting data points in MAPDAT. Each symbol type is referenced by an integer.



APPENDIX B

Setting up a structures database

Numerical coding of symbol

Each structural symbol must be defined numerically in a text string held in a column of type LONG. The dimensions of the symbols should be taken from Symbols used on Geological Maps published by BMR. The unit of measurement used in definitions is 1/100 mm.

Symbol drawings

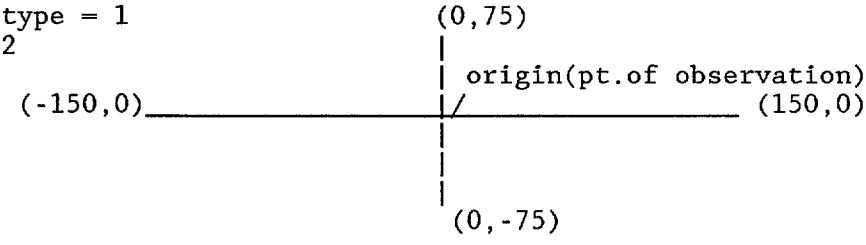
Draw each symbol on graph paper and label all coordinates necessary to define the symbol. Keep all such drawings together for later reference.

Note requirements for the drawings:

* symbols are defined for dip-direction/plunge directed Northwards
* the origin for the coordinates is at the point of observation

Example of Drawing:

Bedding - General - Vertical
Structure type = 1
Subtype = 2



Symbol Code

The symbol code is a string of numbers each separated by no more than 5 spaces. Within the symbol code string are 3 special numbers - 1001, 1002 and 1003.

These numbers have the following meaning

1001 - draw straight lines through the following coordinates.
1002 - draw a curved (smoothed) line through the following coordinates.
1003 - draw a circle or arc. The '1003' must be followed by 5 numbers; x,y of the centre, the radius, and the start and finish angle of the arc. To draw an arc clockwise, the finish angle must be less than the start angle. To draw an arc anti-clockwise, the finish angle must be greater than the start angle.

Examples:

The above symbol is represented by the following string:

1001 -150,0 150,0 1001 0,75 0,-75

Note: More than two points can follow a '1001' or a '1002'.
: You cannot use a coordinate value of 1001,1002 or 1003.

The following string will draw the top half of a circle centred at (1,1) with radius 3:

1003 1,1 3 0 180

The following string will draw the bottom half of a circle centred at (-1,5) with radius 12:

1003 -1,5 12 0 -180

Proposed type and subtype numbers - symbols could be uniquely identified using any number of identifiers. It is proposed, however, to use a type and a subtype to uniquely identify each symbol. The list of symbols will of course be extended over time, but currently comprises the following:

Type	Subtype	Legend
Planes (Types 1-19)		
1	1	Bedding (gen. dipping)
1	2	Bedding (gen. vertical)
1	3	Bedding gen. horizontal
1	4	Bedding gen. overturned
1	11	Bedding (facing definite)
1	12	Bedding vertical
1	13	Bedding horizontal
1	14	Bedding overturned
1	15	Bedding horizontal overturned
1	21	Bedding (facing unknown)
1	22	Bedding unknown vertical
1	23	Bedding unknown horizontal
2	1	Cleavage dipping
2	2	Cleavage vertical
2	3	Cleavage horizontal
2	11	Crenulation cleavage
2	12	Crenulation cleavage vertical
2	13	Crenulation cleavage horizontal
3	1	Foliation dipping
3	2	Foliation vertical
3	3	Foliation horizontal
4	1	Igneous layering dipping
4	2	Igneous layering vertical
4	3	Igneous layering horizontal
5	1	Axial surface dipping
5	2	Axial surface vertical
5	3	Axial surface horizontal
6	1	Fault dipping
6	2	Fault vertical
6	3	Fault horizontal
7	1	Vein quartz
7	2	Vein porphyry
7	3	Vein dolerite
7	4	Vein granite
7	5	Vein lamprophyre
7	6	Vein pegmatite
7	7	Vein rodingite
8	1	Joint dipping
8	2	Joint vertical
8	3	Joint horizontal

Lineations (Types 20-29)

20	1	Fold hinge
21	1	Mineral lineation
21	2	Stretching lineation
21	3	Intersection lineation
21	4	Crenulation lineation
21	5	Slickenside
21	6	Mullion
22	1	Palaeocurrent
23	1	Boudin axis

Miscellaneous

31	1	Kink band
32	1	Shearing direction
35	1	Mylonite fabric

Valid values for dip, dip-direction, pitch and octant

	Dip	Dip-direction	Pitch
Dipping (upright or overturned)	0-89	0-359	0-90
Vertical	90	0-359	0-90
Horizontal(upright or overturned)	0	0	0-90

Octant N, NE, E, SE, S, SW, W or NW

APPENDIX C

Complex line data held in ORACLE

Coordinates representing coastlines, geological boundaries, rivers and other complex lines can be stored in ORACLE in a binary form and be accessed easily and efficiently by MAPDAT and other programs containing the appropriate routines from the MAPDAT source code.

