Computers & Geosciences, Vol. 3, pp. 395–427. Pergamon Press, 1977. Printed in Great Britain

STRATIGRAPHIC ANALYSIS SYSTEM: SAS

BRIAN R. SHAW
Department of Geology, Syracuse University, Syracuse, NY 13210, U.S.A.

RICHARD SIMMS
Department of Computer Sciences, University of Michigan, Ann Arbor, MI 48104, U.S.A.

(*Received* 30 *November* 1976)

Abstract—Stratigraphic Analysis System (SAS) is an on-line, interactive data-base analysis system designed for use in a subsurface laboratory. The program is written in FORTRAN and ALGOL W and presently runs under the Michigan Terminal System at the University of Michigan.

The SAS system was designed to overcome several problems in geological data-base systems. Both data discontinuities and substring indexing have been considered as well as three-dimensional location of information.

The system consists of four procedures; the command processor, the user aid package, the data-set loader and general data processors. The data set is composed of hierarchical records in a one-dimensional array which consists of logical flags to index an internal dictionary.

Presently output contains well listings, well displays, data editing and data search capabilities.

Key Words: Stratigraphy, Data system, Subsurface geology.

INTRODUCTION

The design of a subsurface data-base system in geology must include features that take into account special constraints imposed by the nature of geological data.

Three initial problems of substring indexing, location assignment in three dimensions, and missing information are all factors that should be incorporated into a system design. Substring indexing is necessary due to the diverse nature of geologic data. The data set should contain lithologic, stratigraphic, and paleontologic data but not all of this information is used at any one time. It is necessary therefore to be able to index a portion of the information record. Geologic information is almost always (or always, for subsurface information) at a depth or elevation involving the assignment of information to a location in three-dimensional space. The last problem, that of missing information, is a difficult one to approach. There are two types of discontinuities; geologic and data imposed. The difference between them is crucial; one implies a lack of data due to geological processes, the other due to sampling difficulties. These must be clearly separated in the data set.

SAS

Stratigraphic Analysis System (SAS) is an on-line interactive computer information system designed to handle subsurface well data. The system allows the user to examine, modify, and manipulate a data set which is organized to reflect the described problems. SAS was intended for use at the University of Michigan Subsurface Laboratory and is based in part on portions of a stratigraphic analysis system developed by Mosher (1963).

SAS is written primarily in the ALGOL W language, created by Stanford University as an extension of the ALGOL 60 language and modified by the University of Newcastle-Upon-Tyne to run under the Michigan Terminal System. The Michigan Terminal System (MTS) is a large resident, reentrant operating subsystem developed by the computing center staff at the University (Univ. Mich. Comp. Ctr., 1971, 1974). ALGOL W was chosen because it lends itself to structured programming. Input–output limitations in ALGOL W were supplemented by externally defined FORTRAN subroutines which allow SAS to use multiple logical input–output units.

The SAS program and its data set are kept on disk files and are run on the University of Michigan's AMDAHL 470 V/6 computer under the control of MTS in conversational mode from any terminal or teletype, or in nonconversational batch mode.

PROGRAM STRUCTURE

The SAS program is composed of several separate internally and externally defined ALGOL W procedures and FORTRAN subroutines which each have a distinct function (Fig. 1). The most important procedures are the command processor, the user aid package, the data set loader, and the general data processors.

The command processor serves as the interface between the user and SAS. The command processor prompts the user to enter commands, checks the commands for syntactical validity, and then invokes the appropriate procedure to satisfy the command request. An invalid command causes the command processor to print an appropriate error message.

One of the special features of the SAS system is the user-aid package. The user-aid package serves the role of an on-line counsellor. It is easy for the user to forget some of the available commands and their functions, as well as becoming frustrated or confused. The user-aid package monitors the progress of the user. Upon request it will explain why a previous command was invalid, and it can produce a list of all avaliable commands as well as explain the function of a specific command.

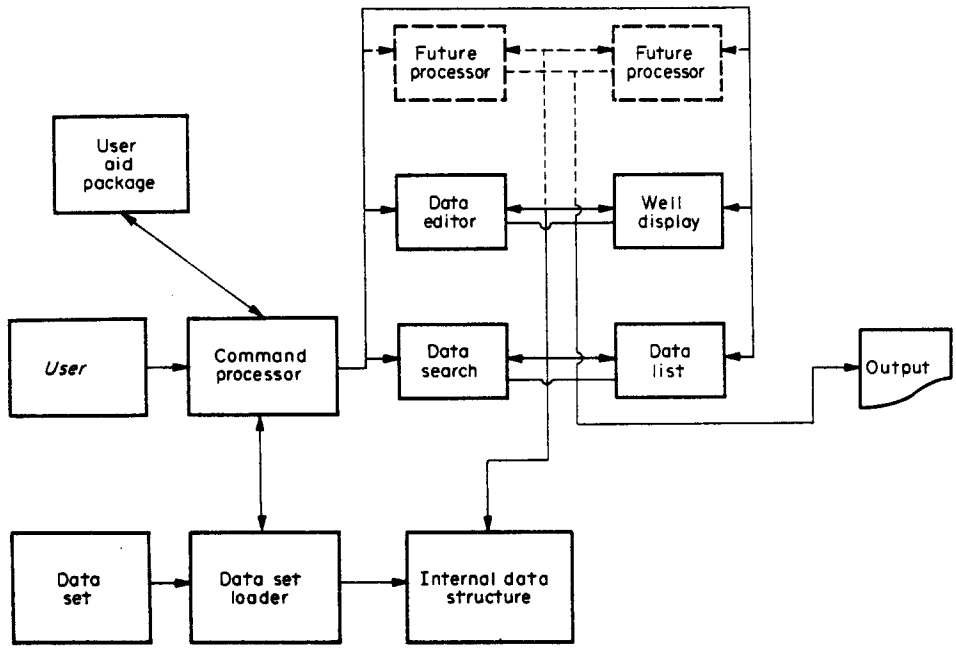


Figure 1. Basic structure of SAS.

The data-set loader procedure which is invoked via the command processor, loads the subsurface data out of a disk file or a magnetic tape into central memory, where the data are indexed and linked together to provide rapid accessing times. Only part of the data set is in central memory at any given time as the computer employs a virtual memory system. Part of the data-set loader was written in FORTRAN to overcome ALGOL W's inability to read data from more than one logical input–output unit. The FORTRAN subroutines allow SAS to input information from the user's terminal as well as data and information from various disk files.

The data processors are general procedures that do the work of manipulating information in the data set. There are several processors with distinct functions, each invoked by appropriate commands to the command processor. There are processors currently to generate cross sections, well and stratigraphic interval listings, as well as procedures to scan and modify the data set. This part of the system is readily expandable. New procedures written in ALGOL W, FORTRAN, or any other language which employ standard IBM S-type linkages (Univ. Mich. Comp. Ctr., 1972) can be appended to the system with an appropriate command addition to the command processor.

DATA SET

The data set of SAS follows a convenient and flexible format which has several important features. The most important aspect of the data format is the telescoping of descriptive information into a specified interval within a well as opposed to a stratigraphic 'top'.

A well can be separated into stratigraphic intervals (Fig. 2) which can be subdivided further into separate lithologic intervals. The data set reflects this natural subdivision of information in the form of three distinct

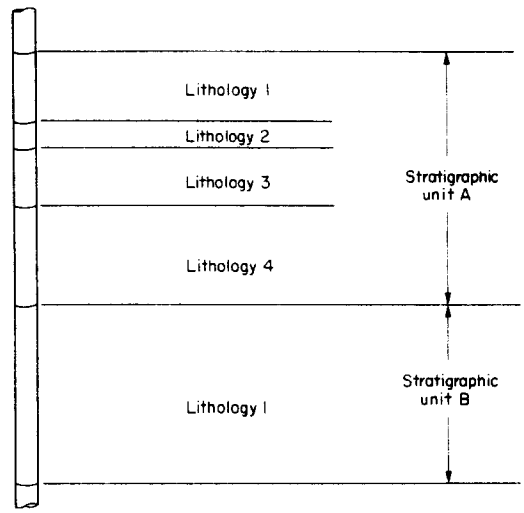


Figure 2. Interval assignment of information in SAS.

hierarchical types of records. In the data set a well is defined by: (1) a well ID record (Fig. 3), which contains data elements describing the name, number, location and elevation of the well; (2) stratigraphic records (Fig. 3) corresponding to each stratigraphic interval in the well; and (3) lithologic records (Fig. 4) to correspond to each lithologic interval within a stratigraphic interval. Each well definition in the data set therefore contains only one well ID record and as many stratigraphic and lithologic records necessary to define the well. Stratigraphic records contain data elements describing the name and boundaries of each stratigraphic interval in the well, and the lithologic records contain the bulk of the physical information in the well, that is data elements describing the physical interval boundaries, lithologies, textural descriptions, fossil records, minerals, and diagenetic,

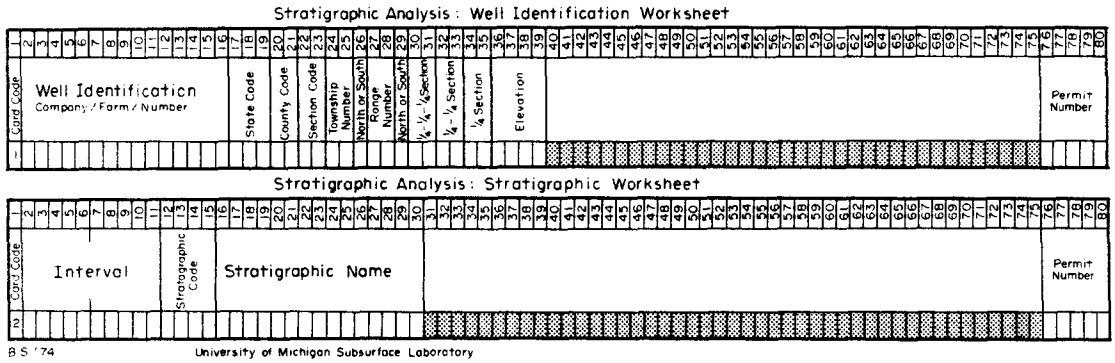


Figure 3. Well ID record worksheet and stratigraphic record worksheet.

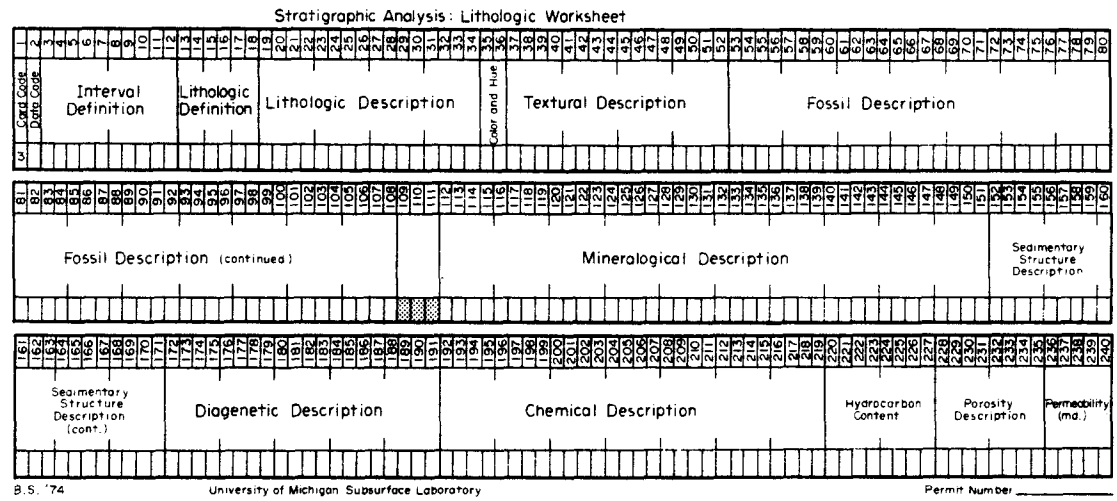


Figure 4. Lithologic record worksheet.

chemical, and other descriptive information such as permeability and porosity within any lithologic interval.

Stratigraphic and lithologic intervals are assigned arbitrarily to a well by the geologist. The applications of stratigraphic information being assigned to an interval as opposed to a horizon (stratigraphic 'top'), then creates a unique situation in which to accommodate discontinuities.

The major types of discontinuities encountered in subsurface analysis are produced by geologic gaps, such as disconformities, unconformities, faults, and nongeologic gaps such as missing data (Fig. 5). The data set of SAS does not require recognition of these features prior to analysis, but instead, an interval that contains no data (such as samples not collected) remains a valid interval in SAS. Missing stratigraphic information due to a discontinuity or fault is not assigned an interval, and the resulting juxtaposition of noncontiguous stratigraphic intervals identifies the discontinuity. Determining the nature of such a break is left to the geologist.

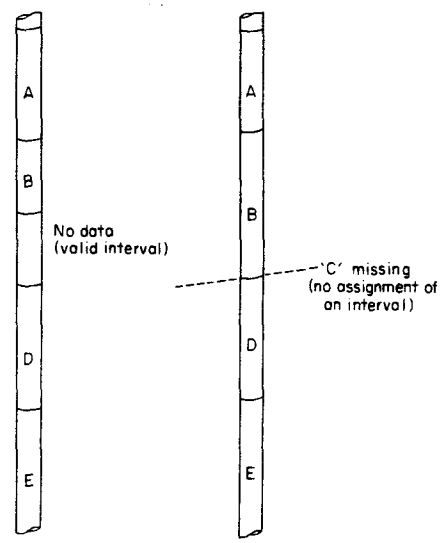


Figure 5. Discontinuities in SAS.

INTERNAL REPRESENTATION OF THE DATA SET

SAS loads the data set into central memory of the computer via the data-set loader. In central memory the records of the data set are kept in a large one-dimensional array. To index the array a list of tree-like structures are generated. Each tree structure represents a single well, and each leaf or node of a tree is an index of a specific record of the well.

