DEPARTMENT OF THE INTERIOR

U.S. GEOLOGICAL SURVEY

Technical Manual and Program Listing for REFORM: A Rock-Sample Database Program in FORTRAN-77

by

Todd T. Fitzgibbon[1]

Open-File Report 87-141B

[1]Menlo Park, California

1987

TABLE OF CONTENTS

INTRODUCTION

This manual documents REFORM version 2.1, and describes system requirements, files used or created by REFORM, the REFORM program and subroutines, and how to modify the REFORM software. The source code for REFORM is included at the end of this manual. Instructions for using REFORM, program specifications, and examples are contained in the companion report, User's Manual for REFORM: A Rock-Sample Database Program in FORTRAN-77, USGS Open File Report 87-141-A. It will be referred to hence as the User's Manual. The present manual assumes you are familiar with the material in the User's Manual.

REFORM is a program that allows you to create and use a database for certain kinds of rock sample data. It has sections for information on sample location, formation, rock type, characteristics (e.g., modes), major and trace element chemistry, and isotopes. It is especially suited to igneous rocks.

REFORM generates printouts of sample data, reformats chemical data for GNAP and TOMGNAP (normative calculation programs), and generates tables of selected data fields.

For portability of the database , data is stored in ASCII text files. For portability of the REFORM system, the software is written in FORTRAN-77. REFORM is currently installed on the U.S. Geological Survey Information Systems Division Digital Equipment Corporation VAX/VMS in Menlo Park. REFORM's hierarchical structure and input and output routines can be modified with relatively simple programming.

Data is stored in one or more text files. New data can be added with REFORM's interactive data entry routine, or via an editor or punched cards. Sorting of database records is done with one of REFORM's routines and/or with a text editor . Corrections to data are made with a text editor.

An initial version of REFORM was written in PL1 by Ming Ko using MRDS on a Honeywell Multics computer. Charlotte Allen assisted with program testing. The FORTRAN version described here is a new stand-alone revision with substantial extensions. It was developed in conjunction with the Pacific-to-Arizona-Crustal-Experiment (PACE) by Bob Simpson and Todd Fitzgibbon.

PROGRAM DESCRIPTION

REFORM offers five options that are displayed on the main menu. They are:

A=Make a new card file
B=Make a printout (of a card file)
C=Make a table
D=Sort a card file
E=Reformat chemistry data for TOMGNAP

If option E is chosen a second menu is displayed showing the TOMGNAP options:

1=Reformat data from the ANALYSIS 1 block
2=Reformat data from the ANALYSIS 2 block
3=Reformat data from the ANALYSIS 3 and 1 blocks

REFORM.FOR is the main loop. Option A is performed by the subroutine MAKECARDS.FOR, option B by OUTSAMP.FOR, option C by OUTTABLE.FOR, option D by SHUFFLE.FOR, option E1 by OUT1.FOR, E2 by OUT2.FOR, and E3 by OUTINAA.FOR. For options B, C, and E an existing card file must be read into memory, one sample at a time, prior to being output to a file, again one sample at a time. Nine "read" subroutines perform the input task.
See the User's Manual for details of what each option does.

HARDWARE AND SOFTWARE REQUIREMENTS

REFORM version 2.1 currently runs on the VAX/VMS system under VMS version 4.2. (See "Porting REFORM to Other Systems" in this manual if you wil not be running REFORM on a VAX/VMS system). REFORM was compiled using VAX-11 FORTRAN version 4.2. The Digital Equipment Corporation manuals for VAX FORTRAN are:

VAX-11 FORTRAN Language Reference Manual

VAX-11 FORTRAN User's Guide

REFORM has three options that format chemistry data from card files for TOMGNAP, a norm calculation and plotting program. GNAP stands for Graphic Normative Analysis Program. TOMGNAP is an extended version of GNAP by Tom Wright and Sam Priebe of the USGS. The reference manual for GNAP is:

Stuckless, J.S., and VanTrump, Jr., G., 1979, A revised version of Graphic Normative Analysis Program (GNAP) with examples of petrologic problem solving: U.S. Geological Survey Open-File Report 79-1237, 51 p.

Version 1.0 of REFORM was the Multics version. Version 2.0 was an early VAX/VMS version that included several preliminary routines that have, in the current version, been either combined or discarded.

FILES

All files created and used by REFORM are sequential formatted (ASCII) files. That is, ACCESS=SEQUENTIAL, and FORM=FORMATTED. Each line is one record.

VAX-11 FORTRAN provides a CARRIAGE CONTROL parameter for OPEN statements. CARRIAGE CONTROL=FORTRAN specifies the normal FORTRAN interpretation of the first character of each line as a carriage control character. CARRIAGE CONTROL=LIST causes the first character of each line not to be interpreteted as a carriage control character, and specifies single spacing between lines. REFORM uses both types as noted below.

DATABASE FILE FORMAT

Data are stored in one or more ASCII text files, referred to as card files. CARRIAGE CONTROL=LIST. STATUS=NEW when opened by MAKECARDS and SHUFFLE, and OLD when opened by REFORM and SHUFFLE. Each sample is stored as a set of records (cards) which are subdivided into one or more data blocks (e.g., LOCATION) containing information of a particular type. Not all kinds of data blocks need be present for any one sample. See page 4 for a printout of the database format, with underlines showing field sizes. Only those fields for which information is available need be entered. Block names (e.g., LOCATION) and field names must be spelled exactly as shown and be positioned in the exact column shown.

Note that this database has a strict column-oriented format, a characteristic inherited from the original Multics program in which input was in the form of punched cards (and thus the term "card files").

For a given sample entry, at most one copy of each block may be present. That is , only one CHARACTERISTICS block (therefore only one mode per sample), only one CHEMISTRY block (therefore only one of each of the three types of analyses provided for), etc. Each block is terminated by COMMENT=$$$ starting in column 11, and the ANALYSIS 1,2, and 3 sections of the CHEMISTRY block are terminated by $$$ starting in column 19.

Each field, with the exception of the COMMENT, SOURCE, and TEXTURE fields, has an alphanumeric key that is used to specify output items in REFORM's table routine. In general, numeric data have numeric keys and character data have alphabetic keys. Some numeric fields, such as CPS, that have variable formats or multiple entries, have alphabetic keys. See the User's Manual for a list of these keys.

Numeric fields may contain the less-than sign (<). It will appear on printouts (option B) of the data, but when the data is output for tables (option C) or for TOMGNAP files (option E) the number without the less-than sign will be used.

Refer to the User's Manual for an example of a card file, and for a glossary of terminalogy and abbreviations.

FORMAT SHOWING FIELD SIZES

```
SAMPLE NO=__________________
 LOCATION:
         MOUNTAIN RANGE=__________________
         LATITUDE=_____    LONGITUDE=______
         QUADRANGLE=_____________________________
         TRS=_________
         SOURCE=_____________________________________________
         SOURCE=_____________________________________________
         SOURCE=_____________________________________________
         SOURCE=_____________________________________________
         SOURCE=_____________________________________________
         SOURCE=_____________________________________________
         SOURCE=_____________________________________________
         SOURCE=_____________________________________________
         SOURCE=_____________________________________________
         SOURCE=_____________________________________________
         COMMENT=____________________________________________
         COMMENT=____________________________________________
         COMMENT=____________________________________________
         COMMENT=____________________________________________
         COMMENT=____________________________________________
         COMMENT=$$$
 FORMATION:
         FORMATION SYMBOL=__________________
         FORMATION NAME=_____________________________________
         COMMENT=____________________________________________
         COMMENT=____________________________________________
         COMMENT=____________________________________________
         COMMENT=____________________________________________
         COMMENT=____________________________________________
         COMMENT=$$$
 ROCK TYPE:
         ROCK TYPE=__________________________________
         GRAIN SIZE=_________________________________
         TEXTURE=____________________________________________
         TEXTURE=____________________________________________
         TEXTURE=____________________________________________
         TEXTURE=____________________________________________
         COMMENT=____________________________________________
         COMMENT=____________________________________________
         COMMENT=____________________________________________
         COMMENT=____________________________________________
         COMMENT=____________________________________________
         COMMENT=$$$
 CHARACTERISTICS:
         CPS=__________ SG=_____ CI=_____ MAG=_______
         SN=_____ SQ=_____ SP=_____ SA=_____ SM=_____
         TN=_____ TQPA=_____ TB=_____ TH=_____ TM=_____ TSP=_____ TMG=_____
         %Q1=_____ %P1=_____ %A1=_____ %Q2=_____ %P2=_____ %A2=_____ %M2=_____
         OTHER MINERALS=_____________________
         COMMENT=____________________________________________
         COMMENT=____________________________________________
         COMMENT=____________________________________________
```

COMMENT=_____

COMMENT=_____

COMMENT=$$$

CHEMISTRY:

ANALYSIS 1 BY: _____

SIO2=_____	TIO2=_____	AL2O3=_____	FE2O3=_____	FEO=_____
MNO=_____	MGO=_____	CAO=_____	NA2O=_____	K2O=_____
P2O5=_____	H2O+=_____	H2O-=_____	CO2=_____	TOTAL=_____
Y=_____	SR=_____	ZR=_____	U=_____	RB=_____
TH=_____	PB=_____	GA=_____	ZN=_____	CU=_____
NI=_____	CR=_____	V=_____	BA=_____	NB=_____

$$$

ANALYSIS 2 BY: _____

SIO2=_____	TIO2=_____	AL2O3=_____	FE2O3=_____	MNO=_____
MGO=_____	CAO=_____	NA2O=_____	K2O=_____	P2O5=_____
Y=_____	SR=_____	ZR=_____	RB=_____	BA=_____
NB=_____	FEO=_____			

$$$

ANALYSIS 3 BY: _____

FE=_____	K=_____	NA=_____	BA=_____
CO=_____	CR=_____	CS=_____	HF=_____
MN=_____	RB=_____	SB=_____	SR=_____
TA=_____	TH=_____	U=_____	ZR=_____
SC=_____	LA=_____	CE=_____	ND=_____
SM=_____	EU=_____	GD=_____	TB=_____
DY=_____	TM=_____	YB=_____	LU=_____
HO=_____			

$$$

RATIOS: BA/RB=_____ BA/SR=_____ RB/SR=_____

RATIOS: U/TH=_____ K/RB=_____ K/BA=_____

COMMENT=_____

COMMENT=_____

COMMENT=_____

COMMENT=_____

COMMENT=_____

COMMENT=$$$

ISOTOPES:

ANALYSIS=_____ ANALYST=_____ MINERAL=_____

AGE=_____ RATIO=_____

ANALYSIS=_____ ANALYST=_____ MINERAL=_____

AGE=_____ RATIO=_____

ANALYSIS=_____ ANALYST=_____ MINERAL=_____

AGE=_____ RATIO=_____

ANALYSIS=_____ ANALYST=_____ MINERAL=_____

AGE=_____ RATIO=_____

ANALYSIS=_____ ANALYST=_____ MINERAL=_____

AGE=_____ RATIO=_____

ANALYSIS=_____ ANALYST=_____ MINERAL=_____

AGE=_____ RATIO=_____

COMMENT=_____

COMMENT=_____

COMMENT=_____

COMMENT=_____

COMMENT=_____

COMMENT=$$$

TABLE FILE FORMAT

Tables have lines up to 126 characters long with data for one sample on one line. Each line contains user-specified fields aligned in columns, separated by a space.

CARRIAGE CONTROL=LIST. STATUS=NEW when opened by REFORM, and OLD when opened by SHUFFLE.

PRINTOUT FILE FORMAT

Printout files contain lines with up to 126 characters per line. They put information for one sample on one or two pages, formatted on the page for easier reading.

CARRIAGE CONTROL=FORTRAN. STATUS=NEW, opened by REFORM.

TOMGNAP FILE FORMAT

TOMGNAP files have 80-character lines. Chemistry data for one sample is contained on up to three lines, called analysis cards in the TOMGNAP manual. The format for TOMGNAP analysis cards is:

```
column 1                                          73      80
       ¦                                          ¦       ¦
       NRMX                    ...                   Y
```

where X is the plotting symbol for the sample
... is the chemistry data in percent with implied decimal points (as specified in the TOMGNAP command file. See the User's Manual and the GNAP manual).
Y is the sample identification, up to eight characters starting in column 73.

CARRIAGE CONTROL=LIST, STATUS=NEW, called by REFORM.

ERROR HANDLING

There are three classes of errors detected by REFORM. The first are caused by unacceptable user input and are detected during an interactive terminal session. The program's action is to display an error message and prompt the user again for input. The second are caused by card file format errors and are detected during reading of the card file or when converting strings to real numbers. The program displays an error message and tries to continue (which may cause more error messages for subsequent lines). The third class are fatal errors. The program displays an error message and stops.

A brief table of error messages follows which includes the class of error as described above (1, 2 or 3). The User's Manual contains a section on error handling and a more detailed description of error messages and the conditions for which they occur. The last error listed in the table below (error...inout variable...) is not described in the User's Manual because it results from incorrect use of the subroutine GETFILE and would be detected during program testing.

TABLE OF ERROR MESSAGES

ERROR CLASS	GENERATED WHEN RUNNING OPTION	MESSAGE (ABBREVIATED)	SUBROUTINE GENERATING MESSAGE
1	C (Tables)	invalid choice ...	GETENTRY
1	"	invalid choice (too long) ...	"
1	"	you've used XX columns ...	"
	E (TOMGNAP)		
2	All of	error trying to convert string...	RLNU
2	above	line reads ... error trying to read in line ...	BADCARD & CARDERROR
	B (Printout)		
2	All of	unrecognized line in XXXX block..	READXXXX &
	above		BADCARD
2	"	unknown datatype at line ...	REFORM
2	"	XX block not properly ended ...	READXXX
2	"	too many analysis lines ...	READISOT
2	"	" comment "	NEWCMT
2	"	" source "	NEWSOURC
2	"	" texture "	READROCK
1	A (Makecards)	answer too long ...	ASK
3	D (Shuffle)	error...array not dimensioned large enough ...	SHUFFLE
1	All of	error in opening ...	GETFILE
1	above	try again ...	REFORM
3	"	error...inout variable in sub getfile not recognized ...	GETFILE

