A STATISTICAL PROCESSOR FOR ANALYZING SIMULATIONS MADE USING THE MODULAR FINITE-DIFFERENCE GROUND-WATER FLOW MODEL

by Jonathon C. Scott

U.S. GEOLOGICAL SURVEY

Water-Resources Investigations Report 89-4159

Oklahoma City, Oklahoma

1990

DEPARTMENT OF THE INTERIOR

MANUEL LUJAN, JR., Secretary

UNITED STATES GEOLOGICAL SURVEY

Dallas L. Peck, Director

For additional information
write to:

District Chief
U.S. Geological Survey
215 Dean A. McGee, Room 621
Oklahoma City, Oklahoma 73102

Copies of this report can
be purchased from:

Books and Open-File Reports
U.S. Geological Survey
Federal Center, Bldg. 810,
Box 25425
Denver, Colorado 80225

CONTENTS

Page

Page

ILLUSTRATIONS

Page

TABLES

ATTACHMENTS

A STATISTICAL PROCESSOR FOR ANALYZING SIMULATIONS MADE USING THE MODULAR FINITE-DIFFERENCE GROUND-WATER FLOW MODEL

By Jonathon C. Scott

ABSTRACT

Many hydrologic studies of ground-water systems are conducted using a digital computer model as an aid to understanding the flow system. One of the most commonly used ground-water modeling programs is the Modular Three-Dimensional Finite-Difference Ground-water Flow Model (Modular Model) by McDonald and Harbaugh. This report presents a computer program to summarize the data input to and output from the Modular Model. The program is named the Modular Model Statistical Processor and is designed to be run following the Modular Model.

The Modular Model Statistical Processor provides ground-water modelers with the capabilities to easily read data input to and output from the Modular Model, calculate descriptive statistics, generate histograms, perform logical tests using relational operators, calculate data arrays using arithmetic operators, and calculate flow vectors for use in a graphical-display program.

CHAPTER 1
INTRODUCTION

Purpose

This report was prepared to aid ground-water modelers in their understanding of the results of ground-water flow simulations made using the Modular Three-Dimensional Finite-Difference Ground-water Flow Model (Modular Model) by McDonald and Harbaugh (1988). In the decade preceding publication of this report, the U.S. Geological Survey conducted a series of Regional Aquifer Systems Analysis (RASA) studies to evaluate the Nation's major aquifer systems (U.S. Geological Survey, 1980, p. 92). Each RASA study included the use of a digital computer ground-water flow model to complement the understanding of the ground-water flow system. The most commonly used ground-water modeling program has been the Modular Model.

Currently, the Survey is conducting several pilot studies before performing a NAtional Water Quality Assessment (NAWQA). Several of the NAWQA projects are studying ground-water flow systems and are using the Modular Model as a tool for testing hypotheses regarding the natural ground-water flow paths. The present report will help ground-water hydrologists to visualize these potential ground-water flow paths.

Recently, computerized methods for preparing the data input to the Modular Model have become available. An interactive data preprocessing program has been written (G.D. Rogers, written commun.,1988), and geographic information systems (GIS) have been used (Kernodle and Philip, 1988). Because arrays of data values that are used as input to the Modular Model are being computed rather than chosen and typed, a tool is needed to assist modelers in checking the logical consistency of the resulting data arrays. Furthermore, because less time is required to prepare the data input to the Modular Model, modelers have more freedom to examine the results of a simulation, check hypotheses regarding the flow system, and determine the adequacy of a simulation. This report describes the Modular Model Statistical Processor (MMSP), a computer program that can be used to facilitate the analysis of ground-water flow simulations made using the Modular Model.

How to Use This Report

The purpose of this report is to describe and document the MMSP program. Throughout this report, it is assumed that the reader is familiar with the documentation for the Modular Model. Much of the material presented in this report complements information presented in "A Modular Three-Dimensional Finite-Difference Ground-water Flow Model," and the information about the Modular Model is not repeated in this report. Therefore, it is important to have a copy of the documentation for the Modular Model handy when using this report. Copies of documentation for the Modular Model can be purchased from:

Books and Open-File Reports Section
Box 25425, Federal Center
Denver, Colorado 80225
Telephone: (303) 236-7476

A user wanting to quickly use the program should read chapter 1 and chapter 2 of the present report, as well as the sections of chapter 3 applicable to the features that will be used. Chapters 3 and 5 can be used for reference, when attempting to find documentation for specific capabilities or solving problems that occur when using the program.

