LOG/MATE ESP - A FOURTH GENERATION LANGUAGE FOR COMPUTER AIDED LOG ANALYSIS

by

E.R.Crain, P.Eng.
D. Jaques
K. Knill
K. Edwards

D & S Petrophysical, a Division of
D & S Petroleum Consulting Group Ltd.
Calgary, Alberta, Canada

ABSTRACT

LOG/MATE ESP is a fourth generation, menu driven, integrated command language for data base management, calculation, and display of scientific or engineering data. It has been specially adapted to meet the needs of log analysts, geologists, and reservoir engineers by paying attention to the interactive graphics, data entry, computation, report generation and communications functions. The system is in an evolutionary stage as we learn more about its inherent capabilities.

The key concept is the use of data files, accessible by the user, to contain much of what is normally considered to be program code. These files include data record descriptors, plot and report descriptors, as well as all mathematical algorithms and routines. They can be modified or augmented by the user to adapt the system to his or her own needs. The system interprets the contents of these records to perform the appropriate task, much as existing third generation languages interpret hard-coded Basic or Fortran programs.

We expect LOG/MATE ESP to be the foundation system for research and development of fifth generation software for log analysis, as well as other areas, for the next decade.

INTRODUCTION

The basic problem with most computer aided log analysis programs is that they are hard-coded to a specific design on a specific machine. This results in an inflexible program which cannot easily be adapted to changing needs of users, new logging tools or methods, or more advanced hardware. Even adding a new curve name or interpretation parameter is costly in many systems. Adding new math algorithms, permanently, requires a cooperative programmer familiar with the system and a computer department manager or supplier willing to allocate resources to the task. Only major requests that suit the needs of many users get satisfied. The once only, lets try it and see what happens stuff, never gets to a useable stage. Systems which do provide some flexibility still do not, as a rule, allow the user access to the guts of the system.

We have addressed this problem by devising an integrated fourth generation

computer language for data base design and display, interactive colour graphics, mathematical processing, and report generation, most of which is accessable to the user, or at least a high level professional user. This allows one to adapt, add to, or change data record formats, screen presentations of data, log analysis methods and algorithms, as well as depth, crossplot, and printed report formats to suit the job at hand or the corporate style. The original software designers' view of the world can therefore be adjusted to conform to reality. The result is a software package similar in concept to an integrated spreadsheet program, like Lotus 1-2-3 or Symphony, but on a much grander scale, and designed with scientific users in mind.

Computer languages, by the way, are classified by level or generation, roughly as follows:

1. native machine language (specific to a particular machine)
2. assembler language (specific as above)
3. high level language, like Fortran, Basic, or Cobol (semi-portable)
4. very high level language, like Visicalc, Lotus 1-2-3, and some graphics command languages (quite portable)
5. extremely high level language, like Lisp and Prolog (very portable)

We place LOG/MATE ESP in the fourth category because of its integrated spreadsheet nature, and its relative portability.

The key concept in LOG/MATE ESP is that we consider most things to be data that are normally considered to be program code. For example, data file formats, printout formats, plot presentations, and log analysis routines are usually hard-coded in Fortran, Basic or machine language. We decided instead that the description of all these items was to be in data files to be interpreted and used by LOG/MATE ESP to produce the results requested. Thus even the math routines which do the log analysis are input into data files. They can be created and added to the system at any time by anybody willing to follow a few simple mathematical and nomenclature rules. These are not stringent rules, and other functions in the system require no more ability than the casual home computer hacker may have.

The best part is that the user does not have to know the intricacies of the data base or the languages that drive the various components of the system. With minor exceptions in the report generator, the languages are entirely menu driven, and no memory or lookup are required. Some of the data base manipulation features, however, are not yet ready to release in this form until further improvement in this area has been completed.

The concept is portable to many high speed desktop computers. It is currently available on Hewlett-Packard Series 200 (HP-9816 or HP-9836), and is being translated to the IBM/PC-AT and DEC Micro-VAX.

THE DATA BASE

Most data bases are either hierarchical or relational. LOG/MATE ESP uses a new concept called an attached records data base (ARDB). The data base designer

decides which records are most likely to need data from each other in the context of the application program, and specifies this connection. All records of the same type are automatically attached to each other. These attachments are called links, and a series of links is called a chain or a path. Each record carries chaining pointers which carry the information needed to travel forward or backward along any path which is attached to the record.

To illustrate, a well record is attached to all other well records on the data base, as well as to a client record, a project or batch record or both, one or more pool records, one or more log header and core header records, and well history records. Log headers are attached to the log curves. All the records attached to each other constitute a file or data base. The data manager allows more than one data base on a system.

The concept of client, project, pool, and batch have been introduced to minimize the search time in catalogs by users. You merely have to locate your project with a cursor in the project catalog to get started. Alternatively, you could start in any of the categories mentioned to get into your data. The client concept could be a department in an oil company or an individual user. Definitions of these and other record types are illustrated in Figure 1 and a typical arrangement of attached records is shown in Figure 2.

The data base language allows a user to specify a record type from a menu at the keyboard, and the data base manager will load the correct record which is attached to the current record. If more than one exists, a catalog of attached records of this type will be presented, from which the user can choose the desired record. For example, a user may wish to edit a gamma ray log attached to the current well record. If more than one gamma ray curve existed, the catalog would offer a choice. Catalogs are also used to locate records not attached to the current one, as one might need to do to start processing a new well.