The process of loading and indexing the records in the data set can be shown by considering two hypothetical

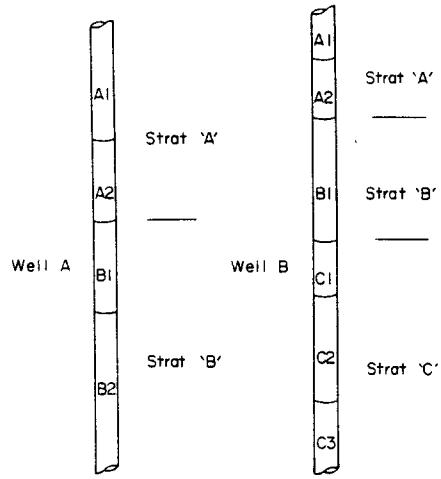


Figure 6. Hypothetical wells.

wells in the data set, well A and well B (Fig. 6). Well A contains two stratigraphic intervals, each of which contains two lithologic intervals. The second well, well B, contains three stratigraphic intervals separated into two, one, and three lithologic intervals respectively. In the data set, well A is described with one well ID record, two stratigraphic records and three lithologic records. Likewise, well B needs one well ID record, three stratigraphic records, and six lithologic records. The internal indexing structure (Fig. 7) for these two wells then is composed of two three-level trees corresponding to the two wells. Note that the tree structures do not contain geologic information, but flags which index the large array with all the physical information. The purpose of storing the data in an array and indexing it with a tree structure is to allow sequential operations on the data as well as specific subsets of it.

CONSTRUCTION OF THE DATA SET

Accurate, uniform data are essential to the performance of any system. The data obtained from a well must be encoded into the three types of records comprising the data set. To establish uniformity, a data dictionary developed from Briggs and Briggs (1971) containing geographic, stratigraphic, lithologic, and other descriptive codes is used to construct records for the data set. However, the use of a dictionary has both advantages and disadvantages. The need to refer to a dictionary of codes when building the data set can be clumsy for occasional usage. In spite of this there are advantages which make use of this reference desirable. Standardized descriptions in geologic information systems are necessary if the system is to be used by many people with varied backgrounds. Subjective terms such as 'some', 'slightly', or 'occasionally' can be interpreted differently. For this reason only extremes or metric information are incorporated into the dictionary. Another advantage of encoding data is the decrease in physical size of the data set.

Specifically designed coding worksheets facilitates the encoding process (Figs. 3 and 4). Initial well data encoded on these forms can later be keypunched onto cards and then stored permanently on disk or magnetic tape. The information is stored in an internal dictionary in the data set loader, the codes on the cards representing the logical flags indexing the information in the dictionary.

USING THE SYSTEM

Using SAS involves initial execution of the program, then issuing a series of commands to the system to operate upon the data as desired. The command language is divided into two modes of operation: the SAS mode, and the EXAMINE mode. When the user is in SAS mode, he is able to issue commands which apply to all the wells in the data set, while in EXAMINE mode, commands apply only to a specific well. The purpose of a double-moded command language is to simplify the issuing of commands and to organize the commands into logical groups.

The user is initially in the SAS mode. To enter the

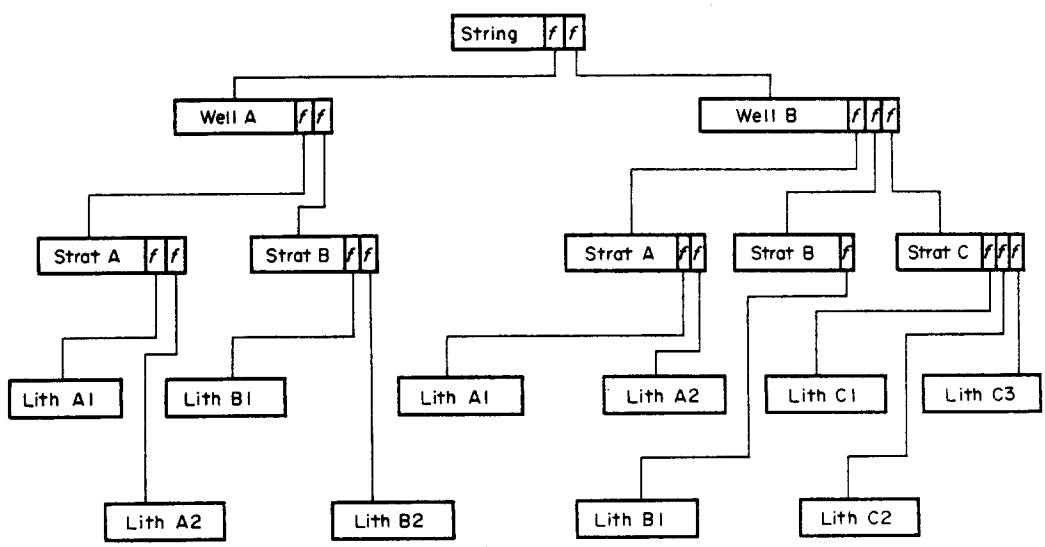


Figure 7. Internal representation of indexing structure in SAS.

EXAMINE mode, an 'EXAMINE' command is issued to inform the system that subsequent commands will apply only to the well which is addressed. The 'SAS' command is given to reenter SAS command mode. The user is able to distinguish which mode he is currently in by the prefix character generated by the command processor for prompting purposes. A "+" indicates SAS mode, whereas a ":" signals EXAMINE mode (see Appendix I).

CONCLUSIONS

SAS is a new and young system. Its output functions and data manipulations are limited in number. What is important is that a framework has been developed and new data processors now can be added easily to the system. These processors could preform the functions of mapping, dynamic well creation, and statistical operations. Applications could extend beyond subsurface information to include other types of geologic information such as surficial geology.

SAS provides the geologist with a useful tool to enable him to simplify the processing and understanding of large amounts of information.

Acknowledgments—This work was begun initially as an investigation under Dr. L. I. Briggs at The University of Michigan. His expertise and comments have added greatly to the quality of the system. Funds for the system were provided by the Northern Michigan Project for 1974 administered by Dr. Briggs. The computing was done at The University of Michigan Computing Center.

REFERENCES

Briggs, L. I., and Briggs, D. Z., 1971, Thesaurus for geologic document analysis and information system (DIAS): Geoscience Documentation, v. 3, no. 3–4, p. 76–88.

Mosher, F., 1963, A computer oriented system in stratigraphic analysis: Inst. Sci. Tech. 66221-1-T, Univ. Michigan, 32 p.

University of Michigan Computing Center, 1971, The Michigan Terminal System, vol. 1: MTS and the Computing Center. Univ. Michigan, Ann Arbor, Michigan, p. 47–64.

University of Michigan Computing Center, 1972, ALGOL W in MTS: Computing Center Memo 199, Univ. Michigan, p. 13–14.

University of Michigan Computing Center, 1974, The Michigan Terminal System, vol. 5: System Services, Univ. Michigan, p. 191–202.

APPENDIX I

The sample run presented here was preformed in 1974 when the system was being tested. The well displayed ('Test Well') is a hypothetical well. The coordinates and geology are imaginary.

The well contains 6 stratigraphic units, labelled A–F, and is located in Maracaibo county, Hawaii. There are 4 runs of the program displaying all or parts of the well. In the middle of the output is an example search and example error response.

The first display is of stratigraphic unit 'C'. All well displays contain unit labels, symbolic display, footages, a brief lithic description, and relative percentages of the lithic elements.

The second display is of the entire well, this time at a scale of 1 in. = 200 ft. After the display there are a series of MTS commands ("#" prefix) followed by a data search of wells in the system at the time. After this the same well is displayed at various scales.

```
**************************

  COORDINATES:  NWNE 32-21N 6W

  STATE:  HAWAII

  COUNTY:  MARACAIBO

  ELEVATION:    742  PERMIT NUMBER:  11705

  SCALE FACTOR:  1 INCH = 6 LINES =   20  FEET


STRATIGRAPHIC    CROSS-
   UNIT          SECTION      DEPTH      LITHOLOGY
-------------------------------------------------------------------------------
 STRAT_C       |CCCCFMOOOO| -   350  CALCAREOUS DOLOMITE            42.75000  %
               |CCCCFMOOOO|          FOSSILIFEROUS MICRITE         13.75000  %
               |CCCCFMOOOO|          MICRITE & DISMICRITE          13.75000  %
               |CCCCFMOOOO|          CTHER                         29.75000  %
               |CCCCFMOOOO|
               |CCCCFMOOOO|
               |CCCCFMOOOO|
               |CCCCFMOOOO|
               |CCCCFMOOOO|
               |CCCCFMOOOO|
               |CCCCFMOOOO|
               |CCCCFMOOOO|
               |CCCCFMOOOO|
               |CCCCFMOOOO|
               |CCCCFMOOOO|
               |CCCCFMOOOO|
               |CCCCFMOOOO|
               |OOCOOCOOOO| -   400

  END OF CROSS-SECTION.

: DIS SCALE=200

STRATIGRAPHIC    CROSS-
   UNIT          SECTION      DEPTH      LITHOLOGY
-------------------------------------------------------------------------------
 STRAT_A       |LLLLCCLLPO| -   100  LITHIC ARENITE               40.61249  %
               |LLLLCCLLPO|          CHERT                        13.94998  %
               |LLLLCCLLPO|          LIMESTONE                    16.36249  %
               |LLLLCCLLPO|          POORLY WASHED BIOSPARITE     8.122498  %
               |LLLLCCLLPO|          FELDSPATHIC GREYWACKE        2.042500  %
               |LLLLCCLLPO|          ANHYDRITE                    1.282499  %
               |LLLLCCLLPO|          CTHER                        9.607559  %
 STRAT_B       |AAAAAAAAAO| -   200  ACIDIC VOLCANIC              90.25000  %
               |AAAAAAAAAO|          CTHER                        9.750000  %
               |AAAAAAAAAO|
               |AAAAAAAAAO|


 STRAT_C       |CCCCFMOOOO| -   350  CALCAREOUS DOLOMITE          42.75000  %
               |CCCCFMOOOO|          FOSSILIFEROUS MICRITE        13.75000  %
               |CCCCFMOOOO|          MICRITE & DISMICRITE         13.75000  %
               |CCCCFMOOOO|          CTHER                        29.75000  %
 STRAT_D       |CCCCCCCCCO| -   400  COAL                         90.25000  %
```

```
                |CCCCCCCCCO|          OTHER                          9.750000  %
                |CCCCCCCCCO|
                |CCCCCCCCCO|
                |CCCCCCCCCO|
                |CCCCCCCCCO|
   STRAT_E      |QQQQQBBBBO| -   600  QUARTZ ARENITE                52.25000  %
                |QQQQQBBBBO|          BIOLITHITE                    42.75000  %
                |QQQQQBBBBO|          OTHER                         5.000000  %
                |QQQQQBBBBO|
                |QQQQQBBBBO|
                |QQQQQBBBBO|
                |QQQQQBBBBO|
   STRAT_F      |SSSSSSSSSO| -   850  SALT                          90.25000  %
                |SSSSSSSSSO|          OTHER                         9.750000  %
                |SSSSSSSSSO|
                |SSSSSSSSSO|
                |OOOOOOOOOO| -  1000

     END OF CROSS-SECTION.

: SAS
+ MTS
# $SOU PREVIOUS
# $C *SOURCE*@SP *PRINT*
> *PRINT* ASSIGNED RECEIPT NUMBER 644625
# *PRINT* 644625 RELEASED, 6 PAGES
# SOU SAS.GO
# $COMMENT:  SIT BACK AND RELAX UNTIL YOU SEE "EXECUTION OF SAS BEGINS:"
# $R SAS.OBJ 5=SAS.DATA 3=*MSOURCE* SCARDS=*MSOURCE* T=5
# EXECUTION BEGINS


  EXECUTION OF SAS BEGINS:
+ LOAD

   NUMBER OF INPUT RECORDS:     85
   NUMBER OF WELLS DETECTED:     6

+ WELLS

     WELL NAME          PERMIT NUMBER      RECORD #
  ------------------------------------------------

    |---WELL A----|         12345              1
    WELL B                  54321              4
    WELL C                  11111              8
    TEST WELL               11705             15
    SUPERIOR ANDROS         00001             29
    CONSUMERS-RUFFI         24179             54

   END OF WELLS.

+ EXAMINE TEST.WELL
  *** WELL: 'TEST.WELL       '  DOES NOT EXIST.
+ EXPLAIN
*
*    *4* THE WELL YOU SPECIFIED ON THE 'EXAMINE' COMMAND
*    DOES NOT NOT EXACTLY MATCH ANY OF THE WELLS THAT WERE
*    IN YOUR DATA.  REMEMBER THAT A WELL NAME
*    CONSISTS OF A COMPANY,FARM,AND WELL NUMBER IN
*    ABBREVIATED FORM. IF YOU CAN'T REMEMBER THE EXACT
*    NAME OF A WELL, USE IT'S PERMIT NUMBER INSTEAD.
*    FOR EXAMPLE USE:
*              EXAMINE   21677
*    INSTEAD OF:
*              EXAMINE WRONGWELLNAME
*
+ EXAMINE TEST WELL
: UNITS
     UNIT           TOP     BOTTOM  RECORD #
  ----------------------------------------
    STRAT_A          100      200       16
    STRAT_B          200      350       19
    STRAT_C          350      400       21
    STRAT_D          400      600       23
    STRAT_E          600      850       25
    STRAT_F          850     1000       27

   END OF UNITS.
```

```
: DISPLAY@TTL SCALE=150

    ****************************
    *                          *
    *     TEST WELL            *
    *                          *
    ****************************

      COORDINATES:  NWNE 32-21N 6W

      STATE:  HAWAII

      COUNTY:  MARACAIBO

      ELEVATION:    742  PERMIT NUMBER:  11705

      SCALE FACTOR:  1 INCH = 6 LINES =  150  FEET
```