LIST OF SUBROUTINES

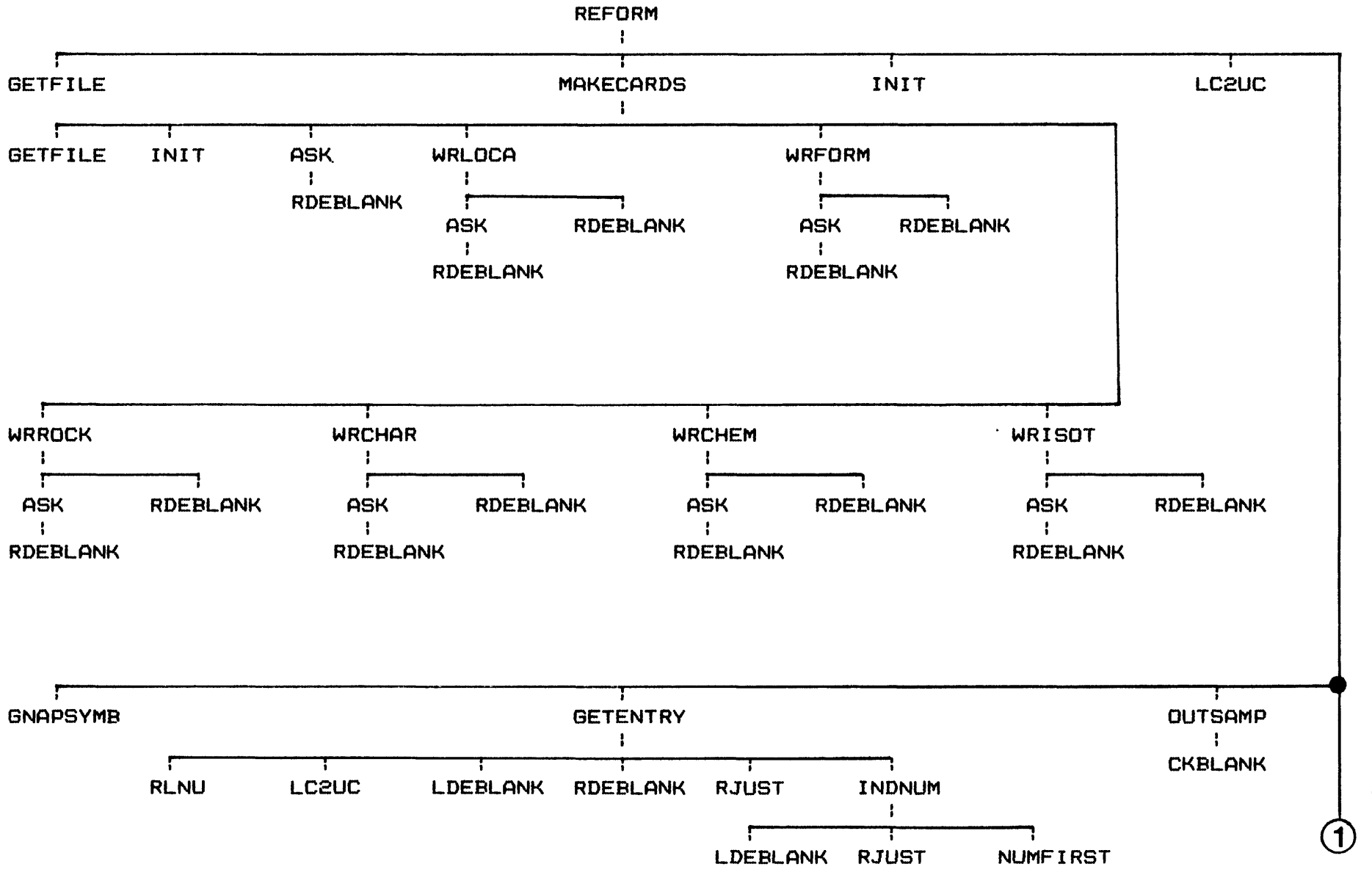
SUBROUTINE	SIZE OF SOURCE CODE FILE IN BYTES		FUNCTION
REFORM.FOR	X	4454	Main loop, displays menus.
VAR.INC	X	2175	Variable include module.
COMMON.INC	X	1329	Common include module.
BLDATA.FOR		4129	Block data.
ASK.FOR	X	944	Prompts user for data fields.
BADCARD.FOR		306	Error message, displays line with the error and the sample number.
BREAKSN.FOR	X	822	Breaks Needles-Project format sample number into components.
CARDERROR.FOR		275	Error message, displays line no.
CHRFIRST.FOR		421	Function, returns position of first alphabetic character in string.
CKBLANK.FOR		368	Counts how many entries in a character array are not blank.
FILLARRY.FOR	X	2200	Assigns data strings to character array or converts numeric strings to reals and assigns to real array.
GETENTRY.FOR	X	2567	Prompts user for fields to include in table.
GETFILE.FOR	X	1886	Opens files.
GNAPSYMB.FOR	X	1535	Prompts for plotting symbol.
INDNUM.FOR	X	1431	Converts alphabetic and numeric keys for a table to integer array.
INIT.FOR		1694	Initializes variables prior to reading each sample's data from card file.
LC2UC.FOR		409	Converts lower case letters to upper case.
LDEBLANK.FOR		530	Deletes blanks at the left end of a string.
MAKECARDS.FOR	X	9894*	Main loop of interactive data entry routine.
NEWCMT.FOR		428	Adds comment to comment array.
NEWSOURC.FOR		459	Adds source (bibliographic reference) to source array.
NUMFIRST.FOR		410	Function, finds position of first number in a string.
OUT1.FOR	X	2376	Outputs 2 cards of data from ANALYSIS 1 formated for TOMGNAP norm program.
OUT2.FOR	X	1943	As above, but from ANALYSIS 2.
OUTINAA.FOR	X	3283	Outputs 3 cards of data from ANALYSES 1 and 3 for TOMGNAP.
OUTSAMP.FOR	X	5324	Produces printout file of sample data.

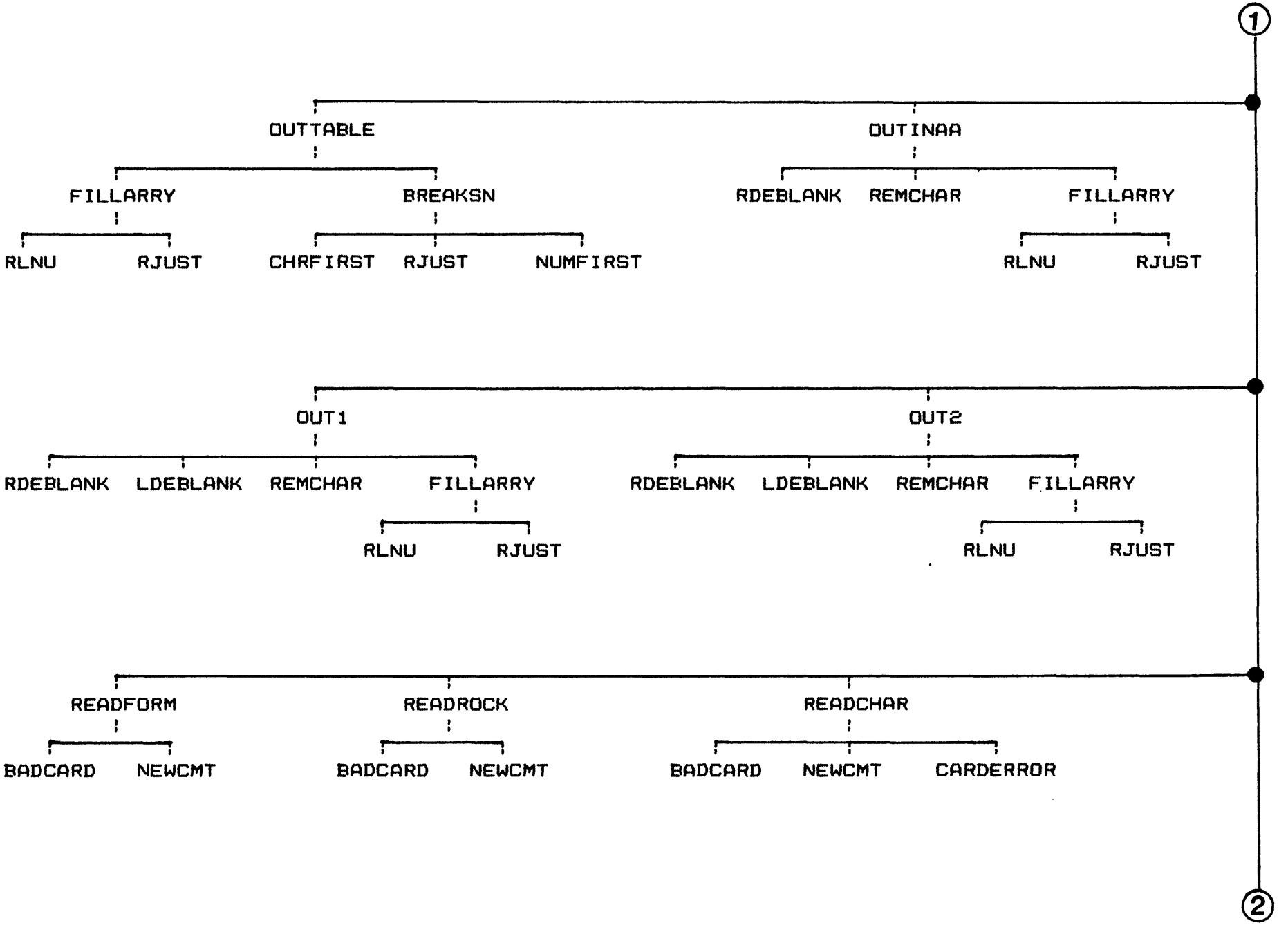
OUTTABLE.FOR	X	3096	Produces a table of selected fields of the database.
RDEBLANK.FOR		528	Deletes blanks at the right end of a string.
READCHAR.FOR	X	1728	-
READCHEM.FOR		2026	¦ These routines read cards from
READFORM.FOR		1141	¦ the blocks of a cardfile and
READISOT.FOR		1614	¦ assign the data to string
READLOCA.FOR		1464	¦ variables. E.g., READCHAR
READNIAA.FOR		1770	¦ reads data from the
READROCK.FOR		1473	¦ CHARacteristics block.
READSHAW.FOR		1595	¦
READXRF.FOR		1388	-
REMCHAR.FOR		511	Removes character from string and returns length of new string.
RJUST.FOR		485	Right justifies string in its length.
RLNU.FOR		892	Function, reads character string containing real number.
SHUFFLE.FOR	X	941	Sorts a card file according to index fields in a table.
WRLOCA		*	-
WRFORM			¦ These routines prompt user
WRROCK			¦ for data, delete trailing
WRCHAR			¦ blanks, and write it to a
WRCHEM			¦ new card file.
WRISOT			-
TOTAL		95535 bytes	

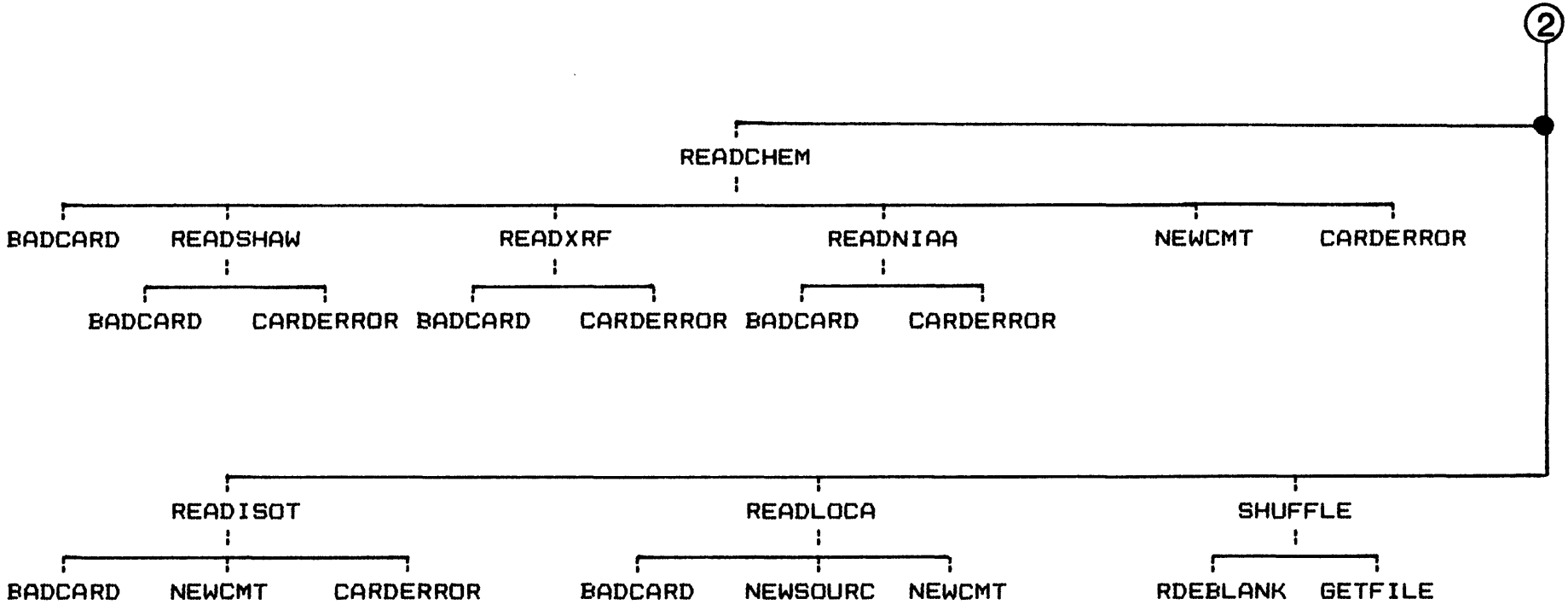
X indicates an extended description is given below.

* the size of MAKECARDS includes the write routines (WRLOCA, etc.) They are described with MAKECARDS below.

SUBROUTINE HIERARCHY




1
OUTTABLE
FILLARRY
RLNU
RJUST
BREAKSN
CHRFIRST
RJUST
NUMFIRST
OUTINAA
RDEBLANK
REMCHAR
FILLARRY
RLNU
RJUST
OUT1
RDEBLANK
LDEBLANK
REMCHAR
FILLARRY
RLNU
RJUST
OUT2
RDEBLANK
LDEBLANK
REMCHAR
FILLARRY
RLNU
RJUST
READFORM
BADCARD
NEWCMT
READROCK
BADCARD
NEWCMT
READCHAR
BADCARD
NEWCMT
CARDERROR
2


②
READCHEM
BADCARD
READSHAW
READXRF
READNIAA
NEWCMT
CARDERROR
BADCARD
CARDERROR
BADCARD
CARDERROR
BADCARD
CARDERROR
READISOT
READLOCA
SHUFFLE
BADCARD
NEWCMT
CARDERROR
BADCARD
NEWSOURC
NEWCMT
RDEBLANK
GETFILE

LIST OF SUBROUTINES AND THE SUBROUTINES THAT CALL THEM

SUBROUTINE	CALLED BY
VAR.INC, COMMON.INC	REFORM, FILLARRY, GETENTRY, GNAPSYMB, INDNUM, INIT, MAKECARDS, OUT1, OUT2, OUTINAA, OUTSAMP, OUTTABLE, READCHAR, READCHEM, READFORM, READISOT, READLOCA, READNIAA, READROCK, READSHAW, READXRF
ASK	MAKECARDS, WRLOCA, WRFORM, WRROCK, WRCHAR, WRCHEM, WRISOT
BADCARD	READCHAR, READCHEM, READFORM, READISOT, READLOCA, READNIAA, READROCK, READSHAW, READXRF
BREAKSN	OUTTABLE
CARDERROR	READCHAR, READSHAW, READXRF, READNIAA, READCHEM, READISOT
CHRFIRST	BREAKSN
CKBLANK	OUTSAMP
FILLARRY	OUT1, OUT2, OUTINAA, OUTTABLE
GETENTRY	REFORM
GETFILE	REFORM, MAKECARDS, SHUFFLE
GNAPSYMB	REFORM
INDNUM	GETENTRY
INIT	REFORM, MAKECARDS, SHUFFLE
LC2UC	REFORM, GETENTRY
LDEBLANK	GETENTRY, INDNUM, OUT1, OUT2
MAKECARDS	REFORM
NEWCMT	READLOCA, READFORM, READROCK, READCHAR, READCHEM, READISOT
NEWSOURC	READLOCA
NUMFIRST	INDNUM, BREAKSN
OUT1	REFORM
OUT2	REFORM
OUTINAA	REFORM
OUTSAMP	REFORM
OUTTABLE	REFORM

RDEBLANK	ASK, WRLOCA, WRFORM, WRROCK, WRCHAR, WRCHEM, WRISOT, GETENTRY, OUTINAA, OUT1, OUT2, SHUFFLE
READCHAR	REFORM
READCHEM	REFORM
READFORM	REFORM
READISOT	REFORM
READLOCA	REFORM
READINAA	READCHEM
READROCK	REFORM
READSHAW	READCHEM
READXRF	READCHEM
REMCHAR	OUTINAA, OUT1, OUT2
RJUST	WRCHEM, GETENTRY, INDNUM, FILLARRY, BREAKSN
RLNU	GETENTRY, FILLARRY
SHUFFLE	REFORM
WRCHAR	MAKECARDS
WRCHEM	MAKECARDS
WRFORM	MAKECARDS
WRISOT	MAKECARDS
WRLOCA	MAKECARDS
WRROCK	MAKECARDS

DESCRIPTIONS OF SELECTED SUBROUTINES

The following program descriptions are in a FORTRAN-like pseudocode.