Alternatively, an application program may need an interpretation parameter from a record containing such data. In this case the path to the correct element in the record would be specified in the program code. If the record attachments were modified at some later date, the program may no longer be able to find the data. To solve this we have devised a series of dictionaries in which all data element names are listed. The application program can check here to find out where to access the required record. If a parameter is needed but not available in a dictionary, the data manager will prompt for a value to allow the current process to finish. The user should then enter the parameter permanently into an appropriate data record.

The dictionaries serve another valuable purpose. They also contain the full English name of a parameter or log curve, the legal abbreviation for it, the name of the units of measurement in both English and Metric units, and the conversion factor from English to Metric units. Thus graphs, reports, or data tables can contain appropriate, consistent, and meaningful headings or labels, and data can be converted from one system of units to another at the press of a button. Again all this information is in data records accessable to the user, so he could change ranges, or add, change or delete parameters or curve names. Since there are no trace numbers, channel numbers, or array locations in this form of data base, there is no need for the user to remember or look up such esoteric computereze while modifying the data structure.

Record descriptors break a record into lumps of similar data types. For example a well record contains two lumps. The first contains three elements of string type data with the full well name, the full well location, and the name of the person entering this data. The second lump has seven real type numeric fields containing datum elevation and other pertinent depth values. Record descriptors also contain the units of measurement in both English and Metric, as well as the permissable range of the data and a default value in both sets of units.

Records can be lengthened or shortened by changing the record descriptor, and suitably editing old data. A lump can be a string or numeric array. Arrays in lumps are self-dimensioned, facilitating the addition of interpretation parameters to a previous list, or additional log curve data to an existing file.

Records can be viewed and edited on the CRT, or can be filled by reading LIS or BIT tapes, or by using a digitizer, or by reading files from a remote site. Data communication to a remote computer data base is also supported, in addition to networking on a shared resource basis.

The data base manager looks after handling the screen in two different ways, depending on the structure of the record. If the record can be displayed conveniently within the confines of the available screen width, it presents the element names, values, and units in an attractive format automatically. If data is represented better as a table or array of data, then the manager formats accordingly, using the screen as a window on the larger array, much as Visicalc or Lotus 1-2-3 treat data editing of such arrays. Cursor movement slides the window over the array as desired.

Both data entry and data editing are fast and self explanatory. Since full English names for all parameters are used, no knowledge of cryptic abbreviations is necessary. Virtually no typing is required except for parameter values. Function keys or cursor movement in a catalog or table answer every question posed by the system.

Other user friendly LOG/MATE ESP data manager functions include an English/Metric toggle switch, and a Printer/Screen/Disc file switch. These can be set at any time to direct the manager to take the appropriate action.

Extensive use is made of context sensitive help files, probably more so than any other scientific system. There are three kinds of help files. The first is Entry Help. When in an entry mode, these files tell you how to fill out the record on the screen or explain the rules for constructing algorithms or reports. Key Help files are available to tell you what each function key will do before you push it. File Help is used to explain the method or purpose of a record, such as a report descriptor, algorithm, or analysis routine. Since these records are the ones most often created or modified directly by users, it pays to keep the help files current with reality. All help files can be edited or expanded by the user so that personal usage or hints for less experienced operators can be incorporated into the system. A typical key help file for the Mode Switch menu is given in Figure 3 and a file help menu for a particular algorithm (SWbvw) in Figure 4.

Hard copy of any or all help files can be had at the push of a button, so no documentation manual on the system is needed. However, a demo or walkthrough is helpful before embarking on some of the more difficult functions.

The record structure and data manager are designed to relieve another burden, namely that of processing or reprocessing a group of wells in a batch or project. For example, after a few wells have been processed interactively (one at a time), they could all be recomputed with a new water resistivity and replotted as a group. Then the balance of the wells in a project could be processed with the same runstream, individually checked for improvements, and batched again for final plots and printouts. The possibilities are endless.

A crucial concern of many with respect to a data base of this type is accidental or intentional injury to data which is really program code in disguise. This concern is alleviated by two builtin safeguards. One is that the first record of any type cannot be deleted under any circumstances. These are the default records for every function in the system.

Passwords also protect many features. User access, for example permits use of any feature, without the ability to change record descriptors, algorithms, routines, or report formats. Manager access permits more advanced status, such as modifying or adding algorithms, routines, report and plot layouts, and help files. System access provides the ability to use all the power of the various components to the fullest extent, presumably to create new products for lower level staff to work with.

An extensive data base suited to log analysis is delivered with the software, which includes demonstration data.

THE ALGORITHM PROCESSOR

The basic element of mathematical processing within LOG/MATE ESP is the algorithm. An algorithm is a series of mathematical steps which is intended to produce a single desired numerical result, such as the volume of shale from the gamma ray or the porosity from the sonic log. A routine is a series of consecutive algorithms which produce all the results desired from one computation run, for example a shaly sand analysis or a tar assay program. An algorithm is analagous to a subroutine or function, and a routine is similar to a program module, in conventional programming parlance.

A typical algorithm (for SWbvw) can be seen in Figure 5, exactly as seen on the CRT. The associated help file for this routine was shown previously in Figure 4. Two personalized routines are presented in Figures 6 and 7. They illustrate the different approaches two analysts might take to solve the same problem, and emphasize why hard-coded log analysis programs might create frustration for these two people.

Single algorithms or complete routines can be run interactively under user control, or under runstream control by the data manager. The algorithm processor decodes the algorithms, enacts the processes defined, and places the results into appropriate records. The processor uses the dictionaries to find the required data in the current records and those attached to them. The processor can distinquish between log curve data and analysis parameters because the data manager can provide this information. If required data cannot e found, the user is prompted to supply it.

Most processes are treated as matrix or vector operations, which speeds the calculations enormously. All input and output data curves are scaled between -32000 and +32000 to preserve storage space and reduce computation time. This range still provides adequate resolution for any conceivable input or output from a log analysis, since no known log can make measurements with five digit accuracy.