STRATIGRAPHIC UNIT	CROSS-SECTION	DEPTH	LITHOLOGY		
STRAT_A	|LLLLCCLLPO|	- 100	LITHIC ARENITE	40.61249	%
	|LLLLCCLLPO|		CHERT	19.94998	%
	|LLLLCCLLPO|		LIMESTONE	18.38249	%
	|LLLLCCLLPO|		POORLY WASHED BIOSPARITE	8.122498	%
	|LLLLCCLLPO|		FELDSPATHIC GREYWACKE	2.062500	%
	|LLLLCCLLPO|		ANHYDRITE	1.282499	%
	|LLLLCCLLPO|		OTHER	9.607559	%
STRAT_B	|AAAAAAAAAO|	- 200	ACIDIC VOLCANIC	90.25000	%
	|AAAAAAAAAO|		OTHER	9.750000	%
	|AAAAAAAAAO|				
	|AAAAAAAAAO|				
	|AAAAAAAAAO|				
	|AAAAAAAAAO|				
STRAT_C	|CCCCFMCOOO|	- 350	CALCAREOUS DOLOMITE	42.75000	%
	|CCCCFMCOOO|		FOSSILIFEROUS MICRITE	13.75000	%
	|CCCCFMCOOO|		MICRITE & DISMICRITE	13.75000	%
	|CCCCFMCOOO|		OTHER	29.75000	%
STRAT_D	|CCCCCCCCCO|	- 400	COAL	90.25000	%
	|CCCCCCCCCO|		OTHER	9.750000	%
	|CCCCCCCCCO|				
	|CCCCCCCCCO|				
	|CCCCCCCCCO|				
	|CCCCCCCCCO|				
	|CCCCCCCCCO|				
STRAT_E	|QQQQQEBBBO|	- 600	QUARTZ ARENITE	52.25000	%
	|QQQQQEBBBO|		BIOLITHITE	42.75000	%
	|QQQQQEBBBO|		OTHER	5.000000	%
	|QQQQQEBBBO|				
	|QQQQQEBBBO|				
	|QQQQQEBBBO|				
	|QQQQQEBBBO|				
	|QQQQQEBBBO|				
	|QQQQQEBBBO|				
STRAT_F	|SSSSSSSSSO|	- 850	SALT	90.25000	%
	|SSSSSSSSSO|		OTHER	9.750000	%
	|SSSSSSSSSO|				
	|SSSSSSSSSO|				
	|SSSSSSSSSO|				
	|SSSSSSSSSO|				
	|OOOOOOOOOO|	- 1000			

```
    END OF CROSS-SECTION.

: SAS
+ COMMENT: LETS GET A HARD COPY OF THIS
+ RTS
# $SOU PREVIOUS
# CONTROL *PRINT* HOLD PRINT=IN
# *PRINT* ASSIGNED RECEIPT NUMBER 044629
# SC *SOURCE*@SP *PRINT*
# RES
+ EXAMINE TEST WELL
: DISPLAY STRAT_C SCALE=50
```

```
STRATIGRAPHIC    CROSS-
    UNIT         SECTION      DEPTH       LITHOLOGY
-------------------------------------------------------------------------------
 STRAT_C       |CCCCFMCCOO| -    350  CALCAREOUS DOLOMITE              42.75000  %
               |CCCCFMCOOO|           FOSSILIFEROUS MICRITE          13.75000  %
               |CCCCFMCOOO|           MICRITE & DISMICRITE           13.75000  %
               |CCCCFMOOOC|           CTHER                          29.75000  %
               |CCCCFMCOOO|
               |CCCCFMCOOO|
               |CCCCFMOOOO|


               |CCCCFMOOOC|
               |CCCCFMCOOO|
               |CCCCFMOOOO|
               |CCCCFMCOOO|
               |CCCCFMOOOC|
               |CCCCFMOOOO|
               |CCCCFMOOOO|
               |CCCCFMCOOC|
               |OCCOOCOOCO| -    400

   END OF CROSS-SECTION.

: SAS
+ STOP

  EXECUTION OF SAS IS TERMINATED


  (( ).3) SECONDS IN EXECUTION
# EXECUTION TERMINATED
# C *SOURCE*@SP -I
```