Longitudes and latitudes are stored as a string of pairs of 4-byte reals. Each pair of 4-byte reals is a coordinate (longitude,latitude) - i.e.(x,y). The longitude is expressed as degrees east of Greenwich and the latitude is in degrees (negative for the Southern Hemisphere).

The coordinate string field has associated with it, a field containing a rank value and four fields containing the westernmost and easternmost longitude and the southernmost and northernmost latitude in the line segment (i.e. the definition of the minimum bounding rectangle).

A table suitable to hold line data to be read by MAPDAT must be created with the column names:

RANK	-	this holds the rank number (used to determine how to dash the line segment)
ELON	-	the easternmost longitude in the line segment
WLON	-	the westernmost longitude
TLAT	-	the top latitude
BLAT	-	the bottom latitude
COORDS	-	the binary string containing the coordinates of the points that make up the line segment

An example of a create statement for such a table follows:

```
create table LINES
     ( RANK      number,
       ELON      number   not null,
       WLON      number   not null,
       TLAT      number   not null,
       BLAT      number   not null,
       COORDS    long raw not null)
 space sp_locn ;

create index elon_ndx on lines (ELON) nocompress pctfree=1;

create index wlon_ndx on lines (WLON) nocompress pctfree=1;

create index tlat_ndx on lines (TLAT) nocompress pctfree=1;

create index blat_ndx on lines (BLAT) nocompress pctfree=1;
```

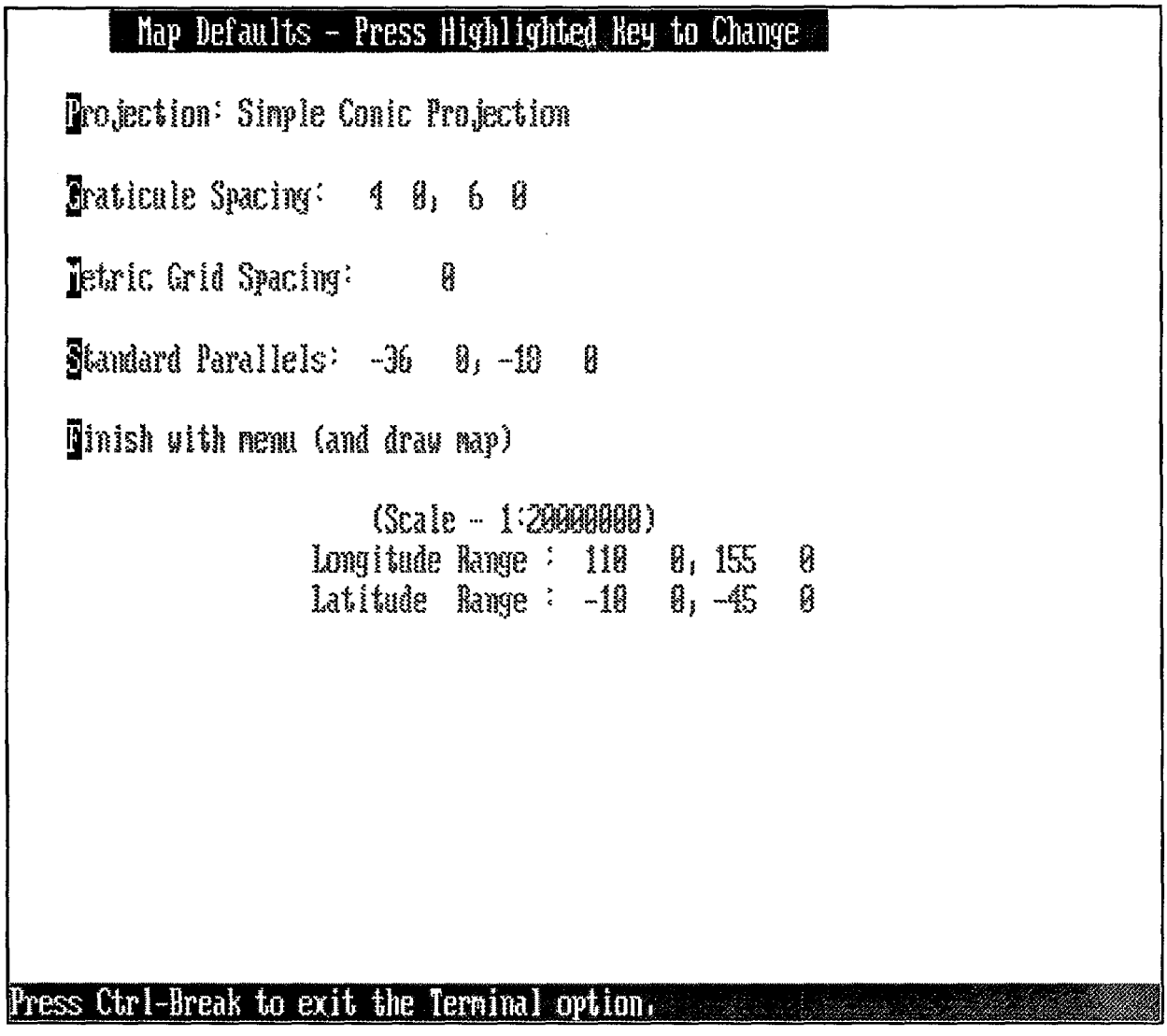
A program called WDBMDT will read a World Data Bank format file and put the line data into a nominated table of the form above. The user is prompted for the ORACLE username and password, the name of the table to put the line data into, and the name of the World Data Bank file that holds the line data. The source code of this program can be modified to read any format line file.