Because of this range limitation, it is important to design algorithms which do not create a larger range, or precision will be degraded. For example, the function 1 / PHI^2 will create very large numbers as PHI approaches zero. If PHI ranges from 1 to 100, the answer ranges from 1 to 0.0001 and you get precision to about 0.00002, which is quite acceptable. If the range was between 0.0001 and 100 then the answer is between 10,000,000 and 0.0001 with a precision of plus or minus 2000, clearly not good enough in the present context. The algorithm should be written to avoid these situations, either by testing input data for acceptable range, by clipping the result between acceptable limits, or by storing a different function (eg. conductivity instead of resistivity) and decoding it later when needed. The problem exists in all computers and calculators to some degree, but in LOG/MATE ESP the user gets to decide how to handle it.

Algorithms are entered into the data base much as you would write them on paper, that is in conventional mathematical notation. All analysis parameters used should be declared in the appropriate dictionary, as well as all log curves input or output from the algorithm. Consistency in spelling of these variable names is recommended to conserve space and reduce user confusion. The dictionaries are designed to aid in this regard.

If branching is required within an algorithm, the structured IF...THEN...ELSE...END IF sequence is permitted. All normal mathematical functions and logical operators found in Fortran or Basic are provided, along with some which are only available in higher level languages, such as Visicalc. These include SUM and PRODuct of vectors. See Figure 5.

Algorithms are assembled into routines by the user and stored for future use or modification. An algorithm may appear in any number of routines, and logic switching is allowed to handle conditional problems such as bad hole, gas corrections, or missing data. See Figures 6 and 7.

User defined help files are attached to each algorithm and routine, so that other users know what the function is supposed to do, and where the data comes from. Comments can be interspersed in algorithms to make them more readable, but cannot be placed within a routine. See Figure 4.

A runstream, created by walking through an actual processing sequence, is akin to a routine of routines, although other functions such as printing and plotting can be invoked in addition to computation. The kinds of functions which could be included in a runstream, their extent, and sequence are defined in Figure 8. The relationship to the other functions of the analst and the system are also shown.

Note that the algorithm processor and the data base are not limited to depth dependent data such as logs, but can work on time series or frequency series data such as reservoir engineering cash flows, seismic two way time, or frequency transforms of seismic data. Also it should be evident that they are

not limited to conventional open or cased hole logs. Mud logs, gas logs, geological descriptions, core data, MWD logs, palynology or paleontology data are equally at home in the system, without any changes required to the basic program code. An extension to allow use of algorithms on spatial data such as maps and crossplots is being designed.

A very complete set of algorithms and routines are delivered with the software, and many users will never have to create new ones.

THE REPORT GENERATOR

Two kinds of reports are available from LOG/MATE ESP. Printer dumps of any individual record in screen image format provide rapid hardcopy for use during analysis. These are formatted automatically by the data manager, based on the record descriptor, and are attractive enough to be included in many reports.

Tabular and text reports can be generated from data in the data base with user defined formats. A standard log analysis report, for example, can gather the correct pool, project, well name, and hydrocarbon volume summary and place them in the correct context within the body of the report. The runstream can place depth plots or crossplots on appropriate pages to automate production of repetitive tasks.

The report generator language allows you to specify the position and content of headers, footers, titles, pagination, data, and text on the page, as well as the page size, line spacing, character size, and display enhancements as in most sophisticated word processors. Although editing capabilities are limited to insert and delete characters and lines, more effort in this area is underway.

The presentation of columnar data is especially flexible, with the ability to specify column width, numeric format, decimal places, and the column headers. These latter items could be abstracted from the data base or entered manually. Thus report formats can be made to a standard corporate style, and yet present only the data available for a particular well. The table report generator also has access to the algorithm processor. Data can be computed on the fly if it is not already in the data base. Routines, however, cannot be accessed this way.

Path names are a very important concept to master in learning to control the LOG/MATE ESP report generator. A path name is simply a way of telling LOG/MATE where to go to look for the data that you want. A path name has the following syntax:

(record name)\(lump name)\(field name)\(parameter)

where:

- (record name) describes the type of data record, ie, LOG
- (lump name) describes the section of the record, ie, LOG DATA
- (field name) describes the field within the lump, ie, CURVE VALUES
- (parameter) describes the element of the field, ie, GR

In the above example all of the names are exactly as they are found on the data base printout except the (parameter) which depends on local data.

The parameter term acts in a somewhat different manner than the others in the path name. For example LOG \ LOG DATA \ CURVE VALUES \ GR will access the entire GR LOG, while CONSTANTS \ CONSTANT VALUE \ CONSTANTS \ GRO would look in the constant names until it found the name GRO and then print the VALUE that is associated with this name.

If you do not have room for the entire path name you may use just the middle two parameters - the trick is to count over the number of elements from the beginning of a data lump and use this with the type. For example, POOL \ S#1 \ 2 is the same as POOL \ IDENTIFICATION \ POOL TYPE (OIL or GAS and POOL \ S#3 \ 1 is the same as POOL \ WATER DATA \ SOURCE OF DATA. Even the full text of a help file or a math algorithm or routine could be brought into your report.

The default path is for logs. To access a log you only need to use the correct curve name, which appears in the dictionary.

Commands are special strings of characters that the computer recognizes as items that should not be printed directly, but that some action must be taken instead. Path names are a type of command with a very special syntax and use.