APPENDIX II

Appendix II contains the complete program listing. Only the lithologic portion of the internal dictionary was reproduced.

```
$LIST 2JAB:CARLA
      1     BEGIN
      2
      3        COMMENT  /*  THIS IS THE MAIN PROGRAM. A
      4                     SIMPLE OPERATING SYSTEM IS USED
      5                     USED TO MANIPULATE WELL DATA.      */;
      6
      7        STRING(80) COMMAND;
      8        STRING(240) BUFFER;
      9
     10        LOGICAL FLAG,ERROR;
     11
     12        INTEGER N_RECORDS,N_WELLS,WELL,TOKEN,VAL,ERR_N;
     13        INTEGER I,J;
     14
     15        STRING(240) ARRAY RAW_DATA(1::29);
     16        STRING(4) ARRAY TEMP(1::60);
     17        STRING(15) ARRAY WELL_ID(1::21);
     18        STRING(5)  ARRAY WELL_EN(1::21);
     19        INTEGER ARRAY WELL_PTR(1::21);
     20
     21
     22        PROCEDURE GET (STRING(4) ARRAY A(*); LOGICAL RESULT B);
     23           FORTRAN "INPUT";
     24
     25
     26        PROCEDURE EXPLAIN (STRING(80) VALUE C;INTEGER VALUE T,N);
     27           ALGOL "EXPLA001";
     28
     29
     30        PROCEDURE SETPFX(STRING(1) VALUE P; INTEGER VALUE LEN);
     31           FORTRAN "SETPFX";
     32
     33
     34        PROCEDURE MTS;
     35           FORTRAN "MTS";
     36
     37
     38        PROCEDURE MTSCMD(STRING(9) VALUE A; INTEGER VALUE B);
     39           FORTRAN "MTSCMD";
     40
     41
     42        PROCEDURE DELLANNO (STRING(80) VALUE A; INTEGER VALUE RESULT N,C);
     43           ALGOL "DELLA001";
     44
     45
     46        PROCEDURE CLOCK (STRING(80) VALUE A);
     47           ALGOL "CLOCK001";
     48
     49
     50     PROCEDURE DISPLAY
     51        (STRING(80) VALUE COMMAND;INTEGER VALUE WELL;INTEGER ARRAY
     52        WELL_PTR(*);STRING(240) ARRAY RAW_DATA(*);INTEGER VALUE RESULT
     53        ERR_N);
     54        ALGOL "DISPL001";
     55
     56
     57        PROCEDURE READ_RECORD;
     58
     59           BEGIN
     60              GET(TEMP,FLAG);
     61              ERROR:=FLAG THEN "READ ERROR:" " FLAG
     62              FOR I:=1 UNTIL 60 DO
     63                 BEGIN BUFFER(I*4-4|4):=TEMP(I) END;
     64           END READ_RECORD;
     65
     66
     67
     68        INTEGER PROCEDURE STI(INTEGER VALUE RESULT I);
     69           BEGIN
     70              COMMENT: THIS PROCEDURE CONVERTS A STRING
     71                       TO A NUMBER.
     72              INTEGER VAL,J;
     73              J:=VAL:=0;
```

```
74              WHILE (COMMAND(I|1)¬=" ") AND (J<10) DO
75                 BEGIN
76                    IF(COMMAND(I|1)<"0")OR(COMMAND(I|1)>"9")THEN
77                       BEGIN  SETPFX(" ",1);
78                          WRITE("*** ILLEGAL CHARACTER: '",COMMAND(I|1),"'");
79                          WRITE("    DETECTED IN NUMERIC CONSTANT.");
80                             IOCONTROL(2);
81                          ERR_N:=11;  ERROR:=TRUE; GOTO RETURN;
82                       END;
83                    VAL:=VAL*10+(DECODE(COMMAND(I|1))-240);
84                    I:=I+1; J:=J+1;
85                 END;
86              IF J=10 THEN
87                 BEGIN  SETPFX(" ",1);
88                    WRITE("*** NUMERIC CONSTANT TOO LARGE TO CONVERT");
89                 IOCONTROL(2);
90                    ERR_N:=12; ERROR:=FALSE; GOTO RETURN;
91                 END;
92              RETURN:
93              VAL
94           END STI;
95
96
97
98
99        PROCEDURE LIST;
100
101          BEGIN
102             COMMENT:  PRODUCE LISTING OF ALL WELLS AND
103                       THEIR PERMIT NUMBERS.         ;
104             INTEGER I;
105
106             SETPFX(" ",1); INTFIELDSIZE:=5;
107             IF N_WELLS=0 THEN
108                  BEGIN
109                       WRITE(" ");
110                       WRITE(" NO WELLS EXIST.");
111                       GOTO RETURN;
112                  END;
113             WRITE(" ");
114             WRITE("   WELL NAME          PERMIT NUMBER     RECORD #");
115             WRITE("-----------------------------------------------");
116             WRITE(" ");
117             FOR I:=1 UNTIL N_WELLS DO
118                  BEGIN
119                       WRITE("  ",WELL_ID(I),"        ",WELL_PN(I),
120                        "            ",WELL_PTR(I));
121                  END;
122             WRITE(" "); WRITE(" END OF WELLS.");
123             RETURN:
124             WRITE(" ");
125             IOCONTROL(2);
126
127          END LIST;
128
129
130
131
132
133
134   PROCEDURE PRINT;WRITE("PRINT");
135   PROCEDURE SCAN;WRITE("SCAN");
136   PROCEDURE ALTER;WRITE("ALTER");
137   PROCEDURE SCAN_DATA;WRITE("SCAN_DATA");
138   PROCEDURE EXECUTE;WRITE("EXECUTE");
139
140
141      PROCEDURE LOAD_FILE;
142
143         BEGIN
144            COMMENT      /*   DATA IS READ VIA A FORTRAN
145                              SUBROUTINE FROM LOGICAL UNIT 5 */;
146
147            INTEGER I,J; I:=J:=0;
148            READ_RECORD;
149            WHILE FLAG DO
150               BEGIN
151                  I:=I+1;
152                  IF I>29 THEN
```

```
153                     BEGIN  SETPFX(" ",1);
154                        WRITE ("*** INTERNAL STORAGE LIMIT EXCEEDED.");
155                        WRITE ("*** LOADING OPERATION HALTED.");
156                        FLAG:=FALSE; ERR_N:=1;
157                     END
158                  ELSE
159                     BEGIN
160                        RAW_DATA(I):=BUFFER;
161                        IF BUFFER(0|1)="1" THEN
162                          BEGIN  COMMENT  /* WELL I.D. DETECTED */;
163                             J:=J+1;
164                             IF J>21 THEN
165                                BEGIN  SETPFX(" ",1);
166                                   WRITE ("*** WELL LIMIT EXCEEDED.");
167                                   WRITE ("*** LOADING OPERATION HALTED.")
168                                   FLAG:=FALSE; ERR_N:=2;
169                                END
170                             ELSE
171                                BEGIN
172                                   WELL_ID(J):=BUFFER(1|15);
173                                   WELL_PN(J):=BUFFER(75|5);
174                                   WELL_PTR(J):=I;
175                                END;
176                          END;
177                     END;
178                  READ_RECORD;
179               END;
180            WELL_PTR(J+1):=I+1;
181            N_RECORDS:=I; N_WELLS:=J;
182            COMMENT   /* OUTPUT LOAD RESULTS  */;
183            SETPFX(" ",1);
184            INTFIELDSIZE:=5;
185            WRITE (" ");
186            WRITE (" NUMBER OF INPUT RECORDS: ",I);
187            WRITE (" NUMBER OF WELLS DETECTED:",J);
188            WRITE (" ");
189            IOCONTROL(2);
190
191         END LOAD_FILE;
192
193
194
195      PROCEDURE PRINT_RAW_DATA;
196
197         BEGIN  INTEGER START,STOP; I:=1;
198            IF N_WELLS=0 THEN
199               BEGIN SETPFX(" ",1);
200                  WRITE(" ");
201                  WRITE(" THERE IS NOTHING TO PRINT.");
202                  GOTO RETURN;
203               END;
204            WHILE COMMAND(I|1)=" " DO I:=I+1;
205            WHILE (COMMAND(I|1)=" ")AND(I<79) DO I:=I+1;
206            IF I=79 THEN START:=STOP:=1 ELSE
207            BEGIN
208               START:=STI(I); IF ERROR THEN GOTO RETURN;
209               WHILE (COMMAND(I|1)=" ")AND(I<79) DO I:=I+1;
210               IF I=79 THEN STOP:=START ELSE
211               STOP:=STI(I); IF ERROR THEN GOTO RETURN;
212            END;
213            IF (START<1)OR(STOP>N_RECORDS) THEN
214               BEGIN SETPFX(" ",1);
215                  WRITE("ILLEGAL RECORD DESIGNATOR.");
216                  ERR_N:=15; GOTO RETURN;
217               END;
218            WRITE(" "); SETPFX(" ",1);
219            FOR I:=START UNTIL STOP DO
220               BEGIN
221                  WRITE("  RECORD ",I); WRITE(" ");
222                  WRITE("COLUMNS                    'RAW' DATA");
223                  WRITE("-"); FOR J:=1 UNTIL 70 DO WRITEON("-");
224                  WRITE("  1- 60: |",RAW_DATA(I)(0|60),"|");
225                  WRITE(" 61-120: |",RAW_DATA(I)(60|60),"|");
226                  WRITE("121-180: |",RAW_DATA(I)(120|60),"|");
227                  WRITE("181-240: |",RAW_DATA(I)(180|60),"|");
228                  WRITE(" "); WRITE(" ");
229               END; WRITE(" "); IOCONTROL(2);
230            RETURN:
231
```

```
232          END PRINT_RAW_DATA;
233
234
235       PROCEDURE PRINT_STRAT_UNITS;
236
237          BEGIN INTEGER FIRST,LAST;
238             FIRST:=WELL_PTR(WELL);
239             LAST:=WELL_PTR(WELL+1)-1;
240             FLAG:=TRUE; SETPFX(" ",1);
241             INTFIELDSIZE:=5;
242             WRITE(" ");
243             WRITE("    UNIT          TOP    BOTTOM  RECORD #");
244             WRITE("-----------------------------------------");
245             FOR I:=FIRST UNTIL LAST DO
246                BEGIN
247                   IF RAW_DATA(I)(0|1)="2" THEN
248                   BEGIN
249                      WRITE("  ",RAW_DATA(I)(1|12),
250                            "   ",RAW_DATA(I)(13|5),
251                            "   ",RAW_DATA(I)(18|5),"   ",I);
252                      FLAG:=FALSE;
253                   END;
254                END;
255                   WRITE(" ");
256             IF FLAG THEN WRITE(" THERE ARE NO STRATIGRAPHIC UNITS.") ELSE
257                   WRITE(" END OF UNITS.");
258             WRITE(" "); IOCONTROL(2);
259
260          END PRINT_STRAT_UNITS;
261
262
263
264       PROCEDURE EXAMINE;
265
266          BEGIN
267
268
269
270             COMMENT  /*  THIS PROCEDURE HANDLES
271                          DISPLAYING, SCANNING AND MODIFYING
272                          OF A SINGLE WELL          */;
273             GET_WELL(WELL);
274             IF ERROR THEN GO TO RETURN;
275             SETPFX(":",1);
276             IOCONTROL(2);
277             READCARD(COMMAND);
278             WHILE COMMAND(0|2)¬="SA" DO
279                BEGIN
280                   IF COMMAND(0|2)="DI" THEN DISPLAY
281                   (COMMAND,WELL,WELL_PTR,RAW_DATA,ERR_N)
282                   ELSE IF COMMAND(0|1)="*" THEN GOTO NEXT
283                   ELSE IF COMMAND(0|1)="S" THEN SCAN
284                   ELSE IF COMMAND(0|1)="A" THEN ALTER
285                   ELSE IF COMMAND(0|1)="P" THEN PRINT
286                   ELSE IF COMMAND(0|1)="U" THEN PRINT_STRAT_UNITS
287                   ELSE IF COMMAND(0|2)="CO" THEN GOTO NEXT
288                   ELSE IF COMMAND(0|2)="EX" THEN EXPLAIN(COMMAND,2,ERR_N)
289                   ELSE IF COMMAND(0|2)="EE" THEN EXPLAIN(COMMAND,2,ERR_N)
290                   ELSE
291                      BEGIN  SETPFX(" ",1);
292                         WRITE ("UNDECODABLE COMMAND");
293                         ERR_N:=9; IOCONTROL(2);
294                      END;
295                   NEXT:
296                   SETPFX(":",1);
297                   IOCONTROL(2);
298                   READCARD(COMMAND);
299                END;
300             RETURN:
301
302          END EXAMINE;
303
304
305
306       PROCEDURE GET_WELL(INTEGER RESULT A);
307
308          BEGIN
309
310             COMMENT    DETERMINE WHICH WELL IS TO BE
311                        EXAMINED.  ALSO CHECK TO SEE IF IT EXISTS.  ;
```

```
312             INTEGER I,J;
313             I:=0; ERROR:=FALSE;
314             WHILE COMMAND(I|1)¬=" " DO I:=I+1;
315             WHILE (COMMAND(I|1)=" ") AND (I<79) DO I:=I+1;
316             IF I=79 THEN
317                BEGIN  SETPFX(" ",1);
318                   WRITE("*** WELL NOT SPECIFIED.");
319                   ERROR:=TRUE; IOCONTROL(2); ERR_N:=3;
320                   GO TO RETURN
321                END;
322             IF (COMMAND(I|1)<"0") OR (COMMAND(I|1)>"9") THEN
323                BEGIN
324                   J:=1;
325                   WHILE (COMMAND(I|15)¬=WELL_ID(J)) AND
326                      (J<=N_WELLS) DO J:=J+1;
327                   IF J>N_WELLS THEN
328                      BEGIN  SETPFX(" ",1);
329                         WRITE ("*** WELL: '",COMMAND(I|15),
330                         "'  DOES NOT EXIST."); IOCONTROL(2);
331                         ERROR:=TRUE; ERR_N:=4; GOTO RETURN;
332                      END
333                   ELSE A:=J
334                END
335             ELSE
336                BEGIN COMMENT: SEARCH WELL_PN'S;
337                   J:=1;
338                   WHILE (COMMAND(I|5)¬=WELL_PN(J)) AND
339                      (J<=N_WELLS) DO J:=J+1;
340                   IF (J>N_WELLS) OR (COMMAND(I+5|1)¬=" ") THEN
341                      BEGIN  SETPFX(" ",1);
342                         WRITE ("*** WELL: '",COMMAND(I|5),
343                         "'  DOES NOT EXIST."); IOCONTROL(2);
344                         ERROR:=TRUE; ERR_N:=5; GOTO RETURN;
345                      END
346                   ELSE A:=J
347                END;
348             RETURN:
349
350          END GET_WELL;
351
352
353
354          COMMENT    /*    DECODE AND EXECUTE COMMANDS
355                           FROM USER.                    */;
356          INTEGER XXCNT; XXCNT:=0;
357          IOCONTROL(4);
358          WRITE("EXECUTION OF SAS BEGINS:");
359          IOCONTROL(2);
360          N_RECORDS:=N_WELLS:=0; ERR_N:=17;
361          SETPFX("+",1);
362          IOCONTROL(2);
363          READCARD(COMMAND);
364          WHILE COMMAND(0|2)¬="ST" DO
365             BEGIN
366                IF COMMAND(0|2)="LO" THEN LOAD_FILE
367                ELSE IF COMMAND(0|1)="R" THEN EXECUTE
368                ELSE IF COMMAND(0|1)="W" THEN LIST
369                ELSE IF COMMAND(0|1)="P" THEN PRINT_RAW_DATA
370                ELSE IF COMMAND(0|2)="SC" THEN SCAN_DATA
371                ELSE IF COMMAND(0|2)="MT" THEN MTS