APPENDIX D

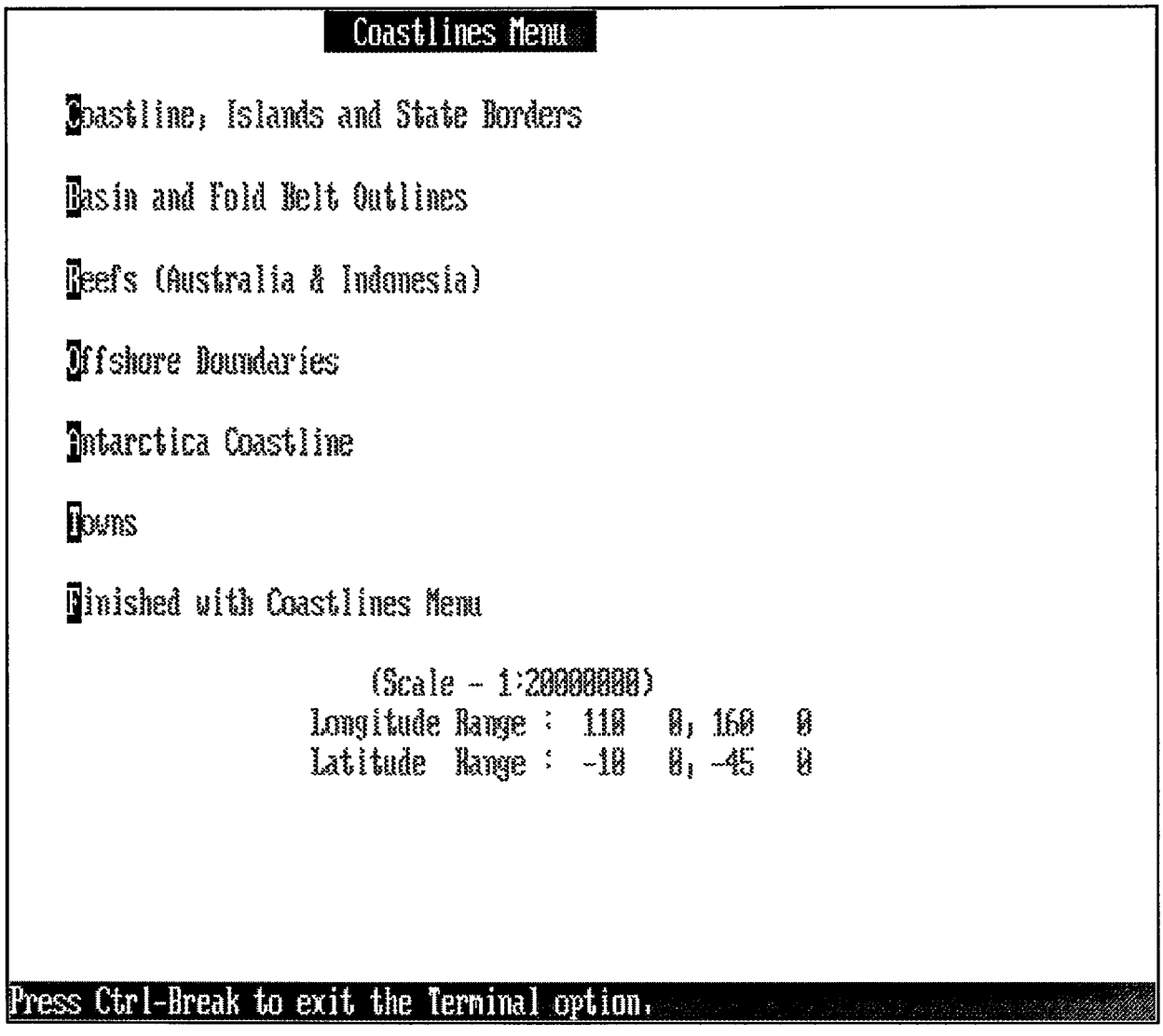
Screens within MAPDAT

```
**** MAPDAT ****

To ABORT MAPDAT at any time - Type: CTRL-C CTRL-B

ORACLE user ID: davidc

ORACLE password:

Plot file name: demo.plt

Scale Choice :

  1 ..     25,000
  2 ..     50,000
  3 ..    100,000
  4 ..    250,000
  5 ..    500,000
  6 ..  1 million
  7 ..  5 million
  8 .. 10 million
  9 .. other scale

Enter choice: 9

Enter scale - 1 : 20000000

Common Map Size for Chosen Scale:
                          map-height  map-width
 10,000,000   Simple comic  -10deg to -45deg, 110deg to 155deg

West & East longitudes (in degrees and mins.) - in order
(Example: 110 00, 160 00) : 110 0 155 0

Enter top and bottom latitude (in degrees and mins.) - in order
(Example: -10 00, -45 00) : -10 0 -45 0
Press Ctrl-Break to exit the Terminal option.
```