REFORM.FOR

```
(This is the main driving routine.)

display main menu and get menu choice
if choice is "A" (make a new card file) call makecards.for
else if choice is "D" (sort a card file) call shuffle.for
else open input card file

     if choice is "B" (make a printout) open printout file
     else if choice is "C" (make a table)
          open table file
          get table entries
     else if choice is "E" (TOMGNAP file)
          display TOMGNAP menu and get choice
          open TOMGNAP file
          get plotting symbol
     else error, redisplay main menu
     endif

     do until end of input card file
          read next line of input file
               if  this is the start of a new sample record
                   (i.e.: if the first 5 characters of the
                   line = "SAMPL") and it is not the first
                   sample of the file, then reformat the
                   last sample record and write it to the
                   output file (e.g., call out1.for)
               else  call appropriate subroutine to read the
                     next block of data (e.g.: if first 5
                     characters of the line are " CHAR" call
                     readchar.for)
               endif
     enddo
endif
reformat last sample record and write it to the output file
display on terminal the number of samples read, etc.
close files
```

GETFILE

```
(Opens files.  Parameters passed to GETFILE are unit,
prompt, inout, form)

if inout=in
     status=old
     open: readonly, shared, sequential, formatted
else if inout=out
```

```
        status=new
        if form=formatted
             open: carriage control=list, sequential, formatted
        else if form=fortran
             open: carriage control=fortran, sequential,
                    formatted
        else if form=unformatted
             open: sequential, unformatted
        endif
   else error
   endif
```

READ ROUTINES: READXXXX.FOR

(where "XXXX" is LOCA, FORM, ROCK, CHAR, CHEM, XRF, SHAW, NIAA, ISOT. Each of these subroutines reads a block of data from the sample entry into the appropriate variables)

```
   do  until error or end of block (blocks are terminated with
       the statement "COMMENT=$$$", or "$$$ for chemistry
       sections)
          read next card
          assign  the data on the line to the appropriate
                  variable(s)
   enddo
```

TOMGNAP OUTPUT ROUTINES: OUT1.FOR, OUT2.FOR, OUTINAA.FOR

(These reformat chemistry data from a sample entry for TOMGNAP and write it to the output file. OUT1 reformats data from the ANALYSIS 1 chemistry section, OUT2 from ANALYSIS 2, and OUTINAA from ANALYSES 3 and part of 1)

```
   get sample identifier (part of the sample number)
   call  fillarry (this assigns the data in the field variables
         to arrays and converts strings to real numbers)
   scale  numeric values for TOMGNAP (convert ppm to %, remove
          decimal points: TOMGNAP wants implicit decimal
          points)
   write data, including plot symbol, in the proper format to
         the output file
```

GNAPSYMB

(Gets plotting symbol for TOMGNAP files. Called once by REFORM.FOR. TOMGNAP output routines write

```
   plotsymb (min(# of chem samples so far, number of plot
        symbols))
```

Thus, samples beyond number of ASCII symbols in array (84) all get last symbol in array)

ask user if wants to assign one plotting symbol for all samples
if yes, read symbol and assign to each element of plot symbol array
else assign a unique plot symbol (ASCII characters A through @) to each element of plot symbol array

FILLARRY.FOR

(Called by the output routines, once for each sample)

Assign alphabetic-keyed variables to string array according to their keys.
get shawno, if present, for sample identification
convert lat/lon to decimal degrees
convert numeric-keyed variables to real numbers and assign them to real array according to their keys

OUTSAMP

Writes data for sample entries to a printer file, one or two samples per page, 126 columns per line. Called by REFORM.FOR, once per sample, after each sample entry's data is read from the card file.

Does no error-checking: writes fields by column position on the line so that despite invalid data or a shifted field the contents in the field's normal columns will be written.

Writes LOCATION, FORMATION, ROCK, and CHARACTERISTICS blocks even if no data in them (writes fieldnames with blanks for data). Writes ANALYSIS 1, 2, or 3, RATIOS, or ISOTOPES only if data is present.

Writes only those COMMENT, SOURCE, TEXTURE, and ISOTOPE ANALYSIS lines that are present.

MAKECARDS AND WRITE ROUTINES

(Prompts user for data and creates a new card file. Inserts field names and positions data in proper columns)

calls write subroutines for each data block

these subroutines (WRLOCA, etc.) call ASK.FOR which prompts for the field, checks that it is not too long,

returns to the end of the calling write subroutine if escape character is typed. The write routines then write the data minus trailing blanks to the card file.

SHUFFLE

(Sorts a card file according to a table. The table must contain index fields: the starting line number in the card file of the sample entry, and the number of lines in the sample entry)

```
open index (table) file, input card file, and output card
     file
read input card file into string array of 80-character
     elements
do until end of table file
     read indexes for next sample entry from table
     delete trailing blanks and write lines from start to
          end of sample entry per its indexes
```

TABLE ROUTINES

The subroutines OUTTABLE.FOR, GETENTRY.FOR, INDNUM.FOR, and BREAKSN combine to produce a table of fields, with data for each sample on one line. The routines are relatively complicated because the user may specify any fields, the formats for each are different, and because most fields are truncated to enable more fields to fit on a line. For instance, some character fields are 40 characters long but commonly only the first few characters are used. Thus FORMATION NAME, for example, is 40 characters in card files but truncated to 20 characters in tables.

REFORM calls GETENTRY once, which calls INDNUM once per choice after it is specified by the user. OUTTABLE is called by REFORM once per sample, and calls BREAKSN each time if expanded sample numbers are specified.

The following table shows the data structures used by the table routines. Data flow is generally from top to bottom in the diagram. The examples show data for a table with the keys B, 102, 19, and D chosen.

DATA STRUCTURES USED BY TABLE ROUTINES

VARIABLE	TYPE	EXAMPLE	
GETENTRY.FOR	(gets user's choices of fields to include in table)		
expand	char*1	N	(flag: break sample number into components?)
uflag	char*1	Y	(flag: include index fields in table?)
tchoice	char*5	D	(present choice of field key)
nchoices	int	4	(number of choices so far)
entry(nchoices)	char*3(30)	[B 102 19 D]	(choices so far)
tablelen(nchoices)	int(150)	[17 11 ... 11 ... 6 ... 9 ...]	(--integer keys as converted by INDNUM)

The tablelen block-data array holds the lengths of fields in their proper format for tables.

ncol	int	37	(nbr of columns in table used so far)
INDNUM.FOR	(converts alphabetic and numeric keys to integers 1 through 135)		
tablenum(nchoices)	int(150)	[2 127 44 4]	(keys A-Y =) 1-25, 1-110 =) 26-135)

Tablenum holds the converted keys for the fields chosen so far.

OUTTABLE.FOR	(writes data for one sample (one line) to table file)		
rcateg(k)	int(30)	[127 44]	(real categories (keys) from tablenum)
ccateg(j)	int(30)	[2 4]	(character categories (keys) from tablenum)

(j+k=nchoices)

```
                                                2              4        (--converted keys
id(ccateg(i),i=1,j) char*40(100)    [ Old Woman Mts  30T11NR30E ]

                                          44  127                       (--converted keys
numlist(rcateg(i),i=1,k) real(100) [ 107  1.0 ]
```

The arrays id and numlist hold data from fields with alphabetic keys and numeric keys respectively. These assignments are made by FILLARRY.FOR.

```
WRITE(FORCHRx)(index fields, id(ccateg(i)i=1,j), numlist(rcateg(i),i=1,k))
```

This is the write statement for one line (sample) in a table.
It uses the format in array FORCHRx.

```
                                                                          converted
                                          2          4          44          127    (--keys
tablefmt( )          char*15(150)   [... A10,1X, ... A10,1X, ... F5.1,1X, ... F8.4,1X, ...]
```

The above block data array holds the special table formats for each field (key).

```
                                                          2        4        127       44
FORCHRx              char*9(150)    [(I5,  I5,2X,  A16,1X,  A10,1X,  A10,1X,  F8.4,1X,  F5.1,1X,]
      |                             |_____________________|  |_____________|  |_____________|
      |                              formats for            formats for       formats for
      |                              indexes & sample #     alphabetic keys   numeric keys
      |
where x is   expand    uflag
```

1	y	y	
2	y	n	The format statement can have one of four forms depending
*3	n	y	on the state of the expand and uflag variables.
4	n	n	

GETENTRY

(Gets user's choices of whether to expand sample numbers and include index fields in the table, and of fields (by keys) to include in the table)

```
prompt user if wants sample numbers expanded
prompt user if wants index fields in table
display menu of choices
prompt user to enter choice
do
  if choice= "XX" stop
  else if choice= "DD" then return
  else if choice= "SS" then start over building table
  else if valid field key
     put choice in array of choices so far
     call INDNUM (convert alphabetic and numeric keys to
          sequence of integers)
     sum number of columns in table used so far
     error if more than 126 columns (start over)
  else invalid choice, prompt user again for choice
```

INDNUM

(Converts alphabetic and numeric keys to sequence of integers. Called by GETENTRY.FOR, once for each choice.)

```
if key is numeric convert string number to integer + 25
else convert alphabetic key to integer (A=1, Y=25)
endif
```

OUTTABLE

(Writes fields for a sample entry in their table format to table file. Called by REFORM.FOR, once for each sample entry.)

```
call fillarry (fill data arrays with current sample entry
     data)
if expand flag=yes call breaksn (expand sample nbrs into
     components)
separate table choices into array of numeric keys (rcateg)
     and array of alphabetic keys (ccateg)
assemble proper format statement into a string array,
     depending on state of expand and index flags:

     first, formats for index fields (if any) and sample
          number (expanded if specified) are assigned to the
          array
     then, formats for the alphabetic-keyed fields
     finally, formats for the numeric-keyed fields
```

write using format statement just created:

index fields (if any)
sample number (expanded if specified)
fields with alphabetic keys: (id(ccateg(i)), i=1,j)
fields with numeric keys: (numlist(rcateg(i)), i=1,k)

BREAKSN

(Breaks (=expands) Needles-Project format sample number into component parts to facilitate sorting)

For example, the sample number BJ87CAL-238AX becomes

```
BJ   87   CAL  238  AX

2    2    3    3    2     <- size  12 + 4 spaces=16 cols

geol year range  sn   sna   <- fields
```

MODIFYING REFORM

The REFORM software may be modified with relative ease. The table output subroutines are somewhat obscure and are dealt with in some detail in the subroutine descriptions above.

Modifications are of four types: (1) changing database structure, (2) adding new output routines, (3) altering the software to run on another system, and (4) more complicated changes. The latter might include, for instance, boolean search and selection options, and are not considered here since they would require significant additions.

After making changes to REFORM be sure to test the new version extensively. Tests should include extreme values, improperly formatted data (to check error-trapping), data for all new fields and for all new options. Add new error-trapping if needed. Finally, update the REFORM documentation.

(1) CHANGING DATABASE STRUCTURE

The programming complexity involved in changing the structure of the database depends on the types of modifications required. The changes to specific subroutines are outlined below for several types of modifications.

Adding more comment, source, texture, and analysis lines:

The maximum allowable number of such lines is stored as the parameters ncmt, nsrc, ntext, and nanal, respectively, in the variable file VAR.INC. Change the appropriate parameter to the new number of lines required. Note that changing ncmt changes the number of comment lines allowed for every data block.

Adding new fields to present data blocks:

Add the new variables to VAR.INC and COMMON.INC.

If the new variables are part of an existing chemistry analysis section or are part of the CHARACTERISTICS section and are best held in array form, add the field names to the appropriate block data arrays in BLDATA.FOR, and change the appropriate parameter in VAR.INC that gives the number of fields in the section (e.g., nshaw for ANALYSIS 1 section). Note that not all CHARACTERISTICS variables are included in block data, only the ones which are most easily handled as part of an array, such as mode data.

Assign keys to the new variables, and place the desired table format for the fields in the TABLEFMT array, and their lengths in the TABLELEN array, both in BLDATA.FOR.

Add the new variables and their keys to FILLARRY.FOR.

If any new character fields (alphabetic-keyed) are longer than 40 characters, increase the size of the id array size in VAR.INC. Change AMENUMAX and NMENUMAX in VAR.INC. These specify the largest numeric and alphabetic keys. Change INIT.FOR to

initialize the new variables.

Modify the read subroutines (e.g., READLOCA) for the data blocks with new variables to read those new variables from the card file.

Modify the write subroutines (e.g., WRCHAR) to prompt for the new variables and write them to new card files.

Modify the output subroutines to write the new variables to the output files. OUTSAMP.FOR must be modified for any new variables. OUT1.FOR, OUT2.FOR, or OUTINAA.FOR must be changed only if the new variables are chemistry data.

Adding a new data block:

For each of the new fields in the new data block make the changes listed above.

Write a new read subroutine to read in the fields in the new block. Model it after one of the existing read subroutines.

Change REFORM.FOR to call the new read subroutine. Note the read subroutines are called twice by REFORM.FOR.

Write a new write subroutine to prompt the user for the fields in the new data block and write them to new card files.

Change MAKECARDS.FOR to call the new write subroutine.

In addition to any other changes, if the total number of variables with alphabetic keys is greater than 25:

Increase the position of the first format and length for numeric-keyed fields in TABLEFMT and TABLELEN, respectively, both in BLDATA.FOR, from 25 to the new number of alphabetic keys (or larger).

Increase the size of arrays TABLEFMT, TABLELEN, ID, NUMLIST in VAR.INC if necessary.

Change "+25" in INDNUM.FOR to the new number of alphabetic characters.

Change "25" in OUTTABLE.FOR associated with the arrays RCATEG and CCATEG to the new number of alphabetic keys.

If the number of alphabetic keys is more than 26, change FILLARRY.FOR and its IND(AKEY) function to handle double alphabetic keys (e.g., AA). Note that DD,SS and XX are used by GETENTRY.FOR as commands. They must be changed, for instance to some word. Also, INDNUM1(TKEY) in INDNUM.FOR must be changed to handle double alphabetic keys.

If the total number of numeric keys is greater than 150, increase the sizes of arrays NUMLIST, TABLEFMT, and TABLELEN.

NEW OUTPUT ROUTINES

In general, output routines will simply write data to a file in the desired format, possibly following scaling or conversion of some kind. The present routines, OUT1, OUT2, OUTINAA, and OUTTABLE call FILLARRY which converts numeric strings to real numbers and assigns variables to string or real arrays to simplify write statements. The output routines then scale the data as required for the particular application.

New output routines may be modeled on existing ones:

OUTSAMP.FOR for text printouts
OUTTABLE.FOR for tabular data
OUT1.FOR, OUT2.FOR, and OUTINAA.FOR for numeric (chemistry) data.

See the descriptions of these subroutines in this manual. Binary (non-ASCII) files may be written by opening an unformatted rather than formatted file (see GETFILE).

Output routines are called each time data for one whole sample in the card file has been read. REFORM.FOR must be modified to call the new output routine in two places. Modify the REFORM menu to list the new option.

For many purposes data may be output in table form with REFORM option C and used as input by another program. In such cases a new output routine will not be needed. The fields in the tables are separated by spaces, not commas, and strings are not enclosed in quotation marks. This may pose a problem for some programs. A new output routine similar to OUTTABLE could be made which adds these delimiters.

PORTING REFORM TO OTHER SYSTEMS

The main considerations in installing REFORM on a system other than VAX/VMS are the VAX-11 FORTRAN extensions to the FORTRAN-77 standard that are used in REFORM (see list below), the sizes of data structures, and the sizes of program files.