372                ELSE IF COMMAND(0|2)="HE" THEN EXPLAIN(COMMAND,1,ERR_N)
373                ELSE IF COMMAND(0|2)="CO" THEN GOTO NEXT
374                ELSE IF COMMAND(0|3)="EXP" THEN EXPLAIN(COMMAND,1,ERR_N)
375                ELSE IF COMMAND(0|2)="EX" THEN EXAMINE
376                ELSE IF COMMAND(0|1)="*" THEN GOTO NEXT
377                ELSE IF COMMAND(0|3)="SIG" THEN MTSCMD("$SIGNOFF ",)
378                ELSE IF COMMAND(0|2)="DD" THEN DELLANNO(COMMAND,ERR_N,XXCNT)
379                ELSE IF COMMAND(0|2)="TI" THEN CLOCK(COMMAND)
380                ELSE
381                   BEGIN  SETPFX(" ",1);
382                      WRITE ("INVALID SAS COMMAND.");
383                      ERR_N:=10; IOCONTROL(2);
384                   END;
385                NEXT:
386                SETPFX("+",1);
387                IOCONTROL(2);
388                READCARD(COMMAND);
389             END;
390          SETPFX(" ",1);
```

```
   391         WRITE (" ");
   392         WRITE ("EXECUTION OF SAS IS TERMINATED");
   393         WRITE (" ");
   394       END.
END OF FILE
$SINK PREVIOUS

$LIST 2JAA:EXTERNAL(1,28)+(107)
     1      PROCEDURE CLOCK (STRING(80) VALUE A);
     2
     3        BEGIN
     4          COMMENT:  THIS IS  A PROCEDURE WHICH PROVIDES ACCESS
     5                    TO DIFFERENT TIMES.  FOR MORE INFORMATION
     6                    SEE "TIME" SUBROUTINE DESCRIPTION MTS VOL 3.
     7
     8          PROCEDURE TIME (INTEGER VALUE A,B);
     9            FORTRAN "TIME";
    10
    11          PROCEDURE SETPFX (STRING(1) VALUE A; INTEGER VALUE B);
    12            FORTRAN "SETPFX";
    13
    14          INTEGER I,J; I:=J:=0;
    15          WHILE A(I|1)¬=" " DO I:=I+1;
    16          WHILE (A(I|1)=" ") AND (I<79) DO I:=I+1;
    17          IF I=79 THEN J:=4 ELSE
    18            WHILE A(I|1)¬=" " DO
    19              BEGIN
    20                J:=J*10+(DECODE(A(I|1))-240);
    21                I:=I+1;
    22              END;
    23          IF (J<0) OR (J>11) THEN J:=4;
    24          SETPFX(" ",1); TIME(J,1);
    25
    26        END .
    27
    28
   107         PROCEDURE EXPLAIN(STRING(80) VALUE C; INTEGER VALUE T,N);
   108
   109            BEGIN
   110               COMMENT:  THIS IS AN EXTERNAL PROCEDURE TO HELP
   111                         THE CONFUSED USER.                    ;
   112
   113               PROCEDURE SETPFX(STRING(1) VALUE P; INTEGER VALUE B);
   114                  FORTRAN "SETPFX";
   115
   116      PROCEDURE MESSAGE;
   117        BEGIN
   118          INTEGER J;
   119          IF N<11 THEN
   120            BEGIN
   121               STRING(53) ARRAY ERR_TAB(1::48);
   122               INTEGER    ARRAY PTR    (1::10);
   123          PTR(1):=1;  PTR(2):=7;  PTR(3):=12;  PTR(4):=17;  PTR(5):=27;
   124          PTR(6):=30  PTR(7)  ;  PTR(8):=40  PTR(9):=44;  PTR(10   46;
   125
   126          ERR_TAB( 1):="*1* THIS ERROR IS NOT YOUR FAULT.  WHAT HAPPENED IS  ";
   127          ERR_TAB( 2):="THAT THE ARRAY WHICH HOLDS ALL YOUR DATA WAS NOT MADE";
   128          ERR_TAB( 3):="LARGE ENOUGH FOR YOU.  YOU HAVE TWO SOLUTIONS: 1) TRY";
   129          ERR_TAB( 4):="AGAIN WITH LESS DATA OR 2) RECOMPILE WHOLE PROGRAM   ";
   130          ERR_TAB( 5):="SETTING UPPER BOUND ON ARRAY 'RAW_DATA' TO HIGHER    ";
   131          ERR_TAB( 6):="VALUE.                                               ";
   132          ERR_TAB( 7):="*2* THIS ERROR IS NOT YOUR FAULT.  WHAT HAPPENED IS  ";
   133          ERR_TAB( 8):="THAT THE ARRAY THAT HOLDS POINTERS TO ALL YOUR WELLS ";
   134          ERR_TAB( 9):="WAS OVERFILLED.  YOU HAVE TWO SOLUTIONS: 1) TRY AGAIN";
   135          ERR_TAB(10):="WITH FEWER WELLS OR 2) RECOMPILE MAIN PROGRAM SETTING";
   136          ERR_TAB(11):="UPPER BOUND ON ARRAY 'WELL_PTR' TO A HIGHER VALUE.   ";
   137          ERR_TAB(12):="*3* TO EXAMINE A WELL, EITHER THE NAME OF THE WELL   ";
   138          ERR_TAB(13):="OR THE PERMIT NUMBER MUST BE SPECIFIED ON THE        ";
   139          ERR_TAB(14):="'EXAMINE' COMMAND.  YOU DIDN'T PUT EITHER ONE ON.  A ";
   140          ERR_TAB(15):="LEGAL COMMAND COULD BE:                              ";
   141          ERR_TAB(16):="        EXAMINE WELL2                                ";
   142          ERR_TAB(17):="*4* THE WELL YOU SPECIFIED ON THE 'EXAMINE' COMMAND  ";
   143          ERR_TAB(18):="DOES NOT NOT EXACTLY MATCH ANY OF THE WELLS THAT WERE";
   144          ERR_TAB(19):="IN YOUR DATA.  REMEMBER THAT A WELL NAME             ";
   145          ERR_TAB(20):="CONSISTS OF A COMPANY,FARM,AND WELL NUMBER IN        ";
   146          ERR_TAB(21):="ABBREVIATED FORM. IF YOU CAN'T REMEMBER THE EXACT    ";
   147          ERR_TAB(22):="NAME OF A WELL, USE IT'S PERMIT NUMBER INSTEAD.      ";
   148          ERR_TAB(23):="FOR EXAMPLE USE:                                     ";
   149          ERR_TAB(24):="          EXAMINE   21677                            ";
```

```
150       ERR_TAB(25):="INSTEAD OF:                                          ";
151       ERR_TAB(26):="         EXAMINE WRONGWELLNAME                       ";
152       ERR_TAB(27):="*5* THE WELL PERMIT NUMBER DOES NOT MATCH ANY OF      ";
153       ERR_TAB(28):="THE PERMIT NUMBERS THAT ARE IN YOUR DATA.  EITHER     ";
154       ERR_TAB(29):="CHECK YOUR DATA, OR TRY ANOTHER PERMIT NUMBER.        ";
155       ERR_TAB(30):="*6* THE MODIFIER YOU USED, NO MATTER HOW CORRECT      ";
156       ERR_TAB(31):="IT LOOKS, IS STILL ILLEGAL. LEGAL MODIFIERS ARE @NLI,";
157       ERR_TAB(32):="@NLO,@NCS,@NWN,@MIN,@POS,@CHM,@TXT,@SIR,@DIO,@POR,    ";
158       ERR_TAB(33):="@PER,@HUE,@HYD,@TTL,@PUL. FOR MORE INFORMATION ON     ";
159       ERR_TAB(34):="MODIFIERS, GIVE THE FOLLOWING COMMAND:                ";
160       ERR_TAB(35):="         EXPLAIN MODIFIERS                            ";
161       ERR_TAB(36):="*7* THE NAME OF THE STRATIGRAPHIC UNIT YOU SPECIFIED  ";
162       ERR_TAB(37):="ON THE DISPLAY COMMAND WAS TOO LONG. NAMES OF         ";
163       ERR_TAB(38):="STRATIGRAPHIC UNITS CAN ONLY BE UP TO 12 CHARACTERS   ";
164       ERR_TAB(39):="LONG.                                                 ";
165       ERR_TAB(40):="*8* THE STRATIGRAPHIC UNIT YOU SPECIFIED ON THE       ";
166       ERR_TAB(41):="DISPLAY COMMAND DOES NOT MATCH ANY OF THE UNITS IN    ";
167       ERR_TAB(42):="THE WELL YOU ARE EXAMINING.  YOU PROBABLY TYPED       ";
168       ERR_TAB(43):="IN THE WRONG CHARACTER SOMEWHERE.                     ";
169       ERR_TAB(44):="*9* THE COMMAND YOU ISSUED DOES NOT EXIST IN THE      ";
170       ERR_TAB(45):="'EXAMINE' MODE.  ALL COMMANDS MUST BEGIN IN COLUMN 1.";
171       ERR_TAB(46):="*10*  THE COMMAND YOU ISSUED IS NOT A SAS COMMAND.    ";
172       ERR_TAB(47):="ALL COMMANDS MUST BEGIN IN COLUMN 1.                  ";
173       ERR_TAB(48):="*11*  ON THE DISPLAY COMMAND, THE SCALE OF THE CROSS-";
174             J:=PTR(N);
175             WRITE("  ",ERR_TAB(J));
176             J:=J+1;
177             WHILE ERR_TAB(J)(0|1)¬="*" DO
178                BEGIN
179                   WRITE("  ",ERR_TAB(J));
180                   J:=J+1;
181                END;
182     END ELSE
183     IF N<20 THEN
184       BEGIN
185     STRING(53) ARRAY  ERR_TAB(48::92); INTEGER ARRAY PTR(11::20);
186        PTR(11):=48;  PTR(12):=53; PTR(13):=57; PTR(14):=63; PTR(15):=68;
187        PTR(16):=75; PTR(17):=76; PTR(18):=77; PTR(19):=83; PTR(20):=88;
188        ERR_TAB(48):="*11*  ON THE DISPLAY COMMAND, THE SCALE OF THE CROSS-";
189        ERR_TAB(49):="SECTION CAN BE SET USING THE 'SCALE=' PARAMETER. THE ";
190        ERR_TAB(50):="SCALE CAN ONLY BE SET TO A NUMERIC CONSTANT. THAT     ";
191        ERR_TAB(51):="MEANS A STRING OF UP TO 9 DIGITS, NOT CONTAINING ANY ";
192        ERR_TAB(52):="BLANKS OR NON-NUMERIC CHARACTERS.                     ";
193        ERR_TAB(53):="*12*  SINCE ALL INTEGER VALUES ARE STORED IN AN ARE   ";
194        ERR_TAB(54):="OF LIMITED SIZE, THERE IS A LIMIT ON THE SIZE OF      ";
195        ERR_TAB(55):="NUMERIC CONSTANTS THAT CAN BE USED. IN SAS THE LIMIT ";
196        ERR_TAB(56):="IS 9 CONSECUTIVE DIGITS FOR THE 'SCALE=' PARAMETER.   ";
197        ERR_TAB(57):="*13* THE VALUE SCALE IS SET TO, DETERMINES            ";
198        ERR_TAB(58):="THE NUMBER OF FEET THAT WILL BE REPRESENTED           ";
199        ERR_TAB(59):="BY ONE INCH OF THE PRINTED CROSS-SECTION.             ";
200        ERR_TAB(60):="AS THE VALUE OF SCALE => ZERO, THE LENGTH             ";
201        ERR_TAB(61):="OF THE CROSS-SECTION APPROACHES INFINITY             ";
202        ERR_TAB(62):="--SO ZERO IS AN ILLEGAL SCALE FACTOR.                 ";
203        ERR_TAB(63):="*14* THERE WAS A NON-NUMERIC CHARACTER IN THE         ";
204        ERR_TAB(64):="INTERVAL COLUMNS OF THE DATA RECORD.  PROBABLY        ";
205        ERR_TAB(65):="A KEYPUNCH ERROR.  AN INTERVAL CONSISTS OF TWO        ";
206        ERR_TAB(66):="FIVE DIGIT INTEGERS SPECIFYING THE TOP AND            ";
207        ERR_TAB(67):="BOTTOM OF THE INTERVAL.                               ";
208        ERR_TAB(68):="*15* THERE ARE NO DATA RECORDS ASSOCIATED WITH THE   ";
209        ERR_TAB(69):="RECORD NUMBER YOU SPECIFIED ON THE 'PRINT'            ";
210        ERR_TAB(70):="COMMAND.  EITHER YOU HAVEN'T ENTERED ANY DATA         ";
211        ERR_TAB(71):="WITH THE 'LOAD' COMMAND, OR YOU SPECIFIED A           ";
212        ERR_TAB(72):="NUMBER THAT WAS TOO HIGH OR ZERO.  THE 'LOAD'         ";
213        ERR_TAB(73):="COMMAND NUMBERS EACH RECORD SEQUENTIALLY START-       ";
214        ERR_TAB(74):="ING WITH 1.                                           ";
215        ERR_TAB(75):="*16* YOUR INTERVALS ARE IN THE WRONG ORDER. ";
216        ERR_TAB(76):="*17* NO SERIOUS ERRORS HAVE BEEN MADE.                ";
217        ERR_TAB(77):="*18* YOU PUT A NUMBER ON THE 'DISPLAY' COMMAND. THAT";
218        ERR_TAB(78):="NUMBER WAS INTERPRETED AS A RECORD DESIGNATOR. AFTER";
219        ERR_TAB(79):="A 'LOAD' COMMAND EACH RECORD IS ASSIGNED A NUMBER     ";
220        ERR_TAB(80):="STARTING WITH 1. YOU SPECIFIED A RECORD THAT IS NOT ";
221        ERR_TAB(81):="IN THE WELL YOU ARE EXAMINING. (REMEMBER THAT IN      ";
222        ERR_TAB(82):="EXAMINE MODE ONLY 1 WELL AT A TIME CAN BE WORKED ON ";
223        ERR_TAB(83):="*19* THE NUMBER YOU PUT ON THE 'DISPLAY' COMMAND WAS";
224        ERR_TAB(84):="INTERPRETED TO BE A RECORD DESIGNATOR.  THE RECORD   ";
225        ERR_TAB(85):="THAT WAS SPECIFIED WAS NOT A STRATIGRAPHIC RECORD.    ";
226        ERR_TAB(86):="GIVE THE 'UNITS' COMMAND TO DETERMINE THE PROPER      ";
227        ERR_TAB(87):="RECORD NUMBER YOU WANT.                               ";
228        ERR_TAB(88):="*20* CHECK YOUR DATA. NO STRATIGRAPHIC RECORDS WERE ";
```

```
229         ERR_TAB(89):="FOUND IN THE WELL YOU WERE TRYING TO DISPLAY. THE   ";
230         ERR_TAB(90):="'DISPLAY' COMMAND MUST HAVE STRATIGRAPHIC UNITS TO  ";
231         ERR_TAB(91):="WORK ON.                                            ";
232         ERR_TAB(92):="*21*                                                ";
233               J:=PTR(N);
234               WRITE("  ",ERR_TAB(J));
235               J:=J+1;
236               WHILE ERR_TAB(J)(0|1)¬="*" DO
237                  BEGIN
238                     WRITE("  ",ERR_TAB(J));
239                     J:=J+1;
240                  END;
241         END;
242      END MESSAGE;
243               STRING(7)  ARRAY COM(1::23);
244               STRING(3)  ARRAY MD(1::16);
245           INTEGER I;
246
247
248                  COM( 1):="MTS    ";  MD( 1):="NLI";
249                  COM( 2):="LOAD   ";  MD( 2):="NCO";
250                  COM( 3):="RUN    ";  MD( 3):="NCS";
251                  COM( 4):="WELLS  ";  MD( 4):="NWN";
252                  COM( 5):="SCAN   ";  MD( 5):="MIN";
253                  COM( 6):="STOP   ";  MD( 6):="FOS";
254                  COM( 7):="HELP   ";  MD( 7):="CHM";
255                  COM( 8):="COMMENT";  MD( 8):="TXT";
256                  COM( 9):="EXAMINE";  MD( 9):="STR";
257                  COM(10):="EXPLAIN";  MD(10):="DIO";
258                  COM(11):="SIGNOFF";  MD(11):="POR";
259                  COM(12):="TIME   ";  MD(12):="PRR";
260                  COM(13):="PRINT  ";  MD(13):="HUF";
261                  COM(14):="       ";  MD(14):="RYD";
262                  COM(15):="SAS    ";  MD(15):="TTL";
263                  COM(16):="DISPLAY";  MD(16):="FUL";
264                  COM(17):="SCAN   ";
265                  COM(18):="ALTER  ";
266                  COM(19):="PRINT  ";
267                  COM(20):="COMMENT";
268                  COM(21):="EXPLAIN";
269                  COM(22):="HELP   ";
270                  COM(23):="UNITS  ";
271
272               I:=0; SETPFX("*",1);WRITE(" ");
273               IF C(0|1)="U" THEN MESSAGE ELSE
274      BEGIN
275               WHILE C(I|1)¬=" " DO I:=I+1;
276               WHILE(C(I|1)=" ")AND(I<79)DO I:=I+1;
277               IF(I=79)OR(C(I|3)="ERR")THEN MESSAGE ELSE
278               COMMENT: EXPLAIN WHAT HE WANTS;
279               IF C(I|5)="COMMA" THEN
280                  BEGIN
281                     WRITE(" SAS COMMANDS");WRITE(" ");
282                     FOR I:=1 UNTIL 13 DO
283                        WRITE("   ",COM(I));WRITE(" ");
284                     WRITE(" EXAMINE COMMANDS"); WRITE(" ");
285                     FOR I:=15 UNTIL 23 DO
286                        WRITE("   ",COM(I));
287                     GO TO RETURN;
288                  END ELSE
289               IF C(I|3)="MOD" THEN
290                  BEGIN
291                     WRITE(" MODIFIERS FOR DISPLAY COMMAND");
292                     WRITE(" ");
293                     FOR I:=1 UNTIL 16 DO
294                        WRITE("  @",MD(I));
295                     GOTO RETURN;
296                  END ELSE
297      IF C(I|4)="LOAD" THEN BEGIN