Commands available in the text and identification sections of a report are:

@TOP DEPTH - prints the top depth of the current report
@BOTTOM DEPTH - prints the bottom depth of the current report
@PAGE# - prints the page number of the current report
@DATE - prints the current date as part of the report
@TIME - prints the current time as part of the report
@IDENTIFY - prints the current user's name as part of the report
@ENTER - allows keyboard entry of comments or text anywhere in a report

These commands are available in the interactive graphics language as well.

Other commands are used to create columnar data or to operate on columns of data within the report. These are:

@DEPTH - prints the depth of the data specified in the PATH field
@SUBSEA - prints the subsea elevation of the data specified in the PATH field
@LINE# - prints the current line number of the report
@PRINT - prints the text in the path field without modification
@CUTOFF'(algorithm name) - runs the cutoff algorithm specified, applies the cutoff to a specified column, and prints the cutoff value in a column
@COMPUTE'(algorithm name)'(presentation) - runs the algorithm specified and prints the results in a column. The optional presentation parameter can be blank, AVErage, or CUMulative.

@SUM - prints the sum of values in a column,
not counting values flagged by cutoffs
@AVERAGE - prints the average of the values in a column
not counting values flagged by cutoffs
@DIFF - prints the difference between the,
first and last value in a column, usually used to show net
accumulations in pay zones
@NET - prints the number of values included in an @SUM,
@AVERAGE, or @DIFF command
@COMBINE - prints the bottom depth of a layer by adding
top depth and interval thickness
@CODE'(code table name) - prints the entry in the code
table corresponding to the entry in the column
@SPELL - prints the full name of the specified data,
by looking it up in the dictionary
@UNITS - prints the full name of theunits of the data,
by looking it up in the dictionary

These last two commands can also be used in the text and
identification sections of the report. Thus generic reports

Text files are entered by simply typing in the text you want to have printed. The format is exactly as you see here, except that you can use embedded PIPES to include information about the current well. For example, if you were to type the text "Enclosed are the results for ! WELL \ NAME INFO \ FULL WELL NAME ! ", then the printout would be: "Enclosed are the results for ABC WELL #1", if the well name is as indicated.

Help files are always a push button away, so the syntax, while simple and minimal, is readily explained. Knowledge about the contents of each record is also available through separate help files generated automatically by the data manager. In addition, help files for each report format created by a user provide extra documentation of purpose and intent.

The common print out formats for detail listings, summary listings, and typical final reports are included in the data base. Examples of reports are not shown here as they tend to look very much like log analysis printouts from earlier LOG/MATE systems. Users can, however, make their own style of report very easily.

THE INTERACTIVE GRAPHICS PACKAGE

Everyone claims to have interactive graphics in their log analysis system. Most of them mean that you can generate a user defined plot (within limits) in a short time and recreate it differently, quickly, by some keyboard commands. However, interactive graphics in LOG/MATE ESP involves more than that. By interactive we mean the following:

On depth plots we can:

- rescale a curve and see the result without replotting the entire plot
- pick the rescale parameters with a cursor on the screen
- stretch and squeeze curves with respect to a reference curve in the same fashion
- edit and redraw any curve, and erase the old one
- overlay one curve on another with cursor movements
- add or delete a curve
- undo any of the previous functions and get the original plot back again
- place the cursor on a point and identify its curve name and data value
- place comments and annotation anywhere on the display
- place DST, perforation, core and production data on the plot from the keyboard
- turn the annotation on and off to improve clarity when needed
- select analysis parameters by cursor movements
- scroll the log display up and down the screen in real time; ie, as fast as the eye can see or the hand can move
- store the changes we have made back into the data base

On crossplots we can:

- identify depth and data values for all points under the cursor
- delete points and rerun the regression analysis
- turn the regression lines and other overlay lines on and off
- change the image from CRT format to plotter format, to preview final products, and back again
- place comments and annotation or draw shapes anywhere on the screen
- pick analysis parameters by cursor movement

All these features have the usual access to the help files, algorithm processor, and data base functions as described earlier, and in living colour, which was pioneered in LOG/MATE and LOG/MATE PLUS. Full size and half size plots can be dumped to the printer at the touch of a button, and hard copy colour plots are just a few more keystrokes.

The layout of all plots is contained in the data base, and can be adapted to any situation. Three track, four track, or multi track presentations can be defined, with depth tracks, borders, titles, scales, curve names, axes names, headers, footers, logos, tics, depth lines, line types, colours, and annotation where and how you want them. This graphics command language is completely menu driven, so knowledge requirements are very low.

As a result, very sophisticated and flexible format plotting can be achieved at

a very low cost, very quickly. For example we designed a 19 track presentation for palynology data in about 10 minutes. A composite well history log, which most oil companies create by drafting and photo reduction, can be designed in similar fashion and become a permanent part of the log analysis production process.

Another unique feature is the concept of shapes, analagous in some ways to custom characters in some systems. The LOG/MATE ESP graphics commands allow you to store custom shapes in the data base for shading and coding lithology, personalized logos, and annotation enhancements.

Again, the standard three and four track plots and numerous crossplots, histograms,and Holgate plots are predefined in the data base delivered with the system. Results from these plots look much like those from LOG/MATE or LOG/MATE PLUS with enhanced shading and, of course, flexible presentation format.

ADVANTAGES AND DISADVANTAGES

Many of the advantages of a fourth generation command language have been described above. The concern over safety, data and system integrity, and misuse are real. The language interpreters and operating system are relatively secure, but the algorithms, plot formats, and print formats are essentially open code for anyone with high enough password status. Thus there are no secrets, and kludge fixes are rapidly exposed.

The corollary is that users who would add to or modify the open code run the risk of failure to achieve desired results due to lack of talent, training, or error. Errors can propagate into unexpected areas of the data base. It may be difficult to debug problems because any of the operating system, data, or user defined record contents could be at fault.