Data structures are not large with the exception of the array CARD used in SHUFFLE.FOR. Its size is 20,000 80-character elements. It is used to hold an entire card file in main memory during sorting. If your system cannot support an array that large, a new memory-efficient scheme for sorting a card file may be needed.

Individual subroutines are not large but on a small system such as a PC not all may fit in memory at one time. It may be necessary to separate the various options into separate stand-alone programs that are called into memory as required.

VAX FORTRAN EXTENSIONS USED IN REFORM

OPEN (...SHARED), used by GETFILE, allows programs to simultaneously access the file. If not supported then delete.

OPEN (...READONLY), used by GETFILE, write-protects files. If not supported then delete.

OPEN (CARRIAGE CONTROL=LIST), OPEN (CARRIAGE CONTROL= FORTRAN), used by GETFILE, specify carriage control, if not supported delete the parameter, and modify write statements in files opened with CARRIAGE CONTROL=LIST so that lines are single spaced (the first character of each line should be a space: ' ').

OPEN defaults to ACCESS=SEQUENTIAL. If this is not the system default, specify ACCESS=SEQUENTIAL.

"$" format descriptor suppresses carriage return at end of line for prompts, used by REFORM, ASK, GETENTRY, GETFILE, and GNAPSYMB. If not supported then delete.

Symbolic names longer than six characters. If not supported replace by shorter names, being careful to change the names everywhere. The following routines use long names:

VAR.INC COMMON.INC ASK GETENTRY GETFILE
INIT LDEBLANK OUT1 OUT2 OUTTABLE
RDEBLANK READLOCA READROCK REMCHAR SHUFFLE

INCLUDE/NOLIST. The NOLIST option keeps the contents of the include file from being listed in the compilation source listing, and is used in REFORM wherever INCLUDE is used (all subroutines except the short utility routines). Delete if not supported.

COMPILING ON THE VMS/VAX

The source code for each of the REFORM subroutines is in a separate file. This allows each subroutine to be compiled individually to simplify debugging. The object files are placed in an object library which is then linked to create the executable module.

To compile new subroutines:

FOR <subroutine name> for each subroutine. This creates a compiled object file for each.

Then:

LIBR/CREATE ROCK.OLB	creates roch object library
LIBR/REPLACE ROCK *.OBJ	places the objects in the rock object library
DELETE *.OBJ.*	delete object files
FOR REFORM	compile the main loop
LINK REFORM ROCK/L/INCLUDE=BLDATA$DATA	links modules
DELETE REFORM.OBJ.*	delete object file

```
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c      reform.for
c      Reads standard card format input and outputs in new format.

       include 'var.inc/nolist'
       include 'common.inc/nolist'
       common/cardcount/ lastsamp

       character choice*1
       ncards=0
       nsamps=0
       shawtot=0
       lastsamp=1
c       xrftot=0
       do 10 i=1,80
 10    blankcard(i:i)=' '

       print*,' '
       print*,'This is REFORM version 2.1, a database program'
       print*,' for rock sample data.'

20     print*,' '
       print*,'These are your choices:'
       print*,' '
       print*,'     A  Make a new card file'
       print*,'     B  Make a printout of a card file'
       print*,'     C  Make a table'
       print*,'     D  Sort a card file'
       print*,'     E  Reformat chem data for TOMGNAP'
       print*,' '
       print*,'     X  (exit REFORM)'
       print*,' '
       print '(/,1h$,a)','*Your choice? '
       read '(a)',choice
       call lc2uc (choice)
       if (choice.eq.'A') then
        call makecards
        stop
       else if (choice.eq.'D') then
        call shuffle
        stop
       else if (choice.eq.'X') then
        stop
       else
        call getfile(21,'*Give card format infile: ','in','formatted')
        tmpfilname=filename

        if (choice.eq.'B') then
        call getfile(22,'*Give listing outfile:    ','out','fortran')


        else if (choice.eq.'C') then
        call getfile(28,'*Give table outfile:      ','out','formatted')
        call getentry

        else if (choice.eq.'E') then
```

```
      print*,' '
30    print*,'This is the Reform option E menu.'
      print*,' '
      print*,'The following routines reformat data
     & for TOMGNAP.'
      print*,' '
      print*,'You may choose to reformat data from:'
      print*,' '
      print*,'     1  Analysis 1'
      print*,'     2  Analysis 2'
      print*,'     3  Analyses 3 and 1 combined'
      print*,' '
      print*,'     X  (exit REFORM)'
      print*,' '
      print '(/,1h$,a)','*Your choice? (1,2,3,or X): '
      read '(a)',choice
      if (choice.eq.'x'.or.choice.eq.'X') then
        close(21)
        stop
       else if (choice.eq.'1'.or.choice.eq.'2'.or.choice.eq.'3') then
      call getfile(30,'*Give tomgnap outfile:    ','out','formatted')
      call gnapsymb
       else
            print*,'Try again'
            print*,' '
            goto 30
      endif

      else
         print*,'Try again'
         print*,' '
         goto 20
      endif
     endif

 100  read(21,'(a80)',end=900) card
     ncards=ncards+1
     datatype=card(1:5)

     if(card.eq.blankcard) then
       continue
     else if(datatype.eq.'SAMPL') then
c      New sample...output previous sample data...
       if(nsamps.gt.0) then
         if (choice.eq.'B')call outsamp
         if (choice.eq.'C')call outtable
         if (choice.eq.'1')call out1  (symdex)
         if (choice.eq.'2') call out2 (symdex)
         if (choice.eq.'3') call outinaa (symdex)
       endif
       call init
       sample=card(11:30)
       nsamps=nsamps+1
     else if(datatype.eq.' LOCA') then
       call readloca
     else if(datatype.eq.' FORM') then
```

```
        call readform
      else if(datatype.eq.' ROCK') then
        call readrock
      else if(datatype.eq.' CHAR') then
        call readchar
      else if(datatype.eq.' CHEM') then
        call readchem
      else if(datatype.eq.' ISOT') then
        call readisot
      else
        print *, ' Unknown datatype at line',ncards
        print *, '  after sample number... ',sample
        print *, ' line reads:'
        print *,   card
        print *, ' Trying to continue...'
      endif

      goto 100

c     Output last sample if not already done...
 900  if(sample.ne.'    ') then
          if (choice.eq.'B')call outsamp
          if (choice.eq.'C')call outtable
          if (choice.eq.'1')call out1    (symdex)
          if (choice.eq.'2')call out2 (symdex)
          if (choice.eq.'3')call outinaa (symdex)
       endif
       if (choice.eq.'1'.or.choice.eq.'2'.or.choice.eq.'3')
     &   print*, 'Number of analysis samples read = ',shawtot


      print*,'Number of cards read = ',ncards
      print*,'Number of samples read = ',nsamps

      close(21)
      close(22)
      close(28)
      close(30)
      stop
      end
```

```
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c      var.inc - VAX include module

c Variables...

c    parameters:
       integer ncmt, ntext, nch, nshaw, nxrf, nniaa, nrat, nanal,nsymbs
      &     nsrc
       parameter (ncmt=5, ntext=4, nch=19, nshaw=30, nxrf=17,
      &             nniaa=29, nrat=6, nanal=6, nsymbs=84, nsrc=10)
       character amenumax*5
        real nmenumax
        parameter (nmenumax=110.)
        parameter (amenumax='T')

c    variables assoc. w/ output routines, eg: outgnap, fillarry, etc.
       dimension plotsymb (nsymbs)
       character akey*1, id(100)*40
       integer nkey(6)
       real numlist(100), xlon, xlat, data(6)

c    general variables:
       character card*80, blankcard*80, datatype*5, symflag*1, plotsymb*1
       character sample1*20, xrfvar*3, filename*50, shawno*3
       integer symdex, ncards, nsamps, shawtot, xrftot

c    sample & category:
       character sample*20, category(10)*4

c    location:
       character range*20, quad*30, trs*10
       character lat*6, lon*7
       character*50 locacmt(ncmt), source(nsrc)

c    formation:
       character formsymb*20, formname*40
       character*50 formcmt(ncmt)

c    rock types:
       character rocktype*40, grainsize*40, texture(ntext)*50
       character*50 rockcmt(ncmt)

c    characteristics:
       character cps*12, sg*5, ci*5, mag*7
       character*5 charval(nch)
       character*5 charname(nch)
       character othermin*50
       character*50 charcmt(ncmt)

c    chemistry:
       character*7 shawval(nshaw), xrfval(nxrf)
       character*10 ratio(nrat), niaaval(nniaa)
       character*5 shawname(nshaw), xrfname(nxrf), rationame(nrat)
       character*50 chemcmt(ncmt), analyst1, analyst2, analyst3
       character niaaname(nniaa)*2
c    isotopes:
```

```
      dimension alysis(nanal), alyst(nanal), mineral(nanal),
     &     age(nanal), isotratio(nanal)
      character alysis*5, alyst*10, mineral*5, age*20, isotratio*25
      character*50 isotcmt(ncmt)

c     table variables:
      character expand*1, entry(30)*3, tablefmt(150)*15, tmpfilname*13
      character uflag*1
      integer tablelen(150), tablenum(150), nchoices
```

```
Ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c       common.inc - VAX include module


c    general variables:
       common/cardnum/ ncards, nsamps, plotsymb, shawtot, xrftot
       common/cardchar/ card, blankcard, sample1, xrfvar, filename,
      &                 shawno
c    sample & category:
       common/sample/ sample, category

c    location:
       common/locachar/ range, quad, trs, locacmt, source
       common/locanum/ lat, lon

c    formation:
       common/formchar/ formsymb, formname, formcmt

c    rock types:
       common/rockchar/ rocktype, grainsize, texture, rockcmt

c    characteristics:
       common/charnum/ cps, sg, ci, mag, charval
       common/charchar/ charname, othermin, charcmt

c    chemistry:
       common/chemnum/ shawval, xrfval, niaaval, ratio
       common/chemchar/ shawname, xrfname, niaaname, rationame,
      &    chemcmt, analyst1, analyst2, analyst3

c    isotopes:
       common/isotchar/ alysis, alyst, mineral, age, isotratio,
      &    isotcmt

c    table variables:
       common/tablchar/ expand, entry, tmpfilname, tablefmt, uflag
       common/tablint/  tablelen, tablenum, nchoices
c    variables assoc. w/ output routines
       common/outchar/ akey, id
       common/outint/ nkey
       common/outreal/ numlist, xlon, xlat, data
```

```
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      subroutine ask(question,answer,*)
c      Asks for variable...checks to see if answer is the escape
c      character... if it is, get alternate return to *.

      character*(*) question, answer
      character string*80
      character underline*50
      character escape*1
      parameter (escape='\')
      string=' '
      underline='__________________________________________________'

      lans=len(answer)

 10   print '(1h ,5x,a,1h ,a)', question, underline(1:lans)
      print '(1h+,5x,a,1h ,$)', question
      read '(a)', string
      if(string.eq.escape) return1

      call rdeblank(string,string,lstring)
      if(lstring.gt.lans) then
        print *, ' ANSWER TOO LONG...BY',lstring-lans
        print *, '     TRY AGAIN!!!!!!'
        go to 10
      else
        answer=string
      endif

      return
      end




ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      subroutine badcard (kard,sampl)
c     Sends error message of card and sample number

      character kard*80, sampl*20

      print*, ' Line reads ', kard
      print*, ' Sampl number is ', sampl

      return
      end
```

```
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      block data

      include 'var.inc/nolist'
      include 'common.inc/nolist'

      data charname/'   N','   Q','   P','   A','   M',
     &              '   N','  QPA','   B','   H','   M','  SP','  MG',
     &              '  Q1','  P1','  A1','  Q2','  P2','  A2','  M2'/

      data shawname/'SIO2','TIO2','AL2O3','FE2O3','FEO',
     &              'MNO','MGO','CAO','NA2O','K2O',
     &              'P2O5','H2O+','H2O-','CO2','TOTAL',
     &              'Y','SR','ZR','U','RB',
     &              'TH','PB','GA','ZN','CU',
     &              'NI','CR','V','BA','NB'/

      data xrfname/'SIO2','TIO2','AL2O3','FE2O3','MNO',
     &             'MGO','CAO','NA2O','K2O','P2O5',
     &             'Y','SR','ZR','RB','BA',
     &             'NB','FEO'/

      data niaaname/'FE','K','NA','BA','CO','CR','CS','HF','MN','RB',
     &              'SB','SR','TA','TH','U','ZR','SC',
     &              'LA','CE','ND','SM','EU',
     &              'GD','TB','DY','TM',
     &              'YB','LU','HO'/

      data rationame/'BA/RB','BA/SR','RB/SR',
     &               'U/TH','K/RB','K/BA'/

      data (tablefmt(i),i=1,30)/
     &              'A16,1X,','A10,1X,','A15,1X,','A10,1X,','A15,1X,',
     &              'A20,1X,','A20,1X,','A20,1X,','A 8,1X,','A20,1X,',
     &              'A 5,1X,','A10,1X,','A 5,1X,','A 5,1X,','A10,1X,',
     &              'A 5,1X,','A20,1X,','A20,1X,','A20,1X,','A 3,1X,',
     &              '       ,','   ,','       ,','       ,','       ,',
     &         'F7.4,1X,','F8.4,1X,','F6.4,1X,','F4.1,1X,','F7.4,1X,'/

      data (tablefmt(i),i=31,60)/
     &         'F5.0,1X,','F5.0,1X,','F5.0,1X,','F5.0,1X,','F5.0,1X,',
     &         'F5.0,1X,','F5.0,1X,','F5.0,1X,','F5.0,1X,','F5.0,1X,',
     &         'F5.0,1X,','F5.0,1X,','F5.1,1X,','F5.1,1X,','F5.1,1X,',
     &         'F5.1,1X,','F5.1,1X,','F5.1,1X,','F5.1,1X,','F5.2,1X,',
     &         'F4.2,1X,','F5.2,1X,','F5.2,1X,','F5.2,1X,','F3.2,1X,',
     &         'F5.2,1X,','F5.2,1X,','F5.2,1X,','F5.2,1X,','F4.2,1X,'/
      data (tablefmt(i),i=61,90)/
     &         'F5.2,1X,','F5.2,1X,','F5.2,1X,','F6.2,1X,','F5.0,1X,',
     &         'F5.0,1X,','F5.0,1X,','F5.0,1X,','F5.0,1X,','F5.0,1X,',
     &         'F5.0,1X,','F5.0,1X,','F5.0,1X,','F5.0,1X,','F5.0,1X,',
     &         'F5.0,1X,','F5.0,1X,','F5.0,1X,','F5.0,1X,','F5.2,1X,',
     &         'F5.2,1X,','F5.2,1X,','F5.2,1X,','F5.3,1X,','F5.2,1X,',
     &         'F5.2,1X,','F5.2,1X,','F5.2,1X,','F5.2,1X,','F5.0,1X,'/
      data (tablefmt(i),i=91,125)/
     &         'F5.0,1X,','F5.0,1X,','F6.1,1X,','F5.0,1X,','F5.0,1X,',
     &         'F5.2,1X,','F6.3,1X,','F6.3,1X,','F6.3,1X,','F9.4,1X',
```

```
&          'F9.4,1X,','F9.4,1X,','F9.4,1X,','F9.4,1X,','F9.4,1X,',
&          'F9.4,1X,','F9.4,1X,','F9.4,1X,','F9.4,1X,','F9.4,1X,',
&          'F9.4,1X,','F9.4,1X,','F9.4,1X,','F9.4,1X,','F9.4,1X,',
&          'F9.4,1X,','F9.4,1X,','F9.4,1X,','F9.4,1X,','F9.4,1X,',
&          'F9.4,1X,','F9.4,1X,','F9.4,1X,','F9.4,1X,','F9.4,1X,'/
 data (tablefmt(i), i=126,150)/
&          'F8.4,1X,',
&          'F8.4,1X,','F8.4,1X,','F8.4,1X,','F9.3,1X,','F9.3,1X,',
&          'F7.2,1X,','F8.4,1X,','F7.2,1X,','F8.4,1X,','          ',
&          '          ','          ','          ','          ','          ',
&          '          ','          ','          ','          ','          ',
&          '          ','          ','          ','          '/

 data tablelen/17,11,16,11,16,21,21,21,09,21,06,11,06,06,11,05,
&              21,21,21,04,12,0,0,0,0,
&              8,9,7,5,8,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,
&              6,6,6,6,6,5,6,6,6,4,6,6,6,6,5,6,6,6,7,6,
&              6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,
&              6,6,6,6,6,6,6,7,6,6,6,7,7,7,10,10,10,10,10,10,
&              10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,
&              10,10,10,10,10,9,9,9,9,10,10,8,9,8,9,0,0,0,0,0,0,
&              0,0,0,0,0,0,0,0,0/
 end
```

```
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      subroutine breaksn(samp,geol,year,rang,sn,sna)
c     Breaks a sample number into...
c         geologist
c         year
c         range
c         number (numeric part)
c         number (alphabetic part, if any)

      character*(*) samp,geol,year,rang,sn,sna

      l=len(samp)
      i1=numfirst(samp)
      geol=samp(1:i1-1)
      year=samp(i1:i1+1)

      i2=numfirst(samp(i1+2:l))+i1+1
      i3=chrfirst(samp(i2:l))+i2-1
      if(samp(i2-1:i2-1).eq.'-') samp(i2-1:i2-1)=' '
      rang=samp(i1+2:i2-1)
      sn=samp(i2:i3-1)
      sna=' '
      if(chrfirst(samp(i2:l)).ne.0) sna=samp(i3:l)

c     Right-justify the fields

      call rjust (geol)
      call rjust (year)
      call rjust (rang)
      call rjust (sn)
      call rjust (sna)

      return
      end




ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      subroutine carderror(nkards)
c     Sends error message if error in card format.

      integer nkards

      print *, ' Error trying to read in line', nkards

      return
      end
```

```
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      function chrfirst(string)
c     Finds the position of the first alphabetic character in a string.
      character string*(*)

      l=len(string)
      chrfirst=0

      do 20 i=1,l
      if(string(i:i).ge.'A'.and.string(i:i).le.'Z') then
        chrfirst=i
        return
      endif
20    continue

      return
      end
```

```
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      subroutine ckblank(array,n,notblank)
c     Counts how many entries in a character array are not blank.
      character blank*80, array(n)*(*)

      blank( : )=' '
      notblank=0

      do 10 i=1,n
 10   if(array(i).ne.blank) notblank=notblank+1

      return
      end
```

```
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      subroutine fillarry

c     Fills id array with character data from present record,
c     converts lat/long to decimal degrees, and fills numlist
c     array with real data.


      include 'var.inc/nolist'
      include 'common.inc/nolist'
c      common/plotf/ ckey, nkey
      common/index/ i


c... functions...
      ind(akey)=ichar(akey)-ichar('A') + 1
      decdeg(xdeg,xmin,xsec)=
     &      xdeg+sign(xmin,xdeg)/60.+sign(xsec,xdeg)/3600.

      id(ind('A'))=sample
      id(ind('B'))=range
      id(ind('C'))=quad
      id(ind('D'))=trs
      id(ind('E'))=formsymb
      id(ind('F'))=formname
      id(ind('G'))=rocktype
      id(ind('H'))=grainsize
      id(ind('I'))=cps
      id(ind('J'))=othermin
      id(ind('K'))=alysis(1)
      id(ind('L'))=alyst(1)
      id(ind('M'))=mineral(1)
      id(ind('N'))=alysis(2)
      id(ind('O'))=alyst(2)
      id(ind('P'))=mineral(2)
      id(ind('Q'))=analyst1
      id(ind('R'))=analyst2
      id(ind('S'))=analyst3

      shawno=' '
      do 1500 j=1,5
         if (locacmt(j)(1:6).eq.'SHAWNO') then
            nidx=index(locacmt(j)(8:12),'S') +7
            shawno=locacmt(j)(8:nidx-2)
            call rjust (shawno)
         endif
1500  continue
      id(ind('T'))=shawno

      xlond=rlnu(lon(1:3))
      xlonm=rlnu(lon(4:5))
      xlons=rlnu(lon(6:7))
      xlon=decdeg(xlond,xlonm,xlons)
      xlatd=rlnu(lat(1:2))
      xlatm=rlnu(lat(3:4))
      xlats=rlnu(lat(5:6))
```

```
      xlat=decdeg(xlatd,xlatm,xlats)


      numlist(1)=xlat
      numlist(2)=xlon
      numlist(3)=rlnu(sg)
      numlist(4)=rlnu(ci)
      numlist(5)=rlnu(mag)
      do 110 i=6,24
110   numlist(i)=rlnu(charval(i-5))
      do 120 i=25,54
120   numlist(i)=rlnu(shawval(i-24))
      do 130 i=55,71
130   numlist(i)=rlnu(xrfval(i-54))
      do 140 i=72,100
140   numlist(i)=rlnu(niaaval(i-71))
      do 150 i=101,106
150   numlist(i)=rlnu(ratio(i-100))
      numlist(107)=rlnu(age(1))
      numlist(108)=rlnu(isotratio(1))
      numlist(109)=rlnu(age(2))
      numlist(110)=rlnu(isotratio(2))

      return

      end
```

```
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      subroutine getentry
c       Prompts user for which fields are to make up the table,
c       any combination up to 126 columns wide (except that the