298      WRITE("                                    LOAD                    ");
299      WRITE("                                                            ");
300      WRITE("                            COMMAND DESCRIPTION             ");
301      WRITE("                                                            ");
302      WRITE("                                                            ");
303      WRITE("Purpose:       To read in users data.                       ");
304      WRITE("                                                            ");
305      WRITE("Prototype:     LOAD                                         ");
306      WRITE("                                                            ");
307      WRITE("Description:   The LOAD command will read in user's data    ");
```

```
308     WRITE("               which should consist entirely of well ID, s",
309     "tratigraphic");
310     WRITE("               and lithologic records.  Each 240 character",
311     " record read");
312     WRITE("               in is numbered sequentially starting with '",
313     "1', so every");
314     WRITE("               record will have a unique record designator",
315     " number.");
316     WRITE("                                                          ");
317     WRITE("               The data is read from the file attached to ",
318     "logical unit 5.");
319     WRITE("                                                          ");
320     WRITE("               Overflow                                   ");
321     WRITE("                     An arbitrary limit has been placed o",
322     "n the total");
323     WRITE("                     number of records that can be loaded",
324     ". If internal");
325     WRITE("                     overflow occurs an error message wil",
326     "l be printed out.");
327     WRITE("                                                          ");
328     WRITE("                     In the near future, dynamic storage ",
329     "allocation will");
330     WRITE("                     be implemented so the user can reset",
331     " the upper limits");
332     WRITE("                     of his storage if he desires.       ");
333     WRITE("                                                          ");
334     WRITE("Example:       Supose the file attached to unit 5 contains",
335     " 1 well ID");
336     WRITE("               record, 2 stratigraphic records, and 4 lith",
337     "ologic records.");
338     WRITE("                                                          ");
339     WRITE("               LOAD                                       ");
340     WRITE("                                                          ");
341     WRITE("               will cause 7 records to be read in and numb",
342     "ered from 1 to 7.");
343     END;
344     END;
345           RETURN:
346              WRITE(" "); IOCONTROL(2);
347
348           END.
349
350
351
352     PROCEDURE DISPLAY
353        (STRING(80) VALUE COMMAND;INTEGER VALUE WELL;INTEGER ARRAY
354        WELL_PTR(*);STRING(240) ARRAY RAW_DATA(*);INTEGER VALUE RESULT
355        ERR_N);
356
357        BEGIN
358           COMMENT:  THIS PROCEDURE MAKES CROSS-SECTIONS OF A WELL;
359
360           PROCEDURE SETPFX(STRING(1) VALUE P; INTEGER VALUE L);
361              FORTRAN "SETPFX";
362
363           LOGICAL ERROR,FLAG;
364           LOGICAL LIT,MIN,FOS,CHM,SEC,TXT,COL,STRC,DIOG,
365                   POR,PER,HUE,HYD,TTL,NWN,FUL;
366           STRING(31) ARRAY LIT_TAB(0::40);
367
368           INTEGER C,TOP,BOT,LTP,REC,TYPE,PTR;
369
370           REAL AMT,PCT,OTHER;
371           INTEGER FIRST,I,LAST,SCALE,K;
372
373           REAL ARRAY MP(1::3);
374           REAL ARRAY LIT_T(1::30);
375           INTEGER ARRAY LIT_P(1::30);
376           STRING(10) LINE;
377           STRING(12) STRAT;
378           STRING(5) DEPTH,DEPTH2;
379              STRING(1) CX,BCD;
380
381
382           PROCEDURE DECODE_COMMAND;
383              BEGIN
384                 COMMENT:  PARSE AND DECODE THE COMMAND FROM USER;
385              INTEGER I,J,K1,K2,T,TOKEN,VAL,MFIRST,MLAST;
386              STRING(12) UNIT; STRING(3) MD;
```

```
387
388
389
390         INTEGER PROCEDURE STI(INTEGER VALUE RESULT I);
391            BEGIN
392               COMMENT: THIS PROCEDURE CONVERTS A STRING
393                        TO A NUMBER.              ;
394               INTEGER VAL,J;
395               J:=VAL:=0;
396               WHILE (COMMAND(I|1)¬=" ") AND (J<10) DO
397                  BEGIN
398                     IF(COMMAND(I|1)<"0")OR(COMMAND(I|1)>"9")THEN
399                        BEGIN  SETPFX(" ",1);
400                         WRITE("*** ILLEGAL CHARACTER: '",COMMAND(I|1),"',");
401                           WRITE("    DETECTED IN NUMERIC CONSTANT.");
402                           IOCONTROL(2);
403                           ERR_N:=11; ERROR:=TRUE; GOTO RETURN;
404                        END;
405                     VAL:=VAL*10+(DECODE(COMMAND(I|1))-240);
406                     I:=I+1; J:=J+1;
407                  END;
408               IF J=10 THEN
409                  BEGIN  SETPFX(" ",1);
410                     WRITE("*** NUMERIC CONSTANT TOO LARGE TO CONVERT.");
411                     IOCONTROL(2);
412                     ERR_N:=12; ERROR:=FALSE; GOTO RETURN;
413                  END;
414               RETURN:
415               VAL
416            END STI;
417
418
419            PROCEDURE SCAN(INTEGER VALUE RESULT I);
420
421               BEGIN
422                  WHILE (COMMAND(I|1)=" ") AND (I<79) DO I:=I+1;
423                  IF I=79 THEN
424                     BEGIN
425                        TOKEN:=4; VAL:=0;
426                     END ELSE
427                  IF COMMAND(I|2)="*F" THEN
428                     BEGIN
429                        TOKEN:=2; VAL:=MFIRST; I:=I+2;
430                     END ELSE
431                  IF COMMAND(I|2)="*L" THEN
432                     BEGIN
433                        TOKEN:=2; VAL  :=MLAST; I:=I+2;
434                  END ELSE
435               IF (COMMAND(I|1)>="0")AND(COMMAND(I|1)<="9") THEN
436                  BEGIN
437                     TOKEN:=2; VAL:=STI(I);
438                     IF ERROR THEN GOTO RETURN;
439                     IF (VAL<WELL_PTR(WELL))OR
440                         (VAL>WELL_PTR(WELL+1)-1) THEN
441                        BEGIN SETPFX(" ",1);
442                         WRITE("*** RECORD DESIGNATOR:  '",VAL,"' SPECIFIES"
443                         );
444                         WRITE("    A RECORD THAT IS NOT IN THE WELL.");
445                         ERR_N:=18; ERROR:=TRUE;
446                         IOCONTROL(2); GOTO RETURN;
447                        END;
448                     IF RAW_DATA(VAL)(0|1)¬="2" THEN
449                        BEGIN SETPFX(" ",1);
450                         WRITE("*** RECORD:  '",VAL,"' IS NOT A");
451                         WRITE("    STRATOGRAPHIC RECORD.");
452                         ERR_N:=19; ERROR:=TRUE;
453                         IOCONTROL(2); GO TO RETURN;
454                        END;
455                  END ELSE
456                  IF COMMAND(I|6)="SCALE=" THEN
457                     BEGIN
458                        TOKEN:=3; I:=I+6; VAL:=STI(I);
459                        IF ERROR THEN GOTO RETURN;
460                     END ELSE
461                  IF COMMAND(I|5)="CHAR=" THEN
462                     BEGIN TOKEN:=5;
463                        I:=I+5; BCD:=COMMAND(I|1);
464                        WHILE COMMAND(I|1)¬=" " DO I:=I+1;
465                     END ELSE
```

```
466                BEGIN
467                   J:=0; UNIT:="              ";
468                   WHILE (COMMAND(I|1)¬=" ") AND (J<=11) DO
469                      BEGIN
470                         UNIT(J|1):=COMMAND(I|1);
471                         I:=I+1; J:=J+1;
472                      END;
473                   IF J>11 THEN
474                      BEGIN
475                         WRITE ("*** STRATIGRAPHIC UNIT '",UNIT,"'");
476                         WRITE ("    EXCEEDS MAXIMUM STRING LENGTH.");
477                IOCONTROL(2);
478                         ERR_N:=7; ERROR:=TRUE; GOTO RETURN;
479                      END;
480                   K1:=WELL_PTR(WELL);
481                   K2:=WELL_PTR(WELL+1)-1;
482          FLAG:=TRUE;
483                   WHILE (K1<K2)AND(FLAG) DO
484                      BEGIN
485                         IF RAW_DATA(K1)(0|1)="2" THEN
486                            BEGIN
487                               FLAG:=TRUE;
488                               IF UNIT(0|12)=RAW_DATA(K1)(1|12) THEN
489                                BEGIN
490                                  TOKEN:=2; VAL:=K1; FLAG:=FALSE;
491                                END;
492                            END;
493                         K1:=K1+1;
494                      END;
495                   IF FLAG THEN
496                      BEGIN SETPFX(" ",1);
497                   WRITE ("*** STRATOGRAPHIC UNIT '",UNIT,"' NOT FOUND.");
498                   ERR_N:=8; IOCONTROL(2); ERROR:=TRUE;
499                      END;
500                   END;
501                   RETURN:
502
503             END SCAN;
504
505           PROCEDURE CHECK;
506
507             BEGIN
508                IF (VAL=WELL_PTR(WELL))OR
509                   (VAL=WELL_PTR(WELL+1)) THEN
510                   BEGIN SETPFX(" ",1);
511                      WRITE("*** NO STRATIGRAPHIC RECORDS FOUND"
512                            ," IN WELL:  '",RAW_DATA(WELL_PTR(WELL))
513                             (1|15),"'.");
514                      WRITE("   DISPLAY ABORTED.");
515
516                      ERR_N:=20; ERROR:=TRUE;
517                      IOCONTROL(2); GOTO RETURN;
518                   END;
519             END CHECK;
520
521
522             COMMENT:  DEFAULT INITIALIZATIONS;
523             LIT:=COL:=SEC:=TRUE; I:=1; NWN:=ERROR:=FALSE;
524             MIN:=FOS:=CHM:=TXT:=STRC:=DIOG:=POR:=PER:=HUE:=HYD:=TTL:=FALSE
525             COMMENT:  NOW FIND ANY MODIFIERS;
526             WHILE COMMAND(I|1)¬=" " DO
527                BEGIN
528                   IF COMMAND(I|1)="@" THEN
529                      BEGIN  COMMENT: MODIFIER DETECTED;
530                         MD:=COMMAND(I+1|3);
531                         IF MD="NLI" THEN LIT :=FALSE ELSE
532                         IF MD="NCO" THEN COL :=FALSE ELSE
533                         IF MD="NCS" THEN SEC :=FALSE ELSE
534                         IF MD="MIN" THEN MIN :=TRUE  ELSE
535                         IF MD="FOS" THEN FOS :=TRUE  ELSE
536                         IF MD="NWN" THEN NWN :=TRUE  ELSE
537                         IF MD="CHM" THEN CHM :=TRUE  ELSE
538                         IF MD="TXT" THEN TXT :=TRUE  ELSE
539                         IF MD="STR" THEN STRC:=TRUE  ELSE
540                         IF MD="DIO" THEN DIOG:=TRUE  ELSE
541                         IF MD="POR" THEN POR :=TRUE  ELSE
542                         IF MD="PER" THEN PER :=TRUE  ELSE
543                         IF MD="HUE" THEN HUE :=TRUE  ELSE
544                         IF MD="HYD" THEN HYD :=TRUE  ELSE
```

```
                        IF MD="TTL" THEN TTL :=TRUE  ELSE
                        BEGIN SETPFX(" ",1);
                           WRITE ("*** ILLEGAL MODIFIER:  '",MD,"'.");
                           ERR_N:=6; ERROR:=TRUE;  GO TO RETURN;
                        END;
                        I:=I+3;
                     END;
                  I:=I+1;
               END;
            COMMENT:  DETERMINE INTERVAL AND SCALE FACTOR;
            ERROR:=FALSE;
            SCALE:=20; CX:="|";
               VAL:=WELL_PTR(WELL);
               WHILE(RAW_DATA(VAL)(0|1)¬="2")AND(VAL<WELL_PTR(WELL+1)) DO
                  VAL:=VAL+1;
               CHECK; MFIRST:=VAL;
               VAL:=WELL_PTR(WELL+1)-1;
               WHILE(RAW_DATA(VAL)(0|1)¬="2")AND(VAL>WELL_PTR(WELL)) DO
                  VAL:=VAL-1; CHECK; MLAST:=VAL;
               FIRST:=MFIRST; LAST:=MLAST;

         T:=J:=0;
         WHILE J<4 DO
         BEGIN
           SCAN(I); IF ERROR THEN GOTO RETURN;
           J:=J+1;
           IF (TOKEN=2) AND (T=0) THEN
           BEGIN
             FIRST:=LAST:=VAL; T:=1;
           END ELSE
           IF (TOKEN=2) AND (T=1) THEN LAST:=VAL ELSE
           IF TOKEN=3 THEN SCALE:=VAL;
           IF TOKEN=5 THEN CX:=BCD;
         END;
         IF SCALE=0 THEN
           BEGIN SETPFX(" ",1);
             WRITE("*** ZERO IS ILLEGAL SCALE FACTOR.");
ERROR:=TRUE; IOCONTROL(2);
             ERR_N:=13; GOTO RETURN;
           END;
            RETURN:
       END DECODE_COMMAND;


INTEGER PROCEDURE ST2(STRING(5) VALUE A);

  BEGIN
    INTEGER VAL; VAL:=0; ERROR:=FALSE;
    FOR I:=0 UNTIL 4 DO
      BEGIN
        IF A(I|1)=" " THEN A(I|1):="0";
        IF(A(I|1)<"0")OR(A(I|1)>"9")THEN
          BEGIN SETPFX(" ",1);
            WRITE("*** ILLEGAL INTERVAL: '",A,"' DETECTED");
            WRITE("     IN DATA FILE.  DISPLAY ABORTED.");
            WRITE("     ERROR FOUND IN RECORD:",REC);
             IOCONTROL(2);
            ERR_N:=14; ERROR:=TRUE; GOTO RETURN;
          END
        ELSE
          VAL:=VAL*10+(DECODE(A(I|1))-240);
      END;
      RETURN:
      VAL

   END ST2;


PROCEDURE CALC_PCNTS;

  BEGIN
   COMMENT: DETERMINE MAIN PERCENTAGE IN
            COLUMNS 13-18 ON LITH CARD.;

    STRING(6) TEMP;
    INTEGER PTR,NDX;
    INTEGER ARRAY CHECK(1::3);
    FOR I:=1 UNTIL 3 DO
      BEGIN
```

```
624             MP(I):=CHECK(I):=0;
625           END;
626
627         TEMP:=RAW_DATA(REC)(12|6);
628         FOR K:=1 UNTIL 3 DO
629           BEGIN
630             PTR:=2*(K-1);
631             NDX:=DECODE(TEMP(PTR|1))-240;
632             IF(TEMP(PTR|1)>="1")AND(TEMP(PTR|1)<="3") THEN
633               BEGIN