Screen 1. Initial input



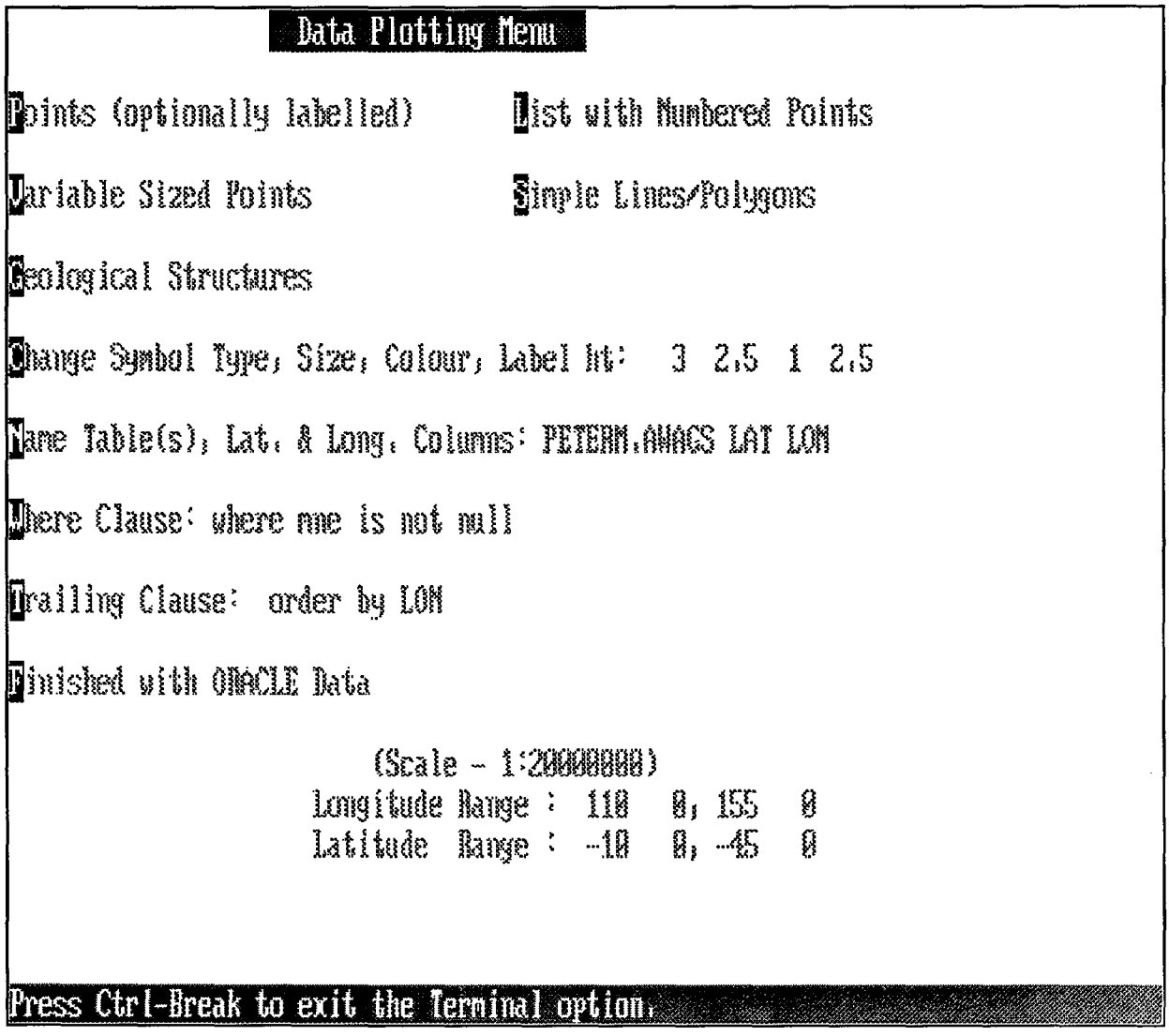
Screen 2. Mapping parameters menu



Screen 3. Coastlines menu

```
Name of table (or tables separated by commas) with data (default-DUAL)
: petern.awags

Name of longitude column(default-'0')
: lon

Name of latitude column.(If southern latitudes are not held as -ve nos. put
a minus(-) sign before the column name) (default-'0')
: lat

Complete the clause (press NEWLINE for nothing):
where,, ane is not null

Minimum and Maximum longitudes: 113 39,153  1
Maximum and Minimum latitudes : -12 22,-38 23
Number of points:         50
To change the WHERE clause type CTRL-A (else press NEWLINE)
:

Press Ctrl-Break to exit the Terminal option.
```

Screen 4. Naming table with latitudes and longitudes
and specifying WHERE clause



Screen 5. Data plotting menu

```
Enter map name: AWAGS

Map description: Australia Wide Array of Geomagnetic Stations


Map height approx:   11.8

Program completed for file DEMO.PLT
STOP

To send the plot file to the big zeta plotter -34 inch max. height-
                  or the small zeta plotter -10.5 inch max. height- type:
                                          ZPLOT/BIG    plotfile
                                       or ZPLOT/SMALL  plotfile

To send the plot file to the printer plotter -13 inch max. height- type:
                                          PLOT plotfile

To send the plot file to a tektronics terminal - type:
                                          TPLOT

Level 0
:DPJ10:DC.ADP:WORK
> plot demo.plt
Press Ctrl-Break to exit the Terminal option.
```

Screen 6. Final input and plotting options

APPENDIX E

Error handling in MAPDAT

ORACLE messages

Most error messages displayed by MAPDAT are generated by the ORACLE database management system. Some examples follow.

Error: ORA-1017: invalid username/password; logon denied

* If at the beginning of the program an incorrect username or password is entered this message appears and the program aborts.

Error: ORA-0942: table or view does not exist