c       sample number is automatically included as a choice)

      include 'var.inc/nolist'
      include 'common.inc/nolist'
      character tchoice*5

10    print*,' '
      print'(/,1h$,a)','*Expand (Needles format) sample numbers
     & (Y/N)? '
      read '(a)',expand
      call lc2uc (expand)

      print*,' '
      print'(/,1h$,a)','*Index fields (Y/N)? '
      read '(a)', uflag
      call lc2uc (uflag)

      print*,' '
      print*,'Enter a key or one of the following command keys:'
      print*,' '

      nchoices=0

100   print*,' '
      print*,'          SS   start over building table '
      print*,'          DD   done building table, start processing'
      print*,'          XX   (exit to VAX system) '
      print*,' '

      print'(/,1h$,a)','Choice? '
      read '(a)',tchoice
      call lc2uc (tchoice)

       call ldeblank(tchoice,tchoice,lnth)
       call rdeblank(tchoice,tchoice,lngth)
       if (lngth.gt.3) then
         print*,'Invalid choice (too long). Try again please.'
         goto 100
       endif
       if (tchoice.eq.'XX') then
         stop
       else if (tchoice.eq.'DD') then
         goto 200
       else if (tchoice.eq.'SS') then
         print*,'Start over now.'
        goto 10
       else if((lngth.eq.1).and.(tchoice.ge.'A'.and.tchoice.le.
     &   amenumax)) then
           goto 120
       else if (tchoice(1:1).lt.':') then
         if  (rlnu(tchoice).ge. 1.0 .and.rlnu(tchoice)
     &   .le. nmenumax) then
```

```
            goto 120
          else
            goto 170
          endif
120       nchoices=nchoices+1
          entry(nchoices)=tchoice
          call rjust(entry(nchoices))
          print*,'Your choices so far are: ',(entry(i),i=1,nchoices)
          call indnum(nchoices)
          ncols=17
          do 150 i=1,nchoices
             ncols=ncols+tablelen(tablenum(i))
150          continue
          if (uflag.eq.'Y') ncols=ncols+12
          if (ncolms.gt.126) then
             print*,'You''ve used ',ncols,' columns which is
     &          more than 126; please start over'
             goto 10
          else
             print*,'Columns used up so far = ', ncols
          endif
        else
170       print*,'Invalid choice, try again please.'
        endif
        goto 100

200     return
        end
```

```
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      subroutine getfile(unit,question,inout,form)
c      Opens files...filenames not recognized by VAX will generate
c       an error message and request a second try.

c      Variables:
c        unit = io unit
c        question = query requesting file name
c        inout = 'in' or 'out'
c        form = 'unformatted' (binary) or
c               'formatted' (ascii with no carriage control) or
c               'fortran' (ascii with fortran carriage control).

c      Sample call:
c           call getfile(21,'*Give input file:','in','formatted')

      include 'var.inc/nolist'
      include 'common.inc/nolist'
      integer unit
      character status*8
      character*(*) question, inout, form

 10   write(*,101) question
 101  format(/,'$',a,' ')
      read '(a50)', filename

      if(inout.eq.'in') then
        status='old'
        open(unit,file=filename,form=form,status=status,err=900,
     &      readonly,shared)
c       Note that readonly and shared are VAX specific...

      else if(inout.eq.'out') then
        status='new'

        if(form.eq.'formatted') then
          open(unit,file=filename,form=form,status=status,
     &     carriagecontrol='list',err=900)
        else if(form.eq.'fortran') then
          open(unit,file=filename,form='formatted',status=status,
     &     carriagecontrol='fortran',err=900)
        else if(form.eq.'unformatted') then
          open(unit,file=filename,form=form,status=status,err=900)
        endif
      else
        print *, 'ERROR...inout variable in sub getfile not recognized.
        print *, '   inout =', inout
        stop
      endif
      return

 900  print *,' ERROR IN OPENING ',filename
      print *,'     try again.....'
      close(unit)
      go to 10
      end
```

```
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      subroutine gnapsymb
c     Prompts for plotting symbol for outgnap.
c     Choices are: program supplies a different one for each sample,
c                  or, user supplies a single one for all samples.
c     Fills plotsymb array.
      include 'var.inc/nolist'
      include 'common.inc/nolist'
      character symop*1

c     For first sample....

      if(nsamps.eq.0) then
       print *,' You may specify a plotting symbol;  otherwise a unique'
       print *,' plotting symbol will be assigned to each sample:'
       print '(/,1h$,a)', '*Do you want to specify a plotting symbol
     & (Y/N) ? '
       read '(a)', symop
       if(symop.eq.'Y'.or.symop.eq.'y') then
        print '(/,1h$,a)', '*Give plotting symbol (1 char): '
        read '(a)', symop
        do 40 i=1,84
40      plotsymb(i)=symop
        else

          do 80 i=1,84

c        First 26 samples get plotsymb = A thru Z....
         if(i.lt.27) then
               plotsymb(i)=char(ichar('A')+i-1)

c        Next 26 samples get plotsymb = a thru z.....
         else if(i.lt.53) then
               plotsymb(i)=char(ichar('a')+i-26)

c        Next 31 samples get plotsymb = ! thru ? (ASCII)
         else if(i.lt.84) then
               plotsymb(i)=char(ichar('!')+i-51)

c        For more than 83 samples total, rest get plotsymb = @
         else if(i.eq.84) then
               plotsymb(i)='@'
         endif
80       continue
        endif
       endif
       return
       end
```

```
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      subroutine indnum(n)
c     Converts alphabetic and numeric keys to sequence of integers.
c     A-Y --> 1-25,  1-110 --> 26-135 in tablenum array.

      include 'var.inc/nolist'
      include 'common.inc/nolist'
      character tkey*3

c     functions: convert character string of 1 to 3 numeric characters
c     to integer of same value
      indnum3(tkey)=(ichar(tkey(1:1)) - ichar('1')+1)*100 +
     &              (ichar(tkey(2:2)) - ichar('1')+1)*10 +
     &              (ichar(tkey(3:3)) - ichar('1')+1)
      indnum2(tkey)=(ichar(tkey(2:2)) - ichar('1')+1)*10 +
     &              (ichar(tkey(3:3)) - ichar('1')+1)
      indnum1(tkey)=ichar(tkey(3:3)) - ichar('1')+1

c     function: converts character string of an alphabetic character
c     to number between 1 and 26
      indchar(tkey)=ichar(tkey(3:3)) - ichar('A') + 1

        if (numfirst(entry(n)).ne.0) then
            call ldeblank(entry(n),entry(n),lgth)
            call rjust(entry(n))
          if (lgth.eq.2) then
            tablenum(n)=(indnum2(entry(n))+25)
          else if (lgth.eq.3) then
            tablenum(n)=(indnum3(entry(n))+25)
          else
            tablenum(n)=(indnum1(entry(n))+25)
          endif
        else
            tablenum(n)=(indchar(entry(n)))
        endif

100   continue


      return
      end
```

```
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      subroutine init
c     Reinitializes variables.

      include 'var.inc/nolist'
      include 'common.inc/nolist'
      blankcard=' '

c   sample & category:
      sample=blankcard
      do 10 i=1,10
 10   category(i)=blankcard
      shawnc=blankcard

c   location:
      range=blankcard
      quad=blankcard
      trs=blankcard
      do 20 i=1,ncmt
 20   locacmt(i)=blankcard
      do 25 i=1,nsrc
 25   source(i)=blankcard
      lat=blankcard
      lon=blankcard

c   formation:
      formsymb=blankcard
      formname=blankcard
      do 30 i=1,ncmt
 30   formcmt(i)=blankcard

c   rock types:
      rocktype=blankcard
      grainsize=blankcard
      do 40 i=1,ntext
 40   texture(i)=blankcard
      do 50 i=1,ncmt
 50   rockcmt(i)=blankcard

c   characteristics:
      cps=blankcard
      sg=blankcard
      ci=blankcard
      mag=blankcard
      do 60 i=1,nch
 60   charval(i)=blankcard
      othermin=blankcard
      do 70 i=1,ncmt
 70   charcmt(i)=blankcard

c   chemistry:
      do 80 i=1,nshaw
 80   shawval(i)=blankcard
      do 82 i=1,nxrf
 82   xrfval(i)=blankcard
      do 84 i=1,nniaa
```

```
 84    niaaval(i)=blankcard
       do 90 i=1,nrat
 90    ratio(i)=blankcard
       do 100 i=1,ncmt
100    chemcmt(i)=blankcard
       analyst1=blankcard
       analyst2=blankcard
       analyst3=blankcard

c    isotopes:
       do 110 i=1,nanal
       alysis(i)=blankcard
       alyst(i)=blankcard
       mineral(i)=blankcard
       age(i)=blankcard
110    isotratio(i)=blankcard
       do 120 i=1,ncmt
120    isotcmt(i)=blankcard

       return
       end
```

```
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      subroutine lc2uc(string)
c     Converts lower case letters to upper case...
      character string*(*), s*1

      lstring=len(string)
      do 20 i=1,lstring
      s=string(i:i)
      if(s.ge.'a'. and .s.le.'z') then
       string(i:i)=char(ichar(s)-'a'+'A')
      endif
 20   continue

      return
      end




ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      subroutine ldeblank(string1,string2,l2)
c     Eliminates blanks at left end of string1,
c      returns as string2 with new length l2.

      character*(*) string1, string2

      l1=len(string1)

c     Find first non-blank character on left
      do 10 i=1,l1
 10   if(string1(i:i).ne.' ') go to 20

 20   ifirst=i
      l2=l1-ifirst+1

      string2(1:l2)=string1(ifirst:l1)
      string2(l2+1: )=' '

      return
      end
```

```
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      subroutine makecards

c     Asks for data interactively and makes standard card format output.
c       (Also deletes trailing blanks from long strings (e.g.,COMMENTs)

      include 'var.inc/nolist'
      include 'common.inc/nolist'

      call getfile(22,'*Give card outfile:    ','out','formatted')

 100  print '(/,a,/)', ' Answer with ''\'' to skip a block or to quit.'
      print '(a,/)', ' Answer with ''return'' for a blank.'

      call init

      call ask('sample no (20)= ',sample,*900)
      call rdeblank(sample,sample,1)
      write(22,1100) sample(1:1)
 1100 format('SAMPLE NO=',a)

      call wrloca
      call wrform
      call wrrock
      call wrchar
      call wrchem
      call wrisot

      goto 100

 900  close(22)
      stop
      end



cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      subroutine wrloca
c     Writes cards in a location block.

      include 'var.inc/nolist'
      include 'common.inc/nolist'

      print *, 'LOCATION BLOCK:'

      call ask('mountain range (20)=',range,*999)
      call rdeblank (range,range,1)
      write(22,1100)
 1100 format(' LOCATION:')
      write(22,1110) range(1:1)
 1110 format(10x,'MOUNTAIN RANGE=',a)

      call ask('latitude (6)=',lat,*900)
      call ask('longitude (7)=',lon,*900)
      write(22,1200) lat,lon
 1200 format(10x,'LATITUDE=',a6,4x,'LONGITUDE=',a7)
```

```
      call ask('quadrangle (30)=',quad,*900)
      call rdeblank(quad,quad,1)
      write(22,1220) quad(1:1)
 1220 format(10x,'QUADRANGLE=',a)

      call ask('trs (10)=',trs,*900)
      write(22,1240) trs
 1240 format(10x,'TRS=',a10)

      do 510 i=1,nsrc
      call ask('source (50)=',source(i),*1305)
      call rdeblank(source(i),source(i),1)
      write(22,1300) source(i)(1:1)
 1300 format(10x,'SOURCE=',a)
 510  continue

 1305 do 500 i=1,ncmt
      call ask('comment (50)=',locacmt(i),*900)
      call rdeblank(locacmt(i),locacmt(i),1)
      write(22,1310) locacmt(i)(1:1)
 1310 format(10x,'COMMENT=',a)
 500  continue

 900  write(22,1400)
 1400 format(10x,'COMMENT=$$$')

 999  return
      end


ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      subroutine wrform
c     Writes cards in a formation block.

      include 'var.inc/nolist'
      include 'common.inc/nolist'

      print *, 'FORMATION BLOCK:'

      call ask('formation symbol (20)=',formsymb,*999)
      call rdeblank(formsymb,formsymb,1)
      write(22,1100)
 1100 format(' FORMATION:')
      write(22,1110) formsymb(1:1)
 1110 format(10x,'FORMATION SYMBOL=',a)

      call ask('formation name (40)=',formname,*900)
      call rdeblank(formname,formname,1)
      write(22,1200) formname(1:1)
 1200 format(10x,'FORMATION NAME=',a)

      do 500 i=1,ncmt
      call ask('comment (50)=',formcmt(i),*900)
      call rdeblank(formcmt(i),formcmt(i),1)
      write(22,1300) formcmt(i)(1:1)
```

```
 1300 format(10x,'COMMENT=',a)
 500  continue

 900  write(22,1400)
 1400 format(10x,'COMMENT=$$$')

 999  return
      end


cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      subroutine wrrock
c     Writes cards in a rock type block.

      include 'var.inc/nolist'
      include 'common.inc/nolist'

      print *, 'ROCK TYPE BLOCK:'

      call ask('rock type (40)=',rocktype,*999)
      call rdeblank(rocktype,rocktype,l)
      write(22,1100)
 1100 format(' ROCK TYPE:')
      write(22,1110) rocktype(1:l)
 1110 format(10x,'ROCK TYPE=',a)

      call ask('grain size (40)=',grainsize,*900)
      call rdeblank(grainsize,grainsize,l)
      write(22,1200) grainsize(1:l)
 1200 format(10x,'GRAIN SIZE=',a)

      do 200 i=1,ntext
      call ask('texture (50)=',texture(i),*900)
      call rdeblank(texture(i),texture(i),l)
      write(22,1300) texture(i)(1:l)
 1300 format(10x,'TEXTURE=',a)
 200  continue

      do 500 i=1,ncmt
      call ask('comment (50)=',rockcmt(i),*900)
      call rdeblank(rockcmt(i),rockcmt(i),l)
      write(22,1500) rockcmt(i)(1:l)
 1500 format(10x,'COMMENT=',a)
 500  continue

 900  write(22,1400)
 1400 format(10x,'COMMENT=$$$')

 999  return
      end


cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      subroutine wrchar
c     Writes cards in a characteristics block.
```

```
      include 'var.inc/nolist'
      include 'common.inc/nolist'

      print *, 'CHARACTERISTICS BLOCK:'

      call ask('cps (12)=',cps,*999)
      call ask('sg (5)=',sg,*199)
      call ask('ci (5)=',ci,*199)
      call ask('mag (7)=',mag,*199)
 199  write(22,1100)
 1100 format(' CHARACTERISTICS:')
      write(22,1110) cps,sg,ci,mag
 1110 format(10x,'CPS=',a12,' SG=',a5,' CI=',a5,' MAG=',a7)

      call ask(charname(1)//' (5)=',charval(1),*900)
      do 200 i=2,nch
      call ask(charname(i)//' (5)=',charval(i),*299)
 200  continue
 299  write(22,1200) (charval(i),i=1,nch)
 1200 format(10x,'SN=',a5,' SQ=',a5,' SP=',a5,' SA=',a5,' SM=',a5,
     &  /,10x,'TN=',a5,' TQPA=',a5,' TB=',a5,' TH=',a5,' TM=',a5,
     &    ' TSP=',a5,' TMG=',a5,
     &  /,10x,'%Q1=',a5,' %P1=',a5,' %A1=',a5,' %Q2=',a5,' %P2=',a5,
     &    ' %A2=',a5,' %M2=',a5)

      call ask('other minerals (25)=',othermin,*399)
      call rdeblank(othermin,othermin,l)
      write(22,1300) othermin(1:l)
 1300 format(10x,'OTHER MINERALS=',a)
 399  continue

      do 500 i=1,ncmt
      call ask('comment (50)=',charcmt(i),*900)
      call rdeblank(charcmt(i),charcmt(i),l)
      write(22,1500) charcmt(i)(1:l)
 1500 format(10x,'COMMENT=',a)
 500  continue

 900  write(22,1400)
 1400 format(10x,'COMMENT=$$$')

 999  return
      end


cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      subroutine wrchem
c     Writes cards in a chemistry block.

      include 'var.inc/nolist'

      character chemans*1
      include 'common.inc/nolist'

      print *, 'CHEMISTRY BLOCK:'
```

```
      call ask('chemistry??? [\ or return]: ',chemans,*999)

      write(22,1000)
 1000 format(' CHEMISTRY:')

c...Shaw block

      print *, ' ANALYSIS 1 BLOCK: (eg:SHAW)'
      call ask('analysis 1 by(50):',analyst1,*199)
      call rdeblank(analyst1,analyst1,l)
      do 110 i=1,nshaw
  110 call rjust(shawname(i))
      call ask(shawname(1)//' =',shawval(1),*199)
      do 150 i=2,nshaw
  150 call ask(shawname(i)//' =',shawval(i),*190)
  190 write(22,1100)analyst1(1:l)
 1100 format(10x,'ANALYSIS 1 BY: ',a)
      write(22,1110) (shawname(i),shawval(i),i=1,nshaw)
 1110 format((10x,5(a6,'=',a7)))
      write(22,1120)
 1120 format(10x,8x,'$$$')
  199 continue

c...xrf block

      print *, ' ANALYSIS 2 BLOCK: (eg:USGS(XRF))'
      call ask('analysis 2 by(50):',analyst2,*299)
      call rdeblank(analyst2,analyst2,l)
      do 210 i=1,nxrf
  210 call rjust(xrfname(i))
      call ask(xrfname(1)//' =',xrfval(1),*299)
      do 250 i=2,nxrf
  250 call ask(xrfname(i)//' =',xrfval(i),*290)
  290 write(22,2000)analyst2(1:l)
 2000 format(10x,'ANALYSIS 2 BY: ',a)
      write(22,1110) (xrfname(i),xrfval(i),i=1,nxrf)
      write(22,1120)
  299 continue


c...niaa block

      print *, ' ANALYSIS 3 BLOCK: (eg:USGS(INAA))'
      call ask('analysis 3 by(50):',analyst3,*399)
      call rdeblank(analyst3,analyst3,l)
      do 310 i=1,nniaa
  310 call rjust(niaaname(i))
      call ask(niaaname(1)//' =',niaaval(1),*399)
      do 350 i=2,nniaa
  350 call ask(niaaname(i)//' =',niaaval(i),*390)
  390 write(22,3000)analyst3(1:l)
 3000 format(10x,'ANALYSIS 3 BY: ',a)
      write(22,1115) (niaaname(i),niaaval(i),i=1,nniaa)
 1115 format((14x,4(a2,'=',a10,1x)))
      write(22,1120)
  399 continue
```

```
c...ratio block

      print *, ' RATIO BLOCK:'
      do 410 i=1,nrat
 410  call rjust(rationame(i))
      call ask(rationame(1)//' =',ratio(1),*499)
      do 450 i=2,nrat
 450  call ask(rationame(i)//' =',ratio(i),*490)
 490  write(22,4110) (rationame(i),ratio(i),i=1,nrat)
 4110 format((10x,'RATIOS:'3(a6,'=',a10)))
 499  continue

      do 500 i=1,ncmt
      call ask('comment (50)=',chemcmt(i),*900)