634                 COMMENT: CALCULATE PERCENTAGE;
635                 MP(NDX):=PERCENT(TEMP(PTR+1|1));
636                 IF(CHECK(NDX)=1)AND(¬NWN) THEN
637                   BEGIN SETPFX(".",1);
638                     WRITE("WARNING---REDEFINITION OF PERCENTAGE.");
639                     WRITE("   '",TEMP,"' FOUND IN COLUMNS 13-18",
640                           " OF RECORD:",REC,",");
641                     WRITE("   SPECIFIES A SECOND DEFINITION OF",
642                           " A PERCENTAGE CODE.");
643                     WRITE("---FIRST DEFINITION IGNORED.  ",
644                           " EXECUTION RESUMES:");
645                                 IOCONTROL(2); SETPFX(" ",1);
646                   END;
647                 CHECK(NDX):=1;
648               END ELSE
649               IF(TEMP(PTR|1)¬=" ")AND(¬NWN) THEN
650                 BEGIN SETPFX(".",1);
651                   WRITE("WARNING---UNEXPECTED '",TEMP(PTR|1),"'",
652                         " WAS DETECTED.");
653                   WRITE("   EXPECTING A '1','2','3', OR A ' '.");
654                   WRITE("   DETECTED IN COLUMN: ",PTR+13,", OF RECORD",
655                         REC,".");
656                   WRITE("---DATA IGNORED.  EXECUTION RESUMES:");
657                         IOCONTROL(2); SETPFX(" ",1);
658                 END;
659           END;
660
661       END CALC_PCNTS;
662
663
664     PROCEDURE BUILD_DESCS(INTEGER VALUE START,LEN,MAX);
665
666       BEGIN
667         COMMENT: THIS PROCEDURE BUILDS A DOUBLE
668                  ARRAY OF DESCRIPTORS WHICH POINT
669                  INTO ENGLISH TABLES.;
670         REAL T;
671         T:=ST2(RAW_DATA(REC)(7|5))-ST2(RAW_DATA(REC)(2|5));
672         T:=T/(BOT-TOP);
673         FOR I:= 0 UNTIL LEN-1 DO
674           BEGIN
675             GET_UNIT(RAW_DATA(REC)(START+I*4|4),REC);
676             IF (PTR>MAX)AND(¬NWN)AND(TYPE¬=0) THEN
677               BEGIN SETPFX(".",1);
678                 WRITE("WARNING---ILLEGAL CODE '",PTR,"' DETECTED.");
679                 WRITE("   FOUND IN RECORD: ",REC,", OF DATA FILE.");
680                 WRITE("--- EXECUTION RESUMES:");
681                       IOCONTROL(2); PTR:=40; SETPFX(" ",1);
682               END;
683             IF TYPE¬=0 THEN
684               BEGIN
685                 LIT_T(C):=MP(TYPE)*PCT*T;
686                 LIT_P(C):=PTR;
687                 C:=C+1;
688               END;
689           COMMENT: NOW SORT TABLE;
690             FLAG:=TRUE;
691             WHILE (FLAG)AND(TYPE¬=0) DO
692               BEGIN INTEGER TEMP; REAL TMP;
693                 FLAG:=FALSE;
694                 FOR I:=1 UNTIL C-2 DO
695                   BEGIN
696                     IF LIT_T(I)<LIT_T(I+1) THEN
697                       BEGIN TEMP:=LIT_P(I+1);
698                         LIT_P(I+1):=LIT_P(I);
699                         LIT_P(I):=TEMP; FLAG:=TRUE;
700                         TMP:=LIT_T(I+1);
701                         LIT_T(I+1):=LIT_T(I);
702                         LIT_T(I):=TMP;
703                       END;
```

```
704                     IF LIT_P(I)=LIT_P(I+1) THEN
705                       BEGIN
706                         LIT_T(I):=LIT_T(I)+LIT_T(I+1);
707                       FOR K:=I+1 UNTIL C-1 DO
708                           BEGIN
709                               LIT_T(K):=LIT_T(K+1);
710                               LIT_P(K):=LIT_P(K+1);
711                           END;
712                       C:=C-1;
713                       END;
714                   END;
715           END ;
716               END;
717
718       END BUILL_DESCS;
719
720
721     PROCEDURE COMPOSE_LINE;
722
723       BEGIN
724         COMMENT:  CROSS SECTION LINE FIRST;
725         INTEGER I,J,K;
726         REAL A;
727         STRING(10) CHAR;
728         LINE:="0000000000";
729         J:=0; I:=1;
730         WHILE (I<C) AND (J<10) DO
731           BEGIN
732             A:=LIT_T(I);
733             FOR K:=0 UNTIL 9 DO CHAR(K|1):=LIT_TAB(LIT_P(I))(3|1);
734             IF (A>5) AND (A<15) THEN  BEGIN LINE(J|1):=CHAR(0|1);J:=J+1 END
735          ELSE IF (A>14) AND (A<26) THEN BEGIN LINE(J|2):=CHAR(0|2);J:=J+2 END
736          ELSE IF (A>25) AND (A<35) THEN BEGIN LINE(J|3):=CHAR(0|3);J:=J+3 END
737          ELSE IF (A>34) AND (A<46) THEN BEGIN LINE(J|4):=CHAR(0|4);J:=J+4 END
738          ELSE IF (A>45) AND (A<55) THEN BEGIN LINE(J|5):=CHAR(0|5);J:=J+5 END
739          ELSE IF (A>54) AND (A<66) THEN BEGIN LINE(J|6):=CHAR(0|6);J:=J+6 END
740          ELSE IF (A>65) AND (A<75) THEN BEGIN LINE(J|7):=CHAR(0|7);J:=J+7 END
741          ELSE IF (A>74) AND (A<86) THEN BEGIN LINE(J|8):=CHAR(0|7);J:=J+7 END
742          ELSE IF (A>74) AND (A<86) THEN BEGIN LINE(J|8):=CHAR(0|8);J:=J+8 END
743          ELSE IF (A>85) AND (A<95) THEN BEGIN LINE(J|9):=CHAR(0|9);J:=J+9 END
744          ELSE IF (A>94) THEN BEGIN LINE(J|10):=CHAR(0|10); J:=J+10;     END;
745           I:=I+1;
746           END
747
748       END COMPOSE_LINE;
749
750
751     PROCEDURE GET_UNIT(STRING(4) VALUE UNIT; INTEGER VALUE REC);
752
753       BEGIN
754         INTEGER PROCEDURE STI(INTEGER VALUE P,L);
755           BEGIN
756             INTEGER V; V:=0;
757             FOR I:=P UNTIL P+L-1 DO
758               BEGIN IF UNIT(I|1)=" " THEN UNIT(I|1):="0";
759                 IF (UNIT(I|1)<"0") OR (UNIT(I|1)>"9") THEN
760                   BEGIN SETPFX(".",1);
761                     WRITE("WARNING---ILLEGAL DESCRIPTIVE UNIT.");
762                     WRITE("    '",UNIT,"' WAS DETECTED IN RECORD:",
763                         REC,", OF DATA FILE.");
764                     WRITE("---UNIT IGNORED.  EXECUTION RESUMES:");
765                         IOCONTROL(2); SETPFX(" ",1);
766                     ERROR:=TRUE;
767                   END ELSE
768                 V:=V*10+(DECODE(UNIT(I|1))-240);
769               END;
770             V
771           END STI;
772         ERROR:=FALSE;
773         IF UNIT="    " THEN TYPE:=0 ELSE
774         BEGIN
775           TYPE:=STI(0,1);
776           PTR:=STI(1,2);
777           PCT:=PERCENT(UNIT(3|1));
778         END;
779         IF ERROR THEN TYPE:=0;
780
781       END GET_UNIT;
782
783
```

```
784     INTEGER PROCEDURE PERCENT(STRING(1) VALUE A);
785
786       BEGIN
787         INTEGER PCNT;
788         IF(A<"0")OR(A>"9")THEN
789           BEGIN     SETPFX(".",1);
790             WRITE("WARNING---ILLEGAL PERCENTAGE CODE '",A,"' DETECTED.");
791             WRITE("    FOUND IN RECORD:",REC,".");
792             WRITE("    WILL BE INTERPRETED AS ZERO PERCENT.");
793             WRITE("---EXECUTION RESUMES:");
794             PCNT:=0;
795           END ELSE
796           CASE (DECODE(A)-240)+1  OF
797           BEGIN
798             PCNT:=5;
799             PCNT:=15;
800             PCNT:=25;
801             PCNT:=35;
802             PCNT:=45;
803             PCNT:=55;
804             PCNT:=65;
805             PCNT:=75;
806             PCNT:=85;
807             PCNT:=95;
808           END;
809         PCNT
810
811       END PERCENT;
812
813
814     PROCEDURE PRINT_TITLE;
815
816       BEGIN
817         COMMENT:  THIS PROCEDURE PRINTS THE TITLE FOR THE CROSS_SECTION;
818         STRING(80) TEMP;
819         SETPFX(" ",1); INTFIELDSIZE:=5;
820         TEMP:=RAW_DATA(WELL_PTR(WELL))(0|80);
821         WRITE(" ");
822             WRITE("  ****************************");
823             WRITE("  *                          *");
824             WRITE("  *      ",TEMP(1|15), "     *");
825             WRITE("  *                          *");
826             WRITE("  ****************************");
827             WRITE(" ");
828         WRITE("     COORDINATES:  ",TEMP(55|14)); WRITE(" ");
829         WRITE("     STATE:  ",TEMP(36|19)); WRITE(" ");
830         WRITE("     COUNTY:  ",TEMP(16|20)); WRITE(" ");
831         WRITE("     ELEVATION:  ",TEMP(69|5),"  PERMIT NUMBER:  ",
832             TEMP(75|5)); WRITE(" ");
833         WRITE(" ");
834         WRITE("     SCALE FACTOR:  1 INCH = 6 LINES =",SCALE,"FEET");
835         WRITE(" "); WRITE(" ");
836
837       END PRINT_TITLE;
838
839
840     PROCEDURE PRINT_HEADING;
841       BEGIN SETPFX(" ",1); WRITE(" ");
842         WRITE("STRATIGRAPHIC    CROSS-");
843         WRITE("    UNIT          SECTION      DEPTH       LITHOLOGY");
844         WRITE("-"); FOR I:=1 UNTIL 77 DO WRITEON("-");
845
846       END PRINT_HEADING;
847
848
849     COMMENT: INITIALIZE LITHOLOGY TABLE;
850
851     LIT_TAB( 0):="07AARENITE                    ";
852     LIT_TAB( 1):="14ALITHIC ARENITE             ";
853     LIT_TAB( 2):="15AARKOSIC ARENITE            ";
854     LIT_TAB( 3):="14AQUARTZ ARENITE             ";
855     LIT_TAB( 4):="09ASUBARKOSE                  ";
856     LIT_TAB( 5):="14ASUBLITHARENITE             ";  (EXAMPLE DICTIONARY RECORD)
857     LIT_TAB( 6):="06WWACKES                     ";
858     LIT_TAB( 7):="11WQUARTZWACKE                ";
859     LIT_TAB( 8):="16WLITHIC GREYWACKE           ";
860     LIT_TAB( 9):="21WFELDSPATHIC GREYWACKE      ";
861     LIT_TAB(10):="13WARKOSIC WACKE              ";
862     LIT_TAB(11):="09-MUDSTONES                  ";
863     LIT_TAB(12):="09LLIMESTONE                  ";
```

```
LIT_TAB(13):="20MMICRITE & DISMICRITE          ";
LIT_TAB(14):="21MFOSSILIFEROUS MICRITE         ";
LIT_TAB(15):="17MSPARSE BIOMICRITE             ";
LIT_TAB(16):="17MPACKED BIOMICRITE             ";
LIT_TAB(17):="24BPOORLY WASHED BIOSPARITE      ";
LIT_TAB(18):="19BUNSORTED BIOSPARITE           ";
LIT_TAB(19):="17BSORTED BIOSPARITE             ";
LIT_TAB(20):="18BROUNDED BIOSPARITE            ";
LIT_TAB(21):="21CCRYSTALLINE LIMESTONE         ";
LIT_TAB(22):="10BBIOLITHITE                    ";
LIT_TAB(23):="19DDOLOMITIC LIMESTONE           ";
LIT_TAB(24):="20DCRYSTALLINE DOLOMITE          ";
LIT_TAB(25):="19DCALCAREOUS DOLOMITE           ";
LIT_TAB(26):="06GGYPSUM                        ";
LIT_TAB(27):="09AANHYDRITE                     ";
LIT_TAB(28):="04SSALT                          ";
LIT_TAB(29):="05CCHERT                         ";
LIT_TAB(30):="04CCOAL                          ";
LIT_TAB(31):="09PPHOSPHATE                     ";
LIT_TAB(32):="14IIRON SEDIMENTS                ";
LIT_TAB(33):="17*CRYSTALLINE ROCKS             ";
LIT_TAB(34):="14*ACIDIC IGNEOUS                ";
LIT_TAB(35):="15VACIDIC VOLCANIC               ";
LIT_TAB(36):="13*BASIC IGNEOUS                 ";
LIT_TAB(37):="14VBASIC VOLCANIC                ";
LIT_TAB(38):="17MMETAMORPHIC ROCKS             ";
LIT_TAB(39):="05OOTHER                         ";
LIT_TAB(40):="09**********                     ";

    FOR I:=1 UNTIL 30 DO
      BEGIN
        LIT_T(I):=0;
        LIT_P(I):=40;
      END;


DECODE_COMMAND; IF ERROR THEN GOTO RETURN;
REC:=FIRST;
IF TTL THEN PRINT_TITLE;
PRINT_HEADING;
WHILE RAW_DATA(REC)(0|1)¬="2" DO REC:=REC+1;
WHILE (REC<=LAST)AND(RAW_DATA(REC)(0|1)="2") DO
  BEGIN
    COMMENT: FIGURE OUT NUMBER OF LINES TO PRINT;
    TOP:=ST2(RAW_DATA(REC)(13|5));
   IF ERROR THEN GOTO RETURN;
    BOT:=ST2(RAW_DATA(REC)(18|5));
   IF ERROR THEN GOTO RETURN;
    IF TOP>BOT THEN
      BEGIN
      SETPFX(" ",1);
        WRITE("*** INTERVAL ERROR DETECTED IN");
        WRITE("    RECORD: ",REC,".");
        ERR_N:=16; IOCONTROL(2); GOTO RETURN;
      END;
    LTP:=TRUNCATE((BOT-TOP)/SCALE*6);
    STRAT:=RAW_DATA(REC)(1|12);
    DEPTH:=RAW_DATA(REC)(13|5);
    DEPTH2:=RAW_DATA(REC)(18|5);
    REC:=REC+1;
    C:=1;
    WHILE RAW_DATA(REC)(0|1)="3" DO
      BEGIN
        CALC_PCNTS;
        BUILD_DESCS(18,4,39);
        REC:=REC+1;
      END; AMT:=0;
    FOR I:=1 UNTIL C-1 DO
      BEGIN
        LIT_T(I):=LIT_T(I)/100;
        AMT:=AMT+LIT_T(I);
      END;
    IF AMT>100 THEN WRITE("TROUBLE AHEAD");
    OTHER:=100-AMT;
    IF OTHER>1 THEN
      BEGIN
        LIT_T(C):=OTHER; LIT_P(C):=39;
        C:=C+1;
      END;
    COMMENT NOW OUTPUT PICTURE;
```

```
    944       IOCONTROL(4);
    945           COMPOSE_LINE; I:=LIT_P(1);
    946           WRITE(" ",STRAT," ",CX,LINE,CX," - ",DEPTH,"  ",LIT_TAB(I)
    947           (3|25),LIT_T(1),"%");
    948          FOR I:=2 UNTIL C-1 DO
    949       WRITE("                ",CX,LINE,CX,"           ",LIT_TAB(LIT_P
    950           (I))(3|25),LIT_T(I),"%");
    951       FOR I:=C UNTIL LTP DO
    952       WRITE("                ",CX,LINE,CX);
    953         END;
    954       WRITE("                ",CX,"0000000000",CX," - ",DEPTH2);
    955          WRITE(" ");
    956          WRITE("   END OF CROSS-SECTION.");
    957          WRITE(" ");
    958          IOCONTROL(2);
    959        RETURN:
    960         END.
END OF FILE
$SINK PREVIOUS

$LIST 2JAC:CARLA
      1             SUBROUTINE BLOCK
      2             LOGICAL*1 LINE(8,4)
      3             LOGICAL*1 BUFF(112)
      4             INTEGER*4 SETPFX,OLD
      5             OLD=SETPFX(' ',1)
      6             DO 2 I=1,4
      7       2     READ(3,102) (LINE(J,I),J=1,8)
      8             WRITE(6,103)
      9             DO 1 I=1,4
     10             WRITE(6,101)
     11             DO 1 J=1,12
     12             CALL BLKLTR(LINE(1,I),J,BUFF,8)
     13       1     WRITE(6,100) BUFF
     14             WRITE(6,101)
     15             RETURN
     16       100   FORMAT(' ',112A1)
     17       101   FORMAT('-')
     18       102   FORMAT(8A1)
     19       103   FORMAT('1')
     20             END
END OF FILE
$SINK PREVIOUS

$LIST 2JAC:FSUB
      1             SUBROUTINE INPUT(BUFFER,FLAG)
      2             DIMENSION BUFFER(60)
      3             LOGICAL*1 FLAG
      4             FLAG=.TRUE.
      5       C
      6       C     WELL DATA IS READ FROM LOGICAL UNIT 5
      7       C
      8             READ (5,100,END=200) BUFFER
      9       100   FORMAT(60A4)
     10             GO TO 300
     11       200   FLAG = .FALSE.
     12       300   RETURN
     13             END
END OF FILE
$SINK PREVIOUS
```