The system can be slower than conventional hard-coded software for certain operations due to initial setup time, for example in defining a new computation routine, or complex plot or report. The system also does more work to accomplish a required task due to the distributed nature of the data base and the fact that computations are interpreted instead of compiled. This latter problem can be solved on certain machines at the expense of a slightly more complicated creation process.

The main problem is that of learning any new language. The more you have worked in one language, the more difficult it is to work in a new one, at least for a while, and you always tend to retain the accent and flavour of the old one. Increased complexity is another people problem. We call it the Swiss Army knife syndrome - anyone can use a steak knife, but it takes more skill to eat with a Swiss Army knife.

The final and most obvious problem is that such software will not run on the smaller, cheaper home and office computers. As it stands now, LOG/MATE ESP requires 1.5 to 2.0 megabytes of memory and 20 to 40 megabytes of hard disk to run effectively.

W

CONCLUSIONS

LOG/MATE ESP is a fourth generation, menu driven, integrated command language for data base management, calculation, and display of scientific or engineering data. It has been specially adapted to meet the needs of log analysts, geologists, and reservoir engineers by paying attention to the interactive graphics, data entry, computation, report generation and communications functions. The system is in an evolutionary stage as we learn more about its inherent capabilities.

The key concept is the use of data files, accessible by the user, to contain much of what is normally considered to be program code. These files include data record descriptors, plot and report descriptors, as well as all mathematical algorithms and routines. They can be modified or augmented by the user to adapt the system to his or her own needs. The system interprets the contents of these records to perform the appropriate task, much as existing third generation languages interpret hard-coded Basic or Fortran programs.

We expect LOG/MATE ESP to be the foundation system for research and development of fifth generation software for log analysis, as well as other areas, for the next decade.