The use of the word "altitude" in this report refers to the distance of a point above or below the National Geodetic Vertical Datum of 1929 (NGVD of 1929)--a geodetic datum derived from a general adjustment of the first-order level nets of both the United States and Canada, formerly called "Mean Sea Level of 1929" and referred in this report as "sea level". The word "elevation" in this report refers to the vertical distance between a point and some datum that is arbitrarily assigned or sea level.

Organization of This Report

Chapter 1 contains information about the this report, and a general description of the capabilities of the MMSP program.

Chapter 2 provides information about the design, installation, and use of the MMSP program.

Chapter 3 contains detailed information about the various commands that may be used to perform analyses of a ground-water simulation. Chapter 3 may be used to gain familiarity with the capabilities of the program and may be used as a reference guide for each command. Command descriptions consist of a discussion of command usage, input instructions, sample input, and sample output. The input instructions describe how the commands to the MMSP program are formatted, and the effects of using various command options.

Chapter 4 presents some sample applications of the MMSP program for analysis of ground-water simulations.

Chapter 5 gives an overview of the internal workings of the MMSP program and describes corrective actions for errors that may occur when using the program.

CHAPTER 2
PROGRAM DESIGN

Hardware and Software Requirements

The MMSP program is written in Fortran 77 (American National Standards Institute, 1978). The program was written and tested on a Prime 1/ model 9955-II mini-computer using revision 21 of the Primos operating system. However, the program should compile with minor modifications on any computer system running standard Fortran 77. The only non-standard language element in the program code is the Prime $INSERT compiler directive (Johnson, 1983).

The $INSERT directive causes the compiler to include the contents of a named file at the location of the directive. This allows repetition of a statement or a "block" of statements in several locations in the program. The $INSERT directive is used in the MMSP code for repeating declaration statements for variables at several places in the program. Other compiler vendors provide a similar compiler directive for Fortran 77. Often the corresponding compiler directive is called INCLUDE. The user of a compiler that does not provide this capability can insert the appropriate statements into the program at the locations of the applicable $INSERT directive.

Six blocks of program code are inserted into the MMSP main program and subroutines. These six blocks of code are presented in attachment A. The locations where these blocks of code are inserted are shown in table 1.

The first inserted block defines the "Z" array that contains most of the data used by the MMSP program. Further explanation of the "Z" array can be found in Chapter 2, Space Allocation.

1/ The use of firm and trade names in this report is for identification purposes only and does not constitute endorsement by the U.S. Geological Survey.

Table 1.--*Routines containing inserted blocks of programming statments*

[An "X" in the table indicates the routine contains the corresponding block of inserted programming statements. The programming statements in each block are presented in Attachment A.]

Name of routine	Name of Inserted Block of Programming Statments					
	ZARRAY.COMMON.INS	STKSIZE.INS	STKDEF.INS	TINY.INS	FLWCOM.INS	MISVAL.INS
MMSP (main)	X	X	X			
BCFDAL					X	
BCFDRP					X	
DFT1CM		X	X			
CTL1AL		X	X			
CTL1CM		X	X	X		
STA1SL					X	
SSTA1E				X		
HIS1EX				X		
COM1EX				X		
HEA1EX						X
REA1EX	X	X	X			
REB1EX				X		
SSLI1Y				X		
SSLI1Q				X		
VEC1EX	X			X		
MAS1EX				X		
MAS1MV				X	X	X
ULC1DS	X	X	X			
USYDUD					X	
USKDUD		X	X			
ULC1LY					X	
UBUBLE		X				
U1DREL				X		
U2DREL				X		

The next two inserted blocks of code pertain to allocation of space for temporary data storage in a "stack". Further explanation of the "stack" is found in Chapter 2, Auxiliary Data Arrays.

The fourth inserted block of code defines small and large single and double precision numbers. Alteration of these values may be necessary for MMSP program use on a computer system with a different precision or floating-point number representation scheme than that in Prime Fortran 77.

The fifth inserted block of code defines the size of the array LAYCON, which limits the maximum number of layers that can be simulated using the Modular Model. If the size of the LAYCON array has been increased in the Modular Model program, the LAYCON array size should be increased in the MMSP program.

The sixth inserted block of code defines three missing value indicators. The missing value indicators are unique numbers that are unlikely to occur in data read or computed by the MMSP program. When the MMSP program prints and writes data arrays, some data can be replaced by the missing value indicators. The presence of the missing value indicators in the data output by the program indicates that these data are not available or have been excluded using the methods described in a subsequent section, Subsetting Data Arrays.

The MMSP program uses some subroutines contained in the proprietary IMSL subroutine library. A listing of these subroutines is shown in table 2. Two different versions of the IMSL library can be used with the program. These subroutines are described in the IMSL library documentation (IMSL Inc., 1982 and 1987).