APPENDIX III

Appendix III contains the complete internal dictionary, worksheet format, and user instructions.

Worksheet Format

Definition codes

Part 1

Card Code — col: 1

Code to identify type of cards when in batch sequence.

Type	Code
Lithology cards	3
Stratigraphic cards	2
ID card	1

Part 2

Data Type Code — col: 2

Code to identify source of data for data level identification.

Type	Code
Core data	1
Chip data	2
Descriptive log data	3

Part 3

Interval Definition Code — cols: 3-12

Interval of well to be described by following information-measured in feet by defining top and bottom of interval. (No commas, right justified)

Top of interval	cols 3-7
Bottom of interval	cols 8-12

example: an interval of 4,000 feet, starting at 4,250 and ending at 8,250.

col	3	4	5	6	7	8	9	10	11	12
	0	4	2	5	0	0	8	2	5	0

Part 4

Lithologic Definition Code — cols: 13-18

This is the code defining only the number and percentages of distinct lithologies present in the defined interval. A maximum of three (3) lithologies are allowed.

The format is in two parts: the lithology code and percentage.

example: pure limestone over entire interval.

cols	13	14	15	16	17	18
	1	9				

1 = primary lithology (lithology type classification is in Description Codes, part 2)
9 = 100% (see Definition Codes, part 5, following)

example: limestone with 10% shale partings.

cols	13	14	15	16	17	18
	1	9	2	1		

1 = primary lithology
9 = 90%
2 = secondary lithology
1 = 10%

example: limestone with 20% shale partings and 7% chert stringers.

cols	13	14	15	16	17	18
	1	7	2	2	3	0

1 = primary lithology
7 = 73%
2 = secondary lithology
2 = 20%
3 = tertiary lithology
0 = 7%

Part 5

Percentage Code

Percentage	used in conjunction with other descriptions Code
0 - 9.9	0
10.0 - 19.9	1
20.0 - 29.9	2
30.0 - 39.9	3
40.0 - 49.9	4
50.0 - 59.9	5
60.0 - 69.9	6
70.0 - 79.9	7
80.0 - 89.9	8
90.0 -100.0	9
not specified	x or blank

DESCRIPTIVE CODES

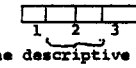
Part 1

Format for the Descriptive Section

The following section concerns the codes for the description of lithologic components and their properties. Each subject is approached in the same manner. The information is coded in a block called the descriptive unit.

The descriptive unit consists of three parts:
1. lithologic identification
2. property code
3. percentage,

expressed in four (4) columns on the data sheet.



one descriptive unit

The first column is for the lithologic identification code (see Definition Codes, part 4). The second and third columns are for the descriptive code (see following code lists). And the fourth column is for the percentage code (see Definition Codes, part 5).

example: some fossil spirifer brachiopods in the major lithology, the brachiopods comprising 80% of the fossils present.

1	4	4	8

1 = major lithology - col 1 of descriptive unit
44 = spirifer brachiopod - cols 2 & 3 of descriptive unit
8 = 80% code - col 4 of descriptive unit

Part 2

Lithologic Description Code — cols: 19-34

This section describes the lithology (rock name) for the lithologic components. Percentage and lithologic component are as previously defined. Four (4) maximum descriptive units are allowed.

Lithology	Code
Clastics	
Arenites	00
lithic arenite	01
arkosic arenite	02
quartz arenite	03
subarkose	04
sublitharenite	05
Wackes	06
quartzwacke	07
lithic greywacke	08
feldspathic greywacke	09
arkosic wacke	10
Mudstones- shales	11
Conglomerates	12
Carbonates	
Limestones	
mudstone	13
wackestone	14
packstone	15
grainstone	16
boundstone	17
crystalline limestone	18
fragmental limestone	19
other	20
other	21
other	22
Dolomites	
Dolomitic limestone	23
Crystalline dolomite	24
Calcarreous dolomite	25
Evaporites	
Gypsum	26
Anhydrite	27
Salt	28
Others	
Chert	29
Coal	30
Phosphate	31
Iron sediments	32
Crystalline rocks	33
acidic igneous	34
acidic volcanic	35
basic igneous	36
basic volcanic	37
Metamorphic rocks	38

Part 3

Color and Hue Description Code — cols: 35,36

The color and description of hue do not conform to the descriptive unit format. Instead it is a two part code.

Column 31 refers to color (see code list following discussion)

x	

color

Column 32 refers to hue (see code list following discussion)

	x

hue

Color	Code
Grey	1
Green	2
Red	3
Brown	4
Yellow	5
Blue	6
Variegated	7
White	8
Black	9

Hue	Code
Light	1
Dark	2

Part 4

Textural Description Code — cols: 37-52

This section describes textures in lithologic sub-units by type and percentage. Four (4) descriptive units are allowed for textures.

Textures	Code
Carbonates	
intraclastic	00
skeletal	01
pelletal	02
lumps	03
algal coated	04
pisolites	05
organic framework	06
crystalline coarse (>2mm; >-1ϕ)	07
fine (<2mm; <-1ϕ)	08
euhedral	09
subhedral	10
anhedral	11
oolites	12
mottled	13
laminate	14
burrowed	15
dismicrite	16
birdseye	17
Evaporites	
crystalline coarse (>2mm; >-1ϕ)	18
fine (<2mm; <-1ϕ)	19
euhedral	20
subhedral	21
anhedral	22
laminate	23
reticulate	24
deccussate	25
flaser	26
Hopper/Chevron	27
Clastics	
maturity	
supermature	28
mature	29
immature	30
textural inversion	31
grain size	
coarse (>0.5mm; >1ϕ)	32
medium (0.5mm-0.125mm; 1ϕ-3ϕ)	33
fine (0.125mm-0.0625mm; 3ϕ-4ϕ)	34
mud (<0.0625mm; <4ϕ)	35
roundness	
very angular	36
angular	37
sub-angular	38
sub-rounded	39
rounded	40
well-rounded	41
sphericity	
platy	42
compact	43
elongated	44
bladed	45
packing	
welded	46
interplanation	47
point contact	48
long contact	49
concave contact	50
floating	51
Shales	
fissile	52
soft	53
hard	54
grain size	
silt (0.0625-0.0039mm; 4ϕ-8ϕ)	55
clay (<0.0039; <8ϕ)	56

Part 5

Fossil Description Code — cols: 53-108

The following is the descriptive list for fossil types. The symbol N/S represents not specified. There are fourteen (14) descriptive units allowed for fossils.

Fossils	Code
Plants (N/S)	0
pteridophyta	1
gymnospermae	2
angiospermae	3
bryophyta	4
thallophyta	5
Algae (N/S)	6
stromatolite	7
algal pisolites/oncolites	8
calcispheres	9
rhodophycophyta	10
chlorophycophyta	11
charophyta	12
schizophyta	13
Invertebrates (N/S)	14
archaeocyanthids	15
Porifera (N/S)	16
desmospongia	17
hyalospongia	18
calcispongia	19
receptaculites	20
spicules	21
stromatoporoids	22
Coelenterates (N/S)	23
hydrozoans	24
schyphozoans	25

conularids	26
tabular corals	27
rugose corals	28
scleractinids	29
Bryozoans	30
ctenostomata	31
cyclostomata	32
trepostomata	33
cryptostomata	34
cheilostomata	35
Brachiopods (N/S)	36
inarticulate	37
articulate	38
orthids	39
strophomenids	40
pentamerids	41
phynchonellids	42
spiriferids	43
tetrabratulids	44
Annelida (N/S)	45
Mollusca (N/S)	46
amphineurans	47
monoplachophorans	48
Gastropodia	49
prosobranchia	50
opistobranchia	51
pulmonata	52
Pelecypodia/Bivalvia (N/S)	53
nucloids	54
mytiloids	55
myalinoids	56
oysters	57
pectenoids	58
pholads	59
rudists	60
lucinoids	61
Schaphods (N/S)	62
Cephalopods (N/S)	63
nautaloids	64
ammonoids	65
coleoidea	66
belemenoids	67
Arthropods (N/S)	68
trilobites (N/S)	69
olinellids	70
agnostia	71
corynexochida	72
ptycoporida	73
phacopida	74
lichida	75
odontopleura	76
aglaspida	77
crustacea	78
arachnoidea	79
insectea	80
Enchinodermata (N/S)	81
crinoids	82
blastoids	83
echinoids	84
cystoids	85
ophiuroids	86
Hemichordata	87
graptozoa	88
conodonts	89
Protozoa	90
foramineferida	91
Allogromiina	92
Textulariina	93
Fusulinia	94
Millolina	95
Rotallina	96
Radiolaria	97
Other	98

Part 6

Mineralogy Description Code — cols: 112-151

The following section is the code list of minerals. There are ten (10) descriptive units allowed.

Mineralogy	Code
Actinolite	00
Andradite garnet	01
Anhydrite	02
Apatite	03
Aragonite	04
Augite	05
Barite	06
Beryl	07
Brucite	08
Calcite	09
Cassiderite	10
Chert	11
Chlorite	12
Corundum	13
Diopside	14
Dolomite	15
Enstatite	16
Epidote	17
Feldspars	18
plagioclase Ca	19
Na	20
orthoclase	21
microcline	22
perthite	23
Fluorite	24
Halite	25
Glauconite	26
Glaucophane	27

Grossularite garnet	28
Gypsum	29
Hornblende	30
Hypersthene	31
Kyanite	32
Monozite	33
Olivine	34
Quartz	35
Riebicite	36
Rutile	37
Siderite	38
Sillmanite	39
Sphalerite	40
Sphene	41
Spinel	42
Staurolite	43
Sylvite	44
Topaz	45
Tourmaline	46
Tremolite	47
Zircon	48
Other	49
Other	50
Other	51
Other	52
Other	53
Other	54
Other	55

Part 7

Sedimentary Structure Code — cols: 152-171

The following is a list of sedimentary structures. There are five (5) descriptive units allowed.

Sedimentary Structures	Code
Planar bedding	00
Laminations	01
Cross-bedding	02
Graded bedding	03
Varves	04
Linear bedding	05
Striations	06
Sand lineation	07
Casts	08
Current markings	09
Drag marks	10
Groove & groove casts	11
Bedding plane markings	12
Wave & wash marks	13
Pits & prints	14
Cut outs	15
Scoops	16
Ripple marks	17
Mud cracks	18
Sole marks	19
Para-ripple	20
Load cast	21
Flute	22
Deformed bedding	23
soft sediment deformation	24
clay balls	25
Distributed bedding	26
Sedimentary sills	27
Sedimentary dikes	28
Convoluted lamination	29
Veins	30
Slump	31
Mudcracks	32
Structureless	33
Borings	34
Tracks & trails	35
Cast & molds	36
Pellets	37
Coprolites	38
Burrows	39
Breccia	40
fault	41
solution	42
Joints	43
vertical	44
horizontal	45
angle - high (>45°)	46
low (<45°)	47

Part 8

Diagenetic Description Code — cols: 172-191

The following is a list of diagenetic features. There are five (5) descriptive units allowed.

Diagenesis	Code
Carbonates	
Solution structures (N/S)	00
Stylolites	01
Corrosion zone	02
Vugs	03
crystal lined	04
salt plugged	05
anhydrite plugged	06
Pisolites	07
Vadose weathering	08
Mottling	09
Ooliclasts	10
Stromatactis	11
Neomorphism (recrystalization)	12
Replacement dolomitization	13
Pervasive dolomitization	14
Selective dolomitization	15

Evaporites	
Solution structures	16
Recrystallization	17
Flows & deformation structures	18
Replacement	19
Clastics	
Replacement	20
Sericitization	21
Chloritization	22
Solution	23
Kaolinization	24
Grain overgrowth	25
Other	26

Part 9

Chemical Description Code — cols: 192-219

The following is a listing of elements and some ions. They cannot be combined in the code. There can be eight (8) descriptive units. The list is not intended to be exhaustive.

Chemical Constituent	Code
H	01
He	02
Li	03
C	04
N	05
O	06
F	07
Ar	08
Cl	09
S	10
Si	11
Al	12
Mg	13
Na	14
K	15
Ca	16
Ti	17
Mn	18
Fe	19
Ni	20
Cu	21
Zn	22
Te	23
Sb	24
Ag	25
Pb	26
Zr	27
Sr	28
Rb	29
Ba	30
Au	31
Hg	32
U	33
I	34
$CO_3^=$	35
$PO_4^=$	36
$CrO_4^=$	37
NO_3^-	38
$SO_4^=$	39
$SiO_2^=$	40
NH_4^+	41
Other	42
Other	43
Other	44
Other	45
Other	46
Other	47

Part 10

Hydrocarbon Description — cols: 220-227

The following is a brief listing of petroleum and gas occurrences. There is room for two (2) descriptive units for each defined interval.

Hydrocarbon Listing	Code
Gas	00
Oil	01
Gilsonite	02
Bleed	03
Show	04
Stain	05
Fluoresce	06
Odor	07
Drill stem test	
Gas	08
Oil	09
Oil cut mud	10
Gas cut mud	11

Part 11

Porosity Description — cols: 228-235

The following is a classification of porosity. The classification is in two parts: type and size. The descriptive unit changes to allow for the type and size as follows:

1st column - lithology identification as before (see Definition Codes, part 4)

x			

2nd column - porosity type (see list following discussion)

	x		

3rd column - porosity size (see list following discussion)

		x	

4th column - percentage code (see Definition Codes, part 5)

			x

example: 27% megavugular porosity in primary lithology.

1	8	1	2

descriptive unit

1 = primary lithology
8 = vugular
1 = mega
2 = 27% code

Porosity Description	Code
Porosity type	
interparticle	1
intraparticle	2
intercrystal	3
moldic	4
fenestral	5
growth-framework	6
fracture/breccia	7
vug/channel/cavern	8
burrow/boring	9
Porosity size	
megapore (4mm-256mm)	1
mesopore (1/16mm-4mm)	2
micropore (< 1/16mm)	3

Part 12

Permeability Description Code — cols: 236-240

The permeability description does not conform to the descriptive unit format. The permeability in milidarcys is entered in the five (5) allowed spaces. There is an assumed decimal point between columns 239 and 240 (measurement to tenths). If measurement is not that accurate, enter a zero in column 240.

example: permeability of 27.5 md.

cols	236	237	238	239	240
	0	0	2	7	5

example: permeability of 260 md.

cols	236	237	238	239	240
	0	2	6	0	0

STRATIGRAPHIC CODES FOR THE MICHIGAN BASIN

System	Code	Series	Code
Quaternary	700	Recent	702
		Pleistocene	701
		Pliocene	705
		Miocene	704
Tertiary	650	Oligocene	653
		Eocene	652
		Paleocene	651
Cretaceous	600		
Jurassic	550		
Triassic	500	Upper	503
		Middle	502
		Lower	501
Permian	450		
Permo-Carboniferous	410		
Pennsylvanian	400	Conemaugh	405
		Pottsville	402
Mississippian	350	Chesterian	354
		Meramecian	353
		Osagian	352
		Kinderhookian	351
Mississipian-Devonian	310	Bradfordian	305
Devonian	300	Chautauquan	304
		Senecan	303
		Erian	302
		Ulsterian	301
Silurian	250	Cayugan	253
		Niagaran	252
		Albion or Alexandrian	251
Ordovician	200	Cincinnatian	203
		Champlainian	202
		Canadian	201
Cambrian	150	Croixian	153
		Albertan	152
		Waucoban	151
Precambrian	100		

(from Briggs and Briggs, 1974)