* After entering the **name of the table** containing the latitude and longitude columns and the **names of the columns holding the longitudes and latitudes**, MAPDAT queries the database to verify the existence of the table and columns. If the table or columns do not exist an error message such as that above will appear and you will be prompted for the information again.

Error: ORA-0920: invalid relational operator

* This error commonly occurs when prompted for the **WHERE clause** and is most commonly the result of typing errors or incorrect syntax. If an error message follows the typing in of a WHERE clause check very carefully the line that has been typed.

The ORACLE manual 'Error Messages and Codes' will supply further information about ORACLE error messages if required.

Other error handling

When prompted for **bounding longitudes and latitudes**, four integers separated by spaces or commas are expected to be entered. If any number has a decimal point the message 'Error entering data' will appear and the user will be prompted for the four integers again. If less than four integers are entered the system waits for more numbers to be entered.

When selecting an option from a **menu**, if an invalid key is pressed the screen is refreshed.

APPENDIX F

Software required to generate MAPDAT from the source code

The following is a list of the software needed to pre-compile, compile and link the MAPDAT source code into executable code.

* ORACLE Pro-Fortran pre-compiler (Version 1.1 and later). The version of the ORACLE RDBMS will subsequently be correct if the pre-compiler is an appropriate version.

* Fortran 77 compiler

* Library with Calcomp graphics routines

  - PLOT
  - NEWPEN
  - NUMBER
  - PLOTS
  - SYMBOL
  - CIRCLE (used only in geological structures routine)

* Library with

  - ECHO_OFF (switches off keyboard input when the password is entered)
  - ECHO_ON (reverse of the above)
  - IFBATCH (function - returns 1 if the program is being run in batch, 0 otherwise)

The calls to these three routines can easily be removed from the code if they are not needed.