Table 2.--*IMSL subroutines used by the Modular Model Statistical Processor*

[Release of IMSL library Edition 9 preceded the release of IMSL library Version 1.]

IMSL Edition 9 subroutine name	IMSL Version 1 subroutine name	MMSP calling subroutine	Purpose of IMSL subroutine
BDCOU1	OWFRQ	HIS1EX	Frequency computation
USHST	VHSTP	HIS1EX	Histogram printing
VSRTAD	DSVRGN	HIS1EX	Sorting
VSRTRD	DSVRBP	STA1EX	Sorting with permutations
UGETIO	UMACH	HIS1EX	File unit specification

The MMSP program uses some of the utility subroutines from the Modular Model. These subroutines are: (1) U2DREL -- the two-dimensional real array reader, (2) U2DINT -- the two-dimensional integer array reader, (3) U1DREL -- the one-dimensional real array reader, (4) ULAPRS -- the strip-format array printer, (5) ULAPRW -- the wrap-format array printer, and (6) UCOLNO -- the column heading printer. Documentation of these subroutines is contained in Chapter 14 of the Modular Model documentation. All these subroutines except UCOLNO have been modified to provide some additional printing options that are not provided in the Modular Model versions of these subroutines. The modified versions are presented in attachment B of this report. The other subroutine is identical to the one used by the Modular Model and is not reprinted in this report.

Structure of the Program

The organization of the program is similar to the Modular Model and consists of a small main program with small modular subprograms that perform specific tasks. Where appropriate, the names of variables and subroutines are the same as or similar to corresponding names in the Modular Model.

The main program calls subroutines to allocate memory and read initial data describing the ground-water simulation. The main program iteratively invokes a command processing subroutine that reads and executes commands from a user-supplied command file. Groups of subroutines are associated with each of the commands that the MMSP program can execute. In addition, a group of utility subroutines are used for the processing of commands.

Space Allocation

Nearly all spatial and time-variant data are stored in memory by the program using a single one-dimensional array called the "Z" array. With this memory management scheme, the user is not obligated to size many different arrays for every simulation. The Modular Model uses a similar method for storing data in a one-dimensional array called the "X" array. However, different data are stored in the MMSP "Z" array than are stored in the Modular Model "X" array. Therefore, the size of the "Z" array for the MMSP program may be different from the size of the "X" array for the Modular Model. The algorithm shown in table 3 provides a method for estimating the approximate size of the "Z" array. If a programmer modifies the MMSP program, the algorithm shown in table 3 may not be accurate.

In the Modular Model, the declaration for the "X" array occurs only in the main program. When adjusting the Modular Model to accomodate different simulations, it is only necessary to edit, recompile, and link the main program to the various packages. Because the declaration for the "Z" array occurs at several places in the MMSP program, resizing of the MMSP program in the same way is not possible. The size of the "Z" array is assigned by the value of the variable LENZ in the block of statements named ZARRAY.COMMON.INS. When using a compiler that can insert these statements during compilation, the MMSP program can be resized by changing this block of statements, recompiling, Table 3.--*An algorithm for estimating of the size of the "Z" array needed for a given ground-water simulation*

[Sum the results of the calculation on each applicable line to compute approximate size required for the simulation and modify the value of LENZ in the inserted block of programming statements ZARRAY.COMMON.INS]

All Simulations:	
	9 * Number of Columns * Number of Rows
	15 * Number of Columns * Number of Rows * Number of Layers
If Modular Model Recharge Package is used:	
	2 * Number of Columns * Number of Rows
If Modular Model Statistical Processor Vector command is used:	
	4 * Number of Columns * Number of Rows

and relinking the program. When using a compiler that cannot insert these statements, the value of LENZ must be changed in all four routines listed in table 1.

After all the memory requirements for time- and space-dependent data have been computed by the MMSP program, a summary of memory utilization of the "Z" array is printed. If the "Z" array contains insufficient storage for a given simulation, an error message is printed and the programs stops. Otherwise, a message is printed showing the size of the "Z" array necessary for processing the simulation.

The value of LENZ should be increased if there is insufficient storage space for processing a simulation. Similarly, the value of LENZ can be decreased if the MMSP program is allocating too much storage space. Allocation of unnecessary storage space may cause a computer system to perform poorly, or may increase the costs charged by a computer center for running the program.

One of the capabilities of the MMSP program is the calculation of flow vectors. More memory space is needed by the program when these calculations are performed. As all users of the MMSP program may not need to calculate flow vectors, the