      call rdeblank(chemcmt(i),chemcmt(i),1)
      write(22,1500) chemcmt(i)(1:1)
 1500 format(10x,'COMMENT=',a)
 500  continue

 900  write(22,1400)
 1400 format(10x,'COMMENT=$$$')

 999  return
      end


ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      subroutine wrisot
c     Writes cards in an isotope block.

      include 'var.inc/nolist'
      include 'common.inc/nolist'

      print *, 'ISOTOPE BLOCK:'

      call ask('analysis (5)=',alysis(1),*999)
      call ask('analyst (10)=',alyst(1),*199)
      call ask('mineral (5)=',mineral(1),*199)
      call ask('age (20)=',age(1),*199)
      call ask('ratio (25)=',isotratio(1),*199)
 199  write(22,1000)
 1000 format(' ISOTOPES:')
      write(22,1100) alysis(1),alyst(1),mineral(1),age(1),isotratio(1)
 1100 format(10x,'ANALYSIS=',a5,' ANALYST=',a10,' MINERAL=',a5,
     &  /,10x,'AGE=',a20,' RATIO=',a25)

      do 295 i=2,nanal
      print *, '      --------'
      call ask('analysis (5)=',alysis(i),*299)
      call ask('analyst (10)=',alyst(i),*290)
      call ask('mineral (5)=',mineral(i),*290)
      call ask('age (20)=',age(i),*290)
      call ask('ratio (25)=',isotratio(i),*290)
 290  write(22,1100) alysis(i),alyst(i),mineral(i),age(i),isotratio(i)
 295  continue
```

```
299   continue

      do 500 i=1,ncmt
      call ask('comment  (50)=',isotcmt(i),*900)
      call rdeblank(isotcmt(i),isotcmt(i),l)
      write(22,1500) isotcmt(i)(1:l)
1500  format(10x,'COMMENT=',a)
500   continue

900   write(22,1400)
1400  format(10x,'COMMENT=$$$')

999   return
      end
```

```
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      subroutine newcmt(cmt,icmt,card,ncards)
c     Appends another comment to comment array

      character card*80, cmt(*)*50

      if(icmt.gt.5) then
          print *, 'Too many comment lines at line',ncards
          print *, 'Trying to continue...'
      else
          cmt(icmt)=card(19:68)
      endif

      return
      end




ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      subroutine newsourc (src,isource,card,ncards)
c     Appends another literature reference source to source array

      character card*80, src(*)*50

      if(isource.gt.10) then
          print *, 'Too many source lines at line',ncards
          print *, 'Trying to continue...'
      else
          src(isource)=card(18:67)
      endif

      return
      end
```

```
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      function numfirst(string)
c     Finds the position of the first number in a string...
      character string*(*)

      l=len(string)
      numfirst=0

      do 20 i=1,l
      if(string(i:i).ge.'0'.and.string(i:i).le.'9') then
        numfirst=i
        return
      endif
 20   continue

      return
      end
```

```
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      function numfirst(string)
c     Finds the position of the first number in a string...
      character string*(*)

      l=len(string)
      numfirst=0

      do 20 i=1,l
      if(string(i:i).ge.'0'.and.string(i:i).le.'9') then
        numfirst=i
        return
      endif
 20   continue

      return
      end
```

```
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      subroutine out1 (symdex)
c     Outputs 2 analysis cards for tomgnap
c     of major element and trace element chemistry
c      from analysis 1.

      include 'var.inc/nolist'
      include 'common.inc/nolist'
c      common/plotf/ ckey, nkey
      common/index/ i

      character ident*8
      integer idata (100)


      blankcard=' '
      ident=blankcard
c     Get last 8 characters of sample number for ident
      call rdeblank(sample,sample1,ns)
      if(ns.gt.8) then
         call remchar ('-',sample,sample1,12)
         call rdeblank(sample1,sample1,ns)
      endif
      ident=sample1(ns-7:ns)

      call fillarry

      numlist(36)=numlist(36)+numlist(37)

      idata(1)=0
      do 210 i=25,38
 210  idata(i)=nint(100.0*numlist(i))
      do 220 i=40,54
 220  idata(i)=nint(numlist(i))

      idata(1)=0
      do 230 i=55,59
230   idata(i)=nint(100.0*numlist(i))
      do 240 i=60,64
240   idata(i)=nint(100.0*numlist(i))
      do 250 i=65,70
250   idata(i)=nint(numlist(i))
      idata(71)=nint(100.0*numlist(71))

      call ldeblank(analyst1,analyst1,11)
      call ldeblank(analyst2,analyst2,12)

      if (analyst1 .ne. blankcard) then
         if (analyst1 .eq. 'S.E.SHAW') then
            do 1500 j=1,5
              if (locacmt(j)(1:6).eq.'SHAWNO') then
                 nidx=index(locacmt(j)(8:12),'S')+7
                 ident=locacmt(j)(8:nidx)
              endif
1500          continue
         endif
```

```
          write(30,1100)plotsymb(min(shawtot+1,nsymbs)),idata(25),
     &     idata(27),idata(28),idata(29),
     &     idata(31),idata(32),idata(33),idata(34),idata(36),idata(26),
     &     idata(35),idata(30),idata(38),ident

          write(30,1200)plotsymb(min(shawtot+1,nsymbs)),idata(40),
     &     idata(41),idata(42),idata(43),idata(44),idata(45),idata(46),
     &     idata(47),idata(48),idata(49),idata(50),idata(51),idata(52),
     &     idata(53),idata(54),ident

1100       format( 'NRM',a1,12I4,3x,I4,2x,3x,3x,3x,2x,a8)
1200       format( 'NRM',a1,15I4,8x,a8)
           shawtot=shawtot+1

        endif

        return
        end
```

```
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      subroutine out2 (symdex)
c     Outputs 2 analysis cards for tomgnap
c     of major element and trace element chemistry from analysis 2.

      include 'var.inc/nolist'
      include 'common.inc/nolist'
c      common/plotf/ ckey, nkey
      common/index/ i

      character ident*8
      integer idata (100)

      blankcard=' '
      ident=blankcard
c     Get last 8 characters of sample number for ident
      call rdeblank(sample,sample1,ns)
      if(ns.gt.8) then
         call remchar ('-',sample,sample1,12)
         call rdeblank(sample1,sample1,ns)
      endif
      ident=sample1(ns-7:ns)

      call fillarry
      numlist(36)=numlist(36)+numlist(37)

      idata(1)=0
      do 210 i=25,38
 210  idata(i)=nint(100.0*numlist(i))
      do 220 i=40,54
 220  idata(i)=nint(numlist(i))

      idata(1)=0
      do 230 i=55,59
230   idata(i)=nint(100.0*numlist(i))
      do 240 i=60,64
240   idata(i)=nint(100.0*numlist(i))
      do 250 i=65,70
250   idata(i)=nint(numlist(i))
      idata(71)=nint(100.0*numlist(71))

      call ldeblank(analyst1,analyst1,11)
      call ldeblank(analyst2,analyst2,12)

      if (analyst2 .ne. blankcard) then
         write(30,1300)plotsymb(min(shawtot+1,nsymbs)),idata(55),
     &     idata(57),idata(58),idata(71),idata(60),idata(61),idata(62)
     &     idata(63),idata(56),idata(64),idata(59),ident
         write(30,1400)plotsymb(min(shawtot+1,nsymbs)),idata(65),
     &     idata(66),idata(67),idata(68),idata(69),idata(70),ident
1300     format( 'NRM',a1,8I4,4x,3i4,20x,a8)
1400     format( 'NRM',a1,3i4,4x,i4,32x,2i4,8x,a8)
         shawtot=shawtot+1
      endif
      return
      end
```

```
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      subroutine outinaa (symdex)
c     Outputs 3 analysis cards for tomgnap
c     of analysis 1 and analysis 3 chemistry

      include 'var.inc/nolist'
      include 'common.inc/nolist'
c      common/plotf/ ckey, nkey
      common/index/ i

      character ident*8
      integer idata (100)



c      Get last 8 characters of sample number for ident
       call rdeblank(sample,sample1,ns)
       if(ns.gt.8) then
        call remchar ('-',sample,sample1,12)
        call rdeblank(sample1,sample1,ns)
        else
       endif
       ident=sample1(ns-7:ns)


      call fillarry


      idata(1)=0
      do 210 i=25,38
 210  idata(i)=nint(100.0*numlist(i))

      idata(40)=nint(numlist(40))
      idata(41)=nint(numlist(41))
      idata(42)=nint(numlist(42))
      idata(44)=nint(numlist(44))
      idata(53)=nint(numlist(53))
      idata(54)=nint(numlist(54))
      idata(75)=nint(numlist(75))
      idata(83)=nint(numlist(83))

      idata(80)=nint(10.0*numlist(80))
      idata(81)=nint(10.0*numlist(81))
      idata(87)=nint(10.0*numlist(87))
      idata(91)=nint(10.0*numlist(91))

      idata(73)=nint(100.0*numlist(73))
      idata(74)=nint(100.0*numlist(74))
      idata(79)=nint(100.0*numlist(79))
      idata(89)=nint(100.0*numlist(89))
      idata(90)=nint(100.0*numlist(90))
      idata(92)=nint(100.0*numlist(92))
      idata(94)=nint(100.0*numlist(94))
      idata(96)=nint(100.0*numlist(96))

      idata(72)=nint(1000.0*numlist(72))
```

```
      idata(76)=nint(1000.0*numlist(76))
      idata(77)=nint(1000.0*numlist(77))
      idata(78)=nint(1000.0*numlist(78))
      idata(84)=nint(1000.0*numlist(84))
      idata(85)=nint(1000.0*numlist(85))
      idata(86)=nint(1000.0*numlist(86))
      idata(88)=nint(1000.0*numlist(88))
      idata(93)=nint(1000.0*numlist(93))
      idata(97)=nint(1000.0*numlist(97))
      idata(98)=nint(1000.0*numlist(98))
      idata(99)=nint(1000.0*numlist(99))

      idata(82)=nint(10000.0*numlist(82))
      idata(95)=nint(10000.0*numlist(95))

      write(29,1100)plotsymb(min(shawtot+1,nsymbs)),idata(25),
     &  idata(26),idata(27),idata(28),idata(29),idata(30),idata(31),
     &  idata(32),idata(33),idata(34),idata(35),idata(36),idata(37),
     &  idata(38),idata(40),idata(41),idata(42),idata(44),ident

      write(29,1200)plotsymb(min(shawtot+1,nsymbs)),idata(53),
     &  idata(54),idata(72),idata(73),idata(74),idata(75),idata(76),
     &  idata(77),idata(78),idata(79),idata(80),idata(81),idata(82),
     &  idata(83),idata(84),idata(85),ident

      write(29,1300)plotsymb(min(shawtot+1,nsymbs)),idata(86),
     &  idata(87),idata(88),idata(89),idata(90),idata(91),idata(92),
     &  idata(93),idata(94),idata(95),idata(96),idata(97),idata(98),
     &  idata(99),ident

1100  format( 'NRM',a1,I4,I3,3I4,I2,4I4,I3,3I4,I3,2I4,I3,2X,a8)
1200  format( 'NRM',a1,i4,i2,i5,3i4,i5,i6,2i4,i5,4i4,i5,a8)
1300  format( 'NRM',a1,i4,4i5,4i4,i5,2i4,i5,i4,6x,a8)
      shawtot=shawtot+1

      return
      end
```

```
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      subroutine outsamp
c     Outputs a sample.

      include 'var.inc/nolist'
      include 'common.inc/nolist'

      character spacer*126, space1*126, space2*126

      do 10 i=1,126
      spacer(i:i)='$'
      space1(i:i)='-'
 10   space2(i:i)='='

      write(22,1000) spacer
 1000 format(/,'1',a126)

      do 1008 i=1,3
      write(22,1005)blankcard
 1008 continue
 1005 format(' ',a80)

c...LOCATION

      write(22,1010) 'SAMPLE:',sample,'RANGE:',range,'QUAD:',quad,
     &   'TRS:',trs
 1010 format(/,' ',a7,1x,a20,5x,a6,1x,a20,5x,a5,1x,a30,5x,a4,1x,a10)
      write(22,1012) space1(1:20)
 1012 format(' ',8x,a20)
      write(22,1016) lat(1:2),lat(3:4),lat(5:6),
     &   lon(1:3),lon(4:5),lon(6:7)
 1016 format(' ',18x,'lat=',a2,':',a2,':',a2,5x,'lon=',a3,':',a2,':',a2)

      do 25 i=1,nsrc
 25   if(source(i).ne.blankcard)       write(22,1025) source(i)
 1025   format(' ',18x,'   source:    ',a50)

      do 20 i=1,ncmt
 20   if(locacmt(i).ne.blankcard)      write(22,1020) locacmt(i)
 1020   format(' ',18x,'   comment:   ',a50)

c...FORMATION

      write(22,1040) 'FORMATION:',formsymb,formname
 1040 format(/,' ',a15,3x,'symbol=',a20,'    name=',a40)
      do 40 i=1,ncmt
 40   if(formcmt(i).ne.blankcard)      write(22,1020) formcmt(i)

c...ROCK TYPE

      write(22,1060) 'ROCK TYPE:',rocktype,grainsize
 1060 format(/,' ',a15,3x,a40,5x,'grain size: ',a40)
      do 60 i=1,ntext
 60   if(texture(i).ne.blankcard)      write(22,1062) texture(i)
 1062   format(' ',18x,'   texture:   ',a50)
      do 65 i=1,ncmt
```

```
 65   if(rockcmt(i).ne.blankcard)      write(22,1020) rockcmt(i)

c...CHARACTERISTICS

      write(22,1080) 'CHARACTER:',cps,sg,ci,mag
 1080 format(/,' ',a15,3x,'cps=',a12,' sg=',a5,'    ci=',a5,' mag=',a7)
      write(22,1081)  (charname(i),charval(i), i=1,5)
 1081 format(' ',18x,'slab:         ',7(1x,a4,': ',a5))
      write(22,1082) (charname(i),charval(i),i=6,12)
 1082 format(' ',18x,'thin section:',7(1x,a4,': ',a5))
      write(22,1084) (charname(i),charval(i),i=13,19)
 1084 format(' ',18x,'modes:        ',7(1x,a4,':',a5,'%'))
      write(22,1086) othermin
 1086 format(' ',18x,'other minerals = ',a25)
      do 80 i=1,ncmt
 80   if(charcmt(i).ne.blankcard)      write(22,1020) charcmt(i)

c...CHEMISTRY

      call ckblank(shawval,nshaw,numshaw)
      call ckblank(xrfval,nxrf,numxrf)
      call ckblank(niaaval,nniaa,numniaa)
      call ckblank(ratio,nrat,numrat)

      if(numshaw.ne.0.or.numxrf.ne.0.or.numniaa.ne.0.or.
     &   numrat.ne.0) then
      write(22,2100) 'CHEMISTRY:'
 2100 format(/,' ',a15,3x,a108)

      if(numshaw.ne.0) then
      write(22,2110) 'analyzed by: ',analyst1
 2110 format(' ',18x,a12,a50)
      write(22,2250)space2(1:108)
      write(22,2200) (shawname(i), i=1,15)
 2200 format(' ',15x,3x,4x,15(a5,2x))
      write(22,2210) (shawval(i), i=1,15)
 2210 format(' ',18x,' %  ',15a7)
      write(22,2250) space1(1:108)
      write(22,2220) (shawname(i), i=16,30)
 2220 format(' ',18x,5x,15(a5,2x))
      write(22,2230) (shawval(i), i=16,30)
 2230 format(' ',18x,'ppm ',15a7)
      write(22,2250) space2(1:108)
 2250 format(' ',18x,a108)
      endif

      if(numxrf.ne.0) then
      write(22,2110) 'analyzed by: ',analyst2
      write(22,2250)space2(1:108)
      write(22,2200) (xrfname(i), i=1,10)
      write(22,2210) (xrfval(i), i=1,10)
      write(22,2250) space1(1:108)
      write(22,2220) (xrfname(i), i=11,16)
      write(22,2230) (xrfval(i), i=11,16)
      write(22,2250) space2(1:108)
      endif
```

```
      if(numniaa.ne.0) then
      write(22,2110) 'analyzed by: ',analyst3
      write(22,2250)space2(1:108)

      write(22,2270) (niaaname(i), i=1,3)
 2270 format(' ',15x,4x,9(5x,a2,4x))
      write(22,2275) (niaaval(i), i=1,3)
 2275 format(' ',18x,' %  ',3(1x,a10))
      write(22,2250) space1(1:108)

      write(22,2270) (niaaname(i), i=4,12)
      write(22,2280) (niaaval(i), i=4,12)
 2280 format(' ',18x,'ppm ',9(1x,a10))
      write(22,2250) space1(1:108)

      write(22,2270) (niaaname(i), i=13,21)
      write(22,2280) (niaaval(i), i=13,21)
      write(22,2250) space1(1:108)

      write(22,2270) (niaaname(i), i=22,28)
      write(22,2280) (niaaval(i), i=22,28)
      write(22,2250) space2(1:108)
      endif

      if(numrat.ne.0) then
      write(22,2110) 'ratios:'
      write(22,2250)space2(1:108)
      write(22,2300) (rationame(i),i=1,6)
 2300 format(' ',15x,3x,4x,6(7x,a5,2x))
      write(22,2310) (ratio(i),i=1,6)
 2310 format(' ',18x,4x,6(7x,a7))
      write(22,2250) space2(1:108)
      endif

      do 280 i=1,ncmt
 280  if(chemcmt(i).ne.blankcard)      write(22,1020) chemcmt(i)

      endif

c...ISOTOPES

      if(alysis(1).ne.blankcard) then
      write(22,2100) 'ISOTOPES:'

      do 300 i=1,nanal
 300  if(alysis(i).ne.blankcard)
     &  write(22,3100) alysis(i),alyst(i),mineral(i),age(i),isotratio(i)
 3100 format(' ',18x,'analysis=',a5,2x,'analyst=',a10,2x,'mineral=',
     &  a5,2x,'age=',a20,2x,'ratio=',a25)
c       write(22,2250) space2(1:108)
      do 380 i=1,ncmt
 380  if(isotcmt(i).ne.blankcard)      write(22,1020) isotcmt(i)
      endif
      return
      end
```

```
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      subroutine outtable
c     Outputs a 126-column table of user specified data.

      include 'var.inc/nolist'
      include 'common.inc/nolist'
c      common/plotf/ ckey, nkey
      common/index/ i
      common/cardcount/ lastsamp

      character space2*126, tstfmt*30
      character*9 forchr1(40), forchr2(40), forchr3(40), forchr4(40)
      character geol*2, year*2, rang*3, sn*3, sna*2
      integer ccateg(30),rcateg(30)

      data (forchr1(i), i=1,7)/'(i5,','i5,2x,','a2,1x,','a2,1x,',
     &                         'a3,1x,','a3,1x,','a2,1x,'/
      data (forchr2(i), i=1,5)/'(a2,1x,','a2,1x,','a3,1x,','a3,1x,',
     &                         'a2,1x,'/
      data (forchr3(i), i=1,3)/'(i5,','i5,2x,','a16,1x,'/
      data  forchr4(1)        /'(a16,1x,'/

      do 10 i=1,126
10    space2(i:i)='='

      newsamp=ncards
      nfirst=lastsamp
      numfirst=newsamp-lastsamp

      call fillarry


      if (expand.eq.'Y') call breaksn(sample,geol,year,rang,
     &                                                sn,sna)

      j=0
      k=0
      do 200 i=1,nchoices
        if (tablenum(i).gt.25) then
          k=k+1
          rcateg(k)=tablenum(i)
        else
          j=j+1
          ccateg(j)=tablenum(i)
        endif
200   continue

      if (expand.eq.'Y') then
        if (uflag.eq.'Y') then

            do 300 i=8,j+7
               forchr1(i)=tablefmt(ccateg(i-7))
300         continue
            do 400 i=8+j,j+k+7
               forchr1(i)=tablefmt(rcateg(i-7-j))
400         continue
```

```
            forchr1(j+k+7)(8:8)=')'

            write(28,forchr1) nfirst,numfirst,geol,year,rang,sn,sna,
     &          (id(ccateg(i)), i=1,j), (numlist(rcateg(i)-25), i=1,k)
         else
            do 500 i=6,j+5
               forchr2(i)=tablefmt(ccateg(i-5))
500         continue
            do 600 i=6+j,j+k+5
               forchr2(i)=tablefmt(rcateg(i-5-j))
600         continue
            forchr2(j+k+5)(8:8)=')'
            write(28,forchr2) geol,year,rang,sn,sna,
     &          (id(ccateg(i)), i=1,j), (numlist(rcateg(i)-25), i=1,k)
         endif
       else
         if (uflag.eq.'Y') then
            do 700 i=4,j+3
               forchr3(i)=tablefmt(ccateg(i-3))