FIGURE 8 : FLOW DIAGRAM FOR COMPUTER AIDED LOG ANALYSIS

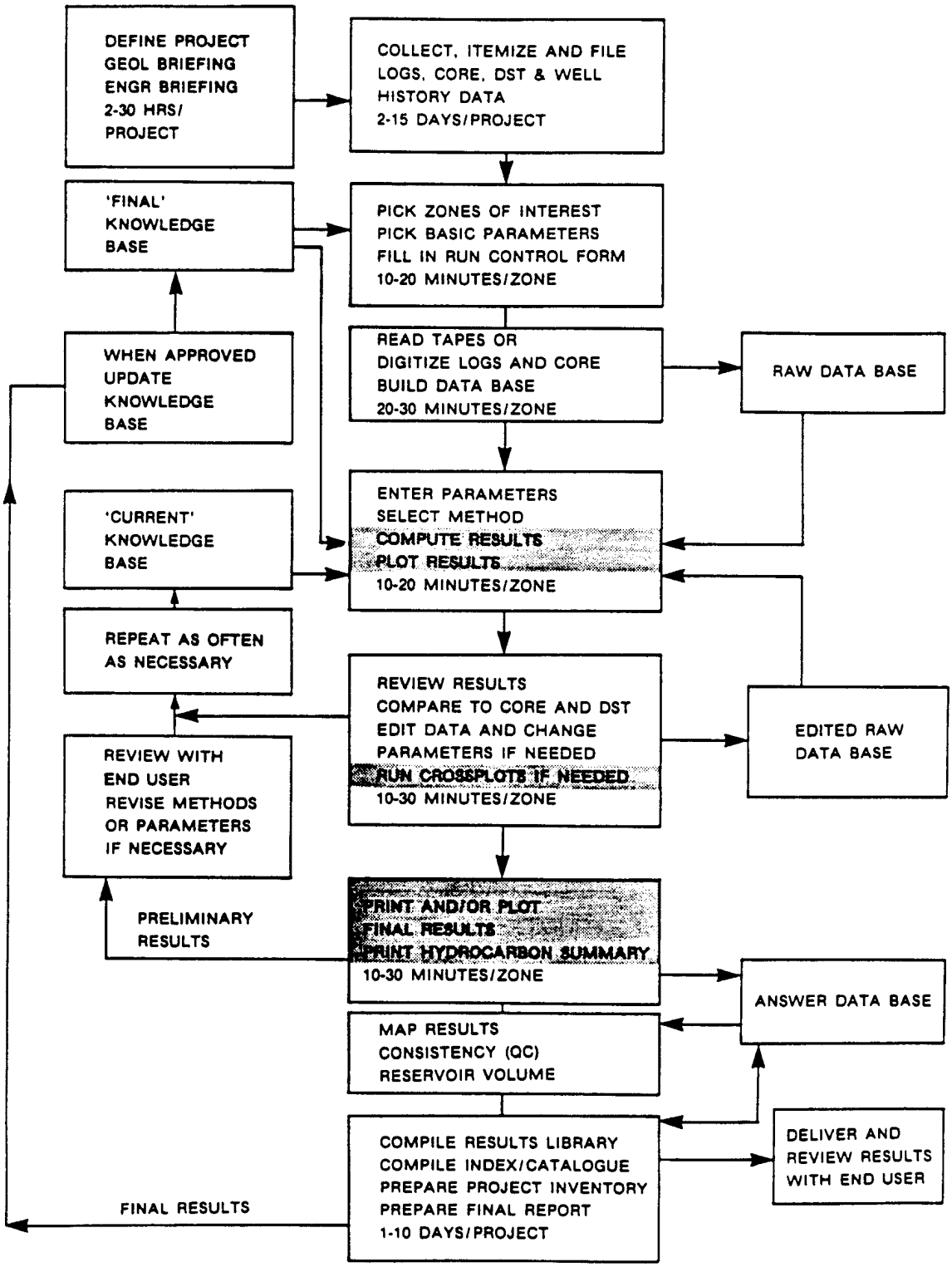


FIGURE 1a : SPATIAL DEFINITIONS FOR A PROJECT

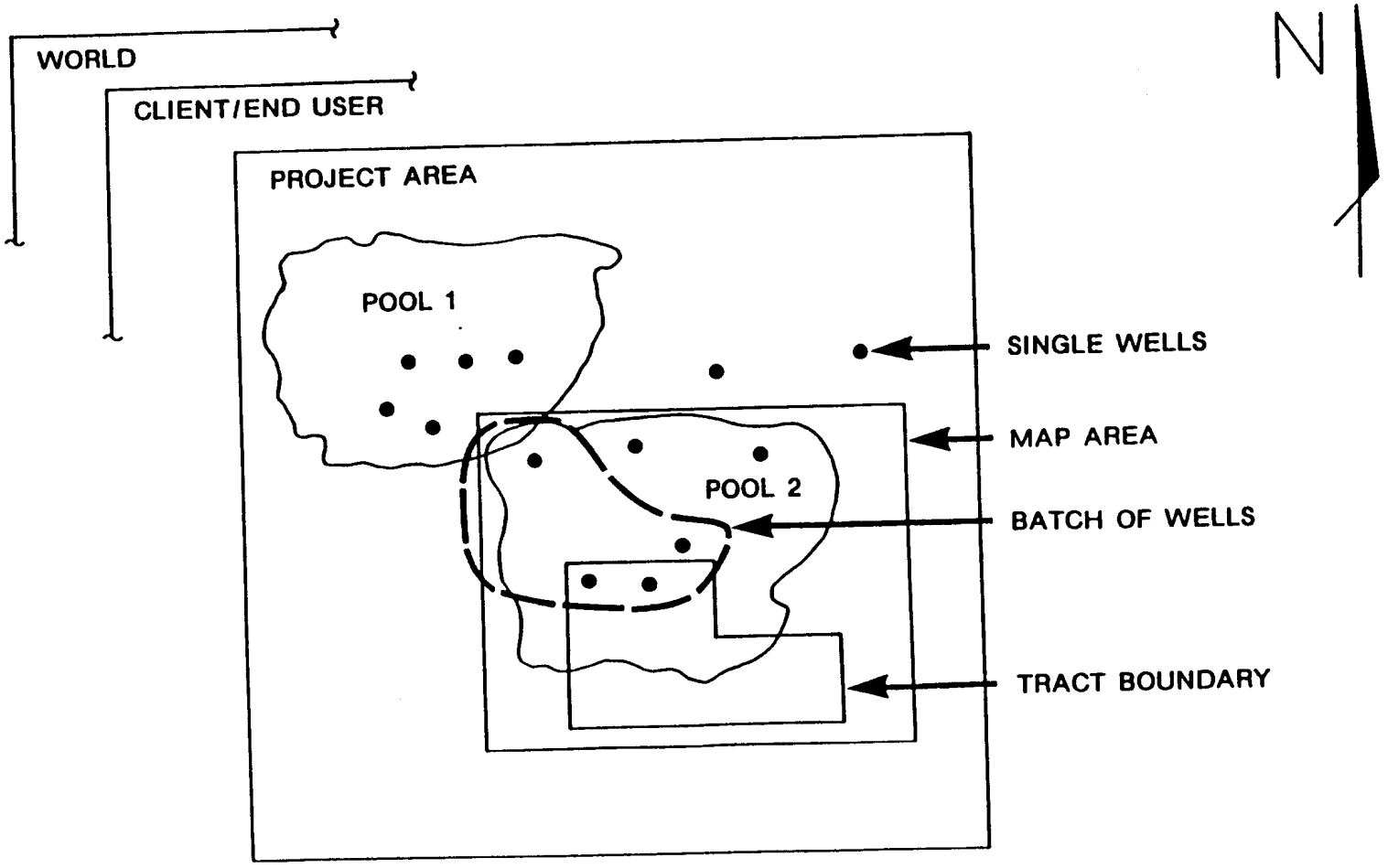


FIGURE 1b : DEPTH DEPENDANT DEFINITIONS FOR A WELL

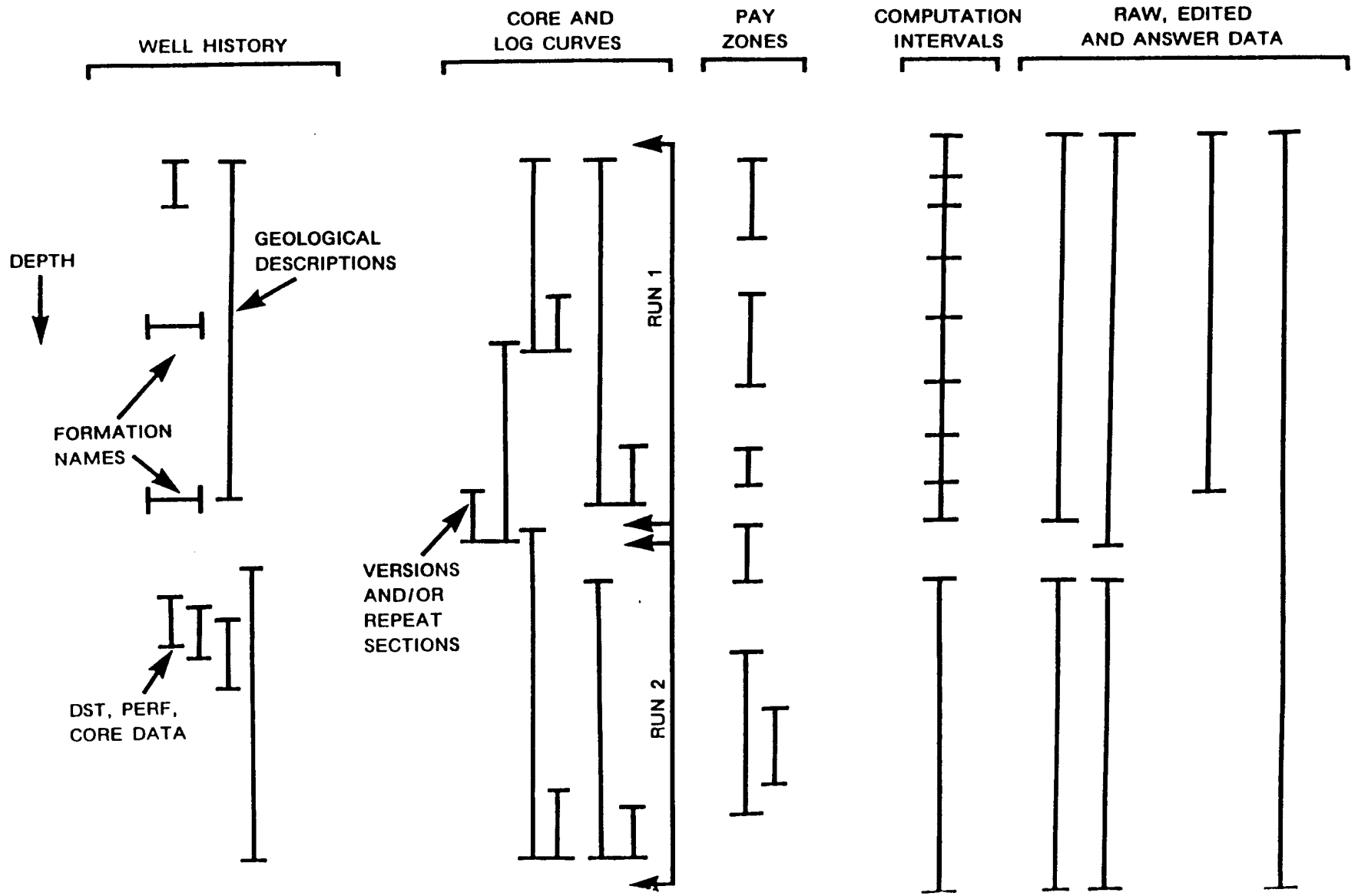
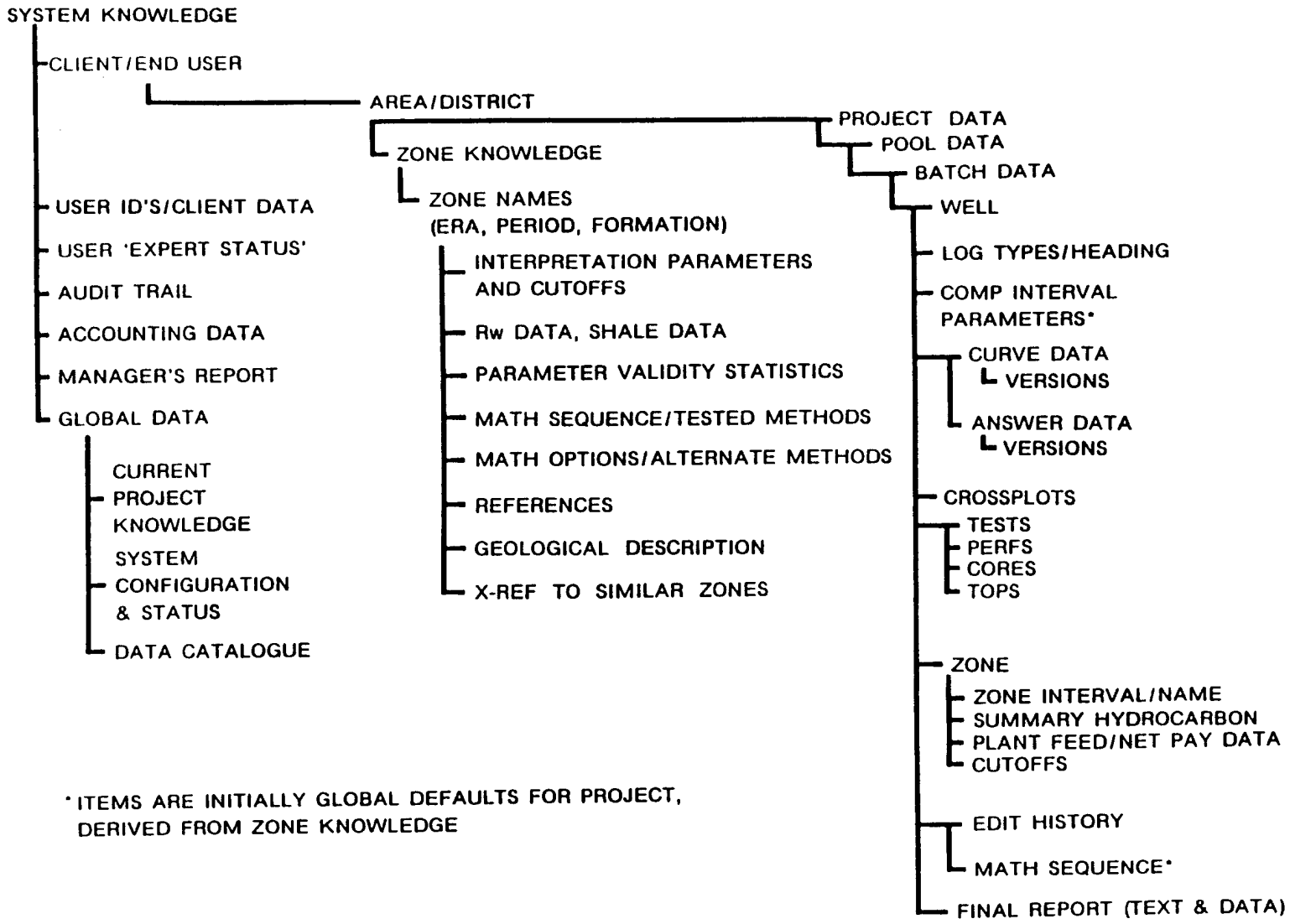


FIGURE 2 : A TYPICAL DATA BASE STRUCTURE



*ITEMS ARE INITIALLY GLOBAL DEFAULTS FOR PROJECT, DERIVED FROM ZONE KNOWLEDGE

FIGURE 3 - KEY HELP : MODE

KEY INFORMATION

USER SELECTION -	opens the entire database to use, this means you may combine information from any well in the database in doing all of the functions in LOG/MATE, for example, crossplotting logs from different holes, the disadvantage is that you have to answer a lot of questions regarding where data is to come from
BY WELL -	possibly the most common mode of operation, where only the data directly related to the current well is used, this means that LOG/MATE can automatically find most of the information required without asking you many questions
BY POOL - BY CLIENT -	BY PROJECT - BY BATCH - these modes are equivalent to BY WELL except that they automatically step through the LIST of WELLS one at a time
USE RUNSTREAM -	will repeat a series of operations automatically

W

FIGURE 4 - FILE HELP : SWbvw

FILE INFORMATION

ALGORITHM SWbvw	computes WATER SATURATION based on BULK VOLUME WATER method
INPUT LOGS	PHIebvw, PHItbvw from ALGORITHM PHIbvw RESDc2 the corrected DEEP RESISTIVITY LOG
CONSTANTS	A,M,N constants in the WATER SATURATION FORMULA RW@FT - water resistivity at formation temperature PHIDDC - apparent density porosity of DRY CLAY PHIDSH, PHINSH - apparent density and neutron porosity of shale
CALCULATED CONSTANTS	PHINDC - apparent neutron porosity of DRY CLAY BVWSH-percent of shale that is water
OUTPUT	SWebvw - log of WATER SATURATION in the effective porosity SWtbvw - log of WATER SATURATION in the total porosity
OPTIONS	if ?TRACE is 1 then a value of 4 is placed in a log called TRACEsw for use with code table SW OPTIONS to printout how the SW was derived

W

FIGURE 5 - ALGORITHM : SWbvw

ALGORITHM

```
PHItbvw = PHItbvw/100  ! ALGORITHM TO CALCULATE SW BY BULK VOLUME WATER  METHOD
PHIebvw = PHIebvw/100 ! NOTE : POROSITIES MUST COME FROM PHIbvw  ALGORITHM
VSH = VSH/100
CONST1 = 100(100-PHIDDC)*(100-PHINSH)/(100-PHIDSH)                    ! PHINDC
CONST2 = ((CONST1*PHIDSH-PHIDDC*PHINSH)/(CONST1-PHIDDC))/100   ! BVWSH
CONST3 = (CONST2^M)*RSH/A                                             ! RWSH
IF PHItbvw> 0.02          ! STOPS THOSE NASTY DIVIDE BY ZERO THINGS
RESULT2 = VSH*CONST2/PHItbvw                                          ! BVW
RESULT3 = A*RW@FT*CONST3/(PHItbvw^M)/(CONST3-RESULT2*(RW@FT-CONST3))   ! RO
RESULT3 = MAX(RESULT3,0.1)
RESULT4(RESULT3/RESDc2)^(1/N)
ELSE                      ! ELSE WE HAVE NO POROSITY SO DON'T  SWEAT SW
RESULT4 = 1               ! JUST CALL IT 100 % AND LET IT GO AT THAT
END IF
IF PHIebvw>0.02           ! SAME STORY ON THE ZERO'S
RESULT3 = (PHItbvw/PHIebvw)*(RESULT4-RESULT2)
ELSE
RESULT3 = 1
END IF
PHIebvw = PHIebvw*100
PHItbvw = PHItbvw*100
VSH = VSH*100
IF ?TRACE
TRACEsw = 4               ! 4 IS BULK VOL WAT IN WATER SAT CODE TABLE
END IF
SWtbvw = 100*RESULT4
SWebvw = 100*RESULT3
```

W

FIGURE 6 - ROUTINES : KEN'S HALFWAY

ALGORITHM NAME	PUT RESULT IN
VSHgr	VSH
VSHbal	
PHINc	PHI
PHIbal	PHINc
PHIDc	
PHIxcl	PHI
PHIbal	
SXOs	SW
SWsmth	SXO
SWs	SW
SWsmth	
RHOma	
PHIsec	
MNlith	
VROCK3mn	
DCAL	
PERMp	PERM
fPHIxS	

FIGURE 7 - ROUTINES : BILL'S HALFWAY

ALGORITHM NAME	PUT RESULT IN
VSHgr	VSH
VSHbal	
PHIxss	PHI
PHIbal	
SXOs	SW
SWsmth	SXO
RESDc2	
SWs	SW
SWsmth	
RHOma	
PHIsec	
VROCK2d	
ANHYtrig	
DCAL	
PERMp	PERM
fPHIxS	

W