700         continue
            do 800 i=4+j, j+k+3
               forchr3(i)=tablefmt(rcateg(i-3-j))
800         continue
            forchr3(j+k+3)(8:8)=')'
            write(28,forchr3) sample,nfirst,numfirst,
     &          (id(ccateg(i)), i=1,j), (numlist(rcateg(i)-25), i=1,k)
         else
            do 900 i=2,j+1
               forchr4(i)=tablefmt(ccateg(i-1))
900         continue
            do 1000 i=j+2,k+j+1
               forchr4(i)=tablefmt(rcateg(i-j-1))
1000        continue
            forchr4(j+k+1)(9:9)=')'
            write(28,forchr4) sample, (id(ccateg(i)), i=1,j),
     &                        (numlist(rcateg(i)-25), i=1,k)
         endif
       endif

       lastsamp=newsamp

       return
       end
```

```
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      subroutine rdeblank(string1,string2,l2)
c     Eliminates blanks at right end of string1,
c      returns as string2 with new length l2.

      character*(*) string1, string2

      l1=len(string1)

c     Find last non-blank character on right
      do 10 i=l1,1,-1
 10   if(string1(i:i).ne.' ') go to 20

 20   ilast=i
      l2=ilast

      string2(1:ilast)=string1(1:ilast)
      string2(l2+1: )=' '

      return
      end
```

```
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      subroutine readchar
c     Reads cards in characteristics block.

      include 'var.inc/nolist'
      include 'common.inc/nolist'

      icmt=0
 100  read(21,'(a80)',end=900) card
      ncards=ncards+1
      if(card.eq.blankcard) then
        continue

      else if(card(1:10).ne.'          ') then
        print *,' Charact. block not properly ended before line',ncards
        call badcard (card,sample)
       print*,'Trying to continue ...'
        return

      else if(card(11:21).eq.'COMMENT=$$$') then
        return
      else if(card(11:14).eq.'CPS=') then
        read(card,1010,err=990) cps, sg, ci, mag
 1010   format(14x,a12,4x,a5,4x,a5,5x,a7)

      else if(card(11:13).eq.'SN=') then
        read(card,1020,err=990) (charval(i), i=1,5)
 1020   format(13x,a5,4(4x,a5))

      else if(card(11:13).eq.'TN=') then
        read(card,1030,err=990) (charval(i), i=6,12)
 1030   format(13x,a5,6x,a5,3(4x,a5),2(5x,a5))

      else if(card(11:14).eq.'%Q1=') then
        read(card,1040,err=990) (charval(i), i=13,19)
 1040   format(14x,a5,6(5x,a5))
      else if(card(11:25).eq.'OTHER MINERALS=') then
        othermin=card(26:50)

      else if(card(11:18).eq.'COMMENT=') then
        icmt=icmt+1
        call newcmt(charcmt,icmt,card,ncards)

      else
        print *, 'Unrecognized line in a charact. block =', ncards
        call badcard (card,sample)
        print *, 'Trying to continue...'
      endif
       goto 100

 990  call badcard (card,sample)
      call carderror(ncards)
      print*,'Trying to continue ...'
      goto 100
 900  return
      end
```

```
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      subroutine readchem
c     Reads cards in chemistry block.

      include 'var.inc/nolist'
      include 'common.inc/nolist'

      icmt=0
      xrfvar='no'

 100  read(21,'(a80)',end=900) card
      ncards=ncards+1

      if(card.eq.blankcard) then
        continue

      else if(card(1:10).ne.'          ') then
        print *,' Chemistry block not properly ended before line',ncards
        call badcard (card, sample)
       print*,'Trying to continue ...'
        return

      else if(card(11:21).eq.'COMMENT=$$$') then
        return

      else if(card(24:27).eq.'SHAW') then
        analyst1='S.E.SHAW'
        call readshaw
      else if(card(11:24).eq.'ANALYSIS 1 BY:') then
        analyst1=card(25: )
        call readshaw

      else if(card(24:33).eq.'USGS (XRF)') then
        xrfvar = 'yes'
        analyst2='USGS (XRF)'
        call readxrf
      else if(card(11:24).eq.'ANALYSIS 2 BY:') then
        xrfvar = 'yes'
        analyst2=card(25: )
        call readxrf

      else if(card(24:34).eq.'USGS (NIAA)') then
        analyst3='USGS (NIAA)'
        call readniaa
      else if(card(11:24).eq.'ANALYSIS 3 BY:') then
        analyst3=card(25: )
        call readniaa

      else if(card(11:24).eq.'RATIOS: BA/RB=') then
        read(card,1040,err=990) (ratio(i), i=1,3)
1040    format(24x,a10,2(7x,a10))

      else if(card(11:24).eq.'RATIOS:  U/TH=') then
        read(card,1040,err=990) (ratio(i), i=4,6)

      else if(card(11:18).eq.'COMMENT=') then
```

```
        icmt=icmt+1
        call newcmt(chemcmt,icmt,card,ncards)

      else
        print *, 'Unrecognized line in a chemistry block =',ncards
        call badcard (card,sample)
        print *, 'Trying to continue...'
      endif

      goto 100

990   call badcard (card,sample)
      call carderror(ncards)
       print*,'Trying to continue ...'

      goto100

900   return

      end
```

```
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      subroutine readform
c     Reads cards in a formation block.

      include 'var.inc/nolist'
      include 'common.inc/nolist'

      icmt=0

 100  read(21,'(a80)',end=900) card
      ncards=ncards+1

      if(card.eq.blankcard) then
        continue

      else if(card(1:10).ne.'          ') then
        print *,' Formation block not properly ended before line',ncards
        call badcard (card,sample)
       print*,'Trying to continue ...'
        return

      else if(card(11:21).eq.'COMMENT=$$$') then
        return

      else if(card(11:27).eq.'FORMATION SYMBOL=') then
        formsymb=card(28:47)

      else if(card(11:25).eq.'FORMATION NAME=') then
        formname=card(26:65)

      else if(card(11:18).eq.'COMMENT=') then
        icmt=icmt+1
        call newcmt(formcmt,icmt,card,ncards)

      else
        print *, 'Unrecognized line in a formation block =', ncards
        call badcard (card,sample)
        print *, 'Trying to continue...'
      endif

      goto 100

 900  return
      end
```

```
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      subroutine readisot
c     Reads cards in an isotope block.

      include 'var.inc/nolist'
      include 'common.inc/nolist'

      icmt=0
      ianal=0

 100  read(21,'(a80)',end=900) card
      ncards=ncards+1

      if(card.eq.blankcard) then
        continue

      else if(card(1:10).ne.'          ') then
        print *,' Isotope block not properly ended before line',ncards
        call badcard (card,sample)
       print*,'Trying to continue ...'
        return

      else if(card(11:21).eq.'COMMENT=$$$') then
        return

      else if(card(11:19).eq.'ANALYSIS=') then
        ianal=ianal+1
        if(ianal.gt.nanal) then
          print *, 'Too many analysis lines at line',ncards
          print *, 'Trying to continue...'
        else
          read(card,1010,err=990) alysis(ianal),alyst(ianal),
     &     mineral(ianal)
 1010     format(19x,a5,9x,a10,9x,a5)
        endif

      else if(card(11:14).eq.'AGE=') then
        read(card,1020,err=990) age(ianal), isotratio(ianal)
 1020   format(14x,a20,7x,a25)

      else if(card(11:18).eq.'COMMENT=') then
        icmt=icmt+1
        call newcmt(isotcmt,icmt,card,ncards)

      else
        print *, 'Unrecognized line in an analysis block =', ncards
        call badcard (card,sample)
        print *, 'Trying to continue...'
      endif
      goto 100
 990  call badcard (card,sample)
      call carderror(ncards)
       print*,'Trying to continue ...'
      goto 100
 900  return
      end
```

```
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      subroutine readloca
c     Reads cards in a location block.

      include 'var.inc/nolist'
      include 'common.inc/nolist'

      icmt=0
      isource=0

 100  read(21,'(a80)',end=900) card
      ncards=ncards+1

      if(card.eq.blankcard) then
        continue

      else if(card(1:10).ne.'          ') then
        print *,' Location block not properly ended before line',ncards
        call badcard (card,sample)
        print*,'Trying to continue...'
        return

      else if(card(11:21).eq.'COMMENT=$$$') then
        return

      else if(card(11:25).eq.'MOUNTAIN RANGE=') then
        range=card(26:45)

      else if(card(11:19).eq.'LATITUDE=') then
        read(card(20:25),'(a6)') lat
        read(card(40:46),'(a7)') lon

      else if(card(11:21).eq.'QUADRANGLE=') then
        quad=card(22:51)

      else if(card(11:14).eq.'TRS=') then
        trs=card(15:24)

      else if(card(11:17).eq.'SOURCE=') then
        isource=isource+1
        call newsourc(source,isource,card,ncards)

      else if(card(11:18).eq.'COMMENT=') then
        icmt=icmt+1
        call newcmt(locacmt,icmt,card,ncards)

      else
        print *, 'Unrecognized line in a location block =', ncards
        call badcard (card,sample)
        print *, 'Trying to continue...'
      endif

      goto 100

 900  return
      end
```

```
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      subroutine readniaa
c     Reads cards in chemistry block.

      include 'var.inc/nolist'
      include 'common.inc/nolist'

 100  read(21,'(a80)',end=900) card
      ncards=ncards+1

      if(card.eq.blankcard) then
        continue

      else if(card(1:12).ne.'            ') then
        print *,' Analysis 3 block not properly ended before
     &  line',ncards
        call badcard (card,sample)
        print*,'Trying to continue ...'
        return

      else if(card(19:21).eq.'$$$') then
        return
      else if(card(15:17).eq.'FE=') then
        read(card,1010,err=990) (niaaval(i), i=1,4)
 1010   format(13x,4(4x,a10))
      else if(card(15:17).eq.'CO=') then
        read(card,1010,err=990) (niaaval(i), i=5,8)
      else if(card(15:17).eq.'MN=') then
        read(card,1010,err=990) (niaaval(i), i=9,12)
      else if(card(15:17).eq.'TA=') then
        read(card,1010,err=990) (niaaval(i), i=13,16)
      else if(card(15:17).eq.'SC=') then
        read(card,1010,err=990) (niaaval(i), i=17,20)
      else if(card(15:17).eq.'SM=') then
        read(card,1010,err=990) (niaaval(i), i=21,24)

      else if(card(15:17).eq.'DY=') then
        read(card,1010,err=990) (niaaval(i), i=25,28)

      else if(card(15:17).eq.'HO=') then
        read(card,1010,err=990) niaaval(29)

      else
        print *, 'Unrecognized line in Analysis 3 block =',ncards
        call badcard (card,sample)
        print *, 'Trying to continue...'
      endif
      goto 100

 990  call badcard (card,sample)
      call carderror(ncards)
       print*,'Trying to continue ...'
      goto 100

 900  return
      end
```

```
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      subroutine readrock
c     Reads cards in a rock type block.

      include 'var.inc/nolist'
      include 'common.inc/nolist'

      icmt=0
      itexture=0

 100  read(21,'(a80)',end=900) card
      ncards=ncards+1

      if(card.eq.blankcard) then
        continue

      else if(card(1:10).ne.'          ') then
        print *,' Rock type block not properly ended before line',ncards
        call badcard (card,sample)
       print*,'Trying to continue ...'
        return

      else if(card(11:21).eq.'COMMENT=$$$') then
        return

      else if(card(11:20).eq.'ROCK TYPE=') then
        rocktype=card(21:60)

      else if(card(11:21).eq.'GRAIN SIZE=') then
        grain size=card(22:61)

      else if(card(11:18).eq.'TEXTURE=') then
        itexture=itexture+1
        if(itexture.gt.4) then
          print *, 'Too many texture lines at line',ncards
          call badcard (card,sample)
          print *, 'Trying to continue...'
        else
          texture(itexture)=card(19:68)
        endif

      else if(card(11:18).eq.'COMMENT=') then
        icmt=icmt+1
        call newcmt(rockcmt,icmt,card,ncards)

      else
        print *, 'Unrecognized line in a rock type block =', ncards
        call badcard (card,sample)
        print *, 'Trying to continue...'
      endif

      goto 100

 900  return
      end
```

```
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      subroutine readshaw
c     Reads cards in chemistry block.

      include 'var.inc/nolist'
      include 'common.inc/nolist'

 100  read(21,'(a80)',end=900) card
      ncards=ncards+1

      if(card.eq.blankcard) then
        continue

      else if(card(1:12).ne.'            ') then
        print *,' Analysis 1 block not properly ended before
     &  line',ncards
        call badcard (card,sample)
        print*,'Trying to continue ...'
        return

      else if(card(19:21).eq.'$$$') then
        return

      else if(card(13:17).eq.'SIO2=') then
        read(card,1010,err=990) (shawval(i), i=1,5)
 1010   format(10x,5(7x,a7))

      else if(card(13:17).eq.' MNO=') then
        read(card,1010,err=990) (shawval(i), i=6,10)

      else if(card(13:17).eq.'P2O5=') then
        read(card,1010,err=990) (shawval(i), i=11,15)

      else if(card(13:17).eq.'   Y=') then
        read(card,1010,err=990) (shawval(i), i=16,20)

      else if(card(13:17).eq.'  TH=') then
        read(card,1010,err=990) (shawval(i), i=21,25)

      else if(card(13:17).eq.'  NI=') then
        read(card,1010,err=990) (shawval(i), i=26,30)

      else
        print *, 'Unrecognized line in Analysis 1 block =',ncards
        call badcard (card,sample)
        print *, 'Trying to continue...'
      endif

      goto 100

 990  call badcard (card,sample)
      call carderror(ncards)
       print*,'Trying to continue ...'
      goto 100
 900  return
      end
```

```
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      subroutine readxrf
c     Reads cards in chemistry block.

      include 'var.inc/nolist'
      include 'common.inc/nolist'

 100  read(21,'(a80)',end=900) card
      ncards=ncards+1

      if(card.eq.blankcard) then
        continue

      else if(card(1:12).ne.'            ') then
        print *,' Analysis 2 block not properly ended before
     &  line',ncards
        call badcard (card,sample)
        print*,'Trying to continue ...'
        return

      else if(card(19:21).eq.'$$$') then
        return

      else if(card(13:17).eq.'SIO2=') then
        read(card,1010,err=990) (xrfval(i), i=1,5)
 1010   format(10x,5(7x,a7))

      else if(card(13:17).eq.' MGO=') then
        read(card,1010,err=990) (xrfval(i), i=6,10)

      else if(card(13:17).eq.'   Y=') then
        read(card,1010,err=990) (xrfval(i), i=11,15)

      else if(card(13:17).eq.'  NB=') then
        read(card,1010,err=990) (xrfval(i), i=16,17)

      else
        print *, 'Unrecognized line in Analysis 2 block =',ncards
        call badcard (card,sample)
        print *, 'Trying to continue...'
      endif

      goto 100

 990  call badcard (card,sample)
      call carderror(ncards)
       print*,'Trying to continue ...'

      goto 100

 900  return

      end
```

```
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      subroutine remchar (ch, string1, string2, l2)
c     Removes character from string and returns length of new string

      character*1 ch
      character*(*) string1, string2

      l1=len(string1)
      index=1

      do 10 i=1,l1
      if (string1(i:i).ne.ch) then
       string2(index:index)=string1(i:i)
       index=index+1
       else
       endif
10    continue

      l2=len(string2)
      return
      end



cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      subroutine rjust(string)
c     Right justifies a string in its alloted length.

      character*(*) string
      character*120 temp
      temp=' '

      l=len(string)

c     Find last non-blank character on right
      do 10 i=l,1,-1
 10   if(string(i:i).ne.' ') go to 20

 20   ilast=i

      temp(l-ilast+1:l)=string(1:ilast)
      string(1:l)=temp(1:l)

      return
      end
```

```
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      function rlnu(string)
c     Converts a character string containing a number to
c      the appropriate floating point number.
c     Built in default converts a blank string to 0.
c     Removes less-than sign (<).

      character string*(*), fmt*20
      parameter (default=0.)
      common/index/i
      include 'var.inc/nolist'
      include 'common.inc/nolist'

      call remchar('<',string,string,10)
      call ldeblank(string,string,11)
      call rdeblank(string,string,12)

      if(12.eq.0) then
        rlnu=default
      else
          write(fmt,1000) 12
1000      format('(f',i2,'.0)')

        read(string(1:12),fmt,err=900) rlnu
      endif

      return

 900  print *, 'ERROR trying to convert string to real number...'
      print *, string
      print *, 'Sample= ',Sample
      stop
      end
```

```
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      subroutine shuffle
c     Sorts a card format input file according to an index.
c     20000 cards max

      parameter (maxdim=20000)
      character*80 card(maxdim), blankcard

      do 5 i=1,80
 5    blankcard(i:i)=' '

      call getfile(21,'*Give index file: ','in','formatted')
      call getfile(22,'*Give infile:     ','in','formatted')
      call getfile(23,'*Give outfile:    ','out','formatted')

      n=1
 10   read(22,'(a80)',end=50) card(n)
      n=n+1
      if(n.gt.maxdim) then
        print *, 'ERROR...array not dimensioned big enough'
        stop
      endif
      go to 10

 50   read(21,'(2i5)',end=900) nfirst,num
      do 60 j=nfirst,nfirst+num-1
      call rdeblank(card(j), card(j), 1)
 60   write(23,'(a)') card(j)(1:1)
      go to 50


 900  print*,'Number of cards read=',n
      stop
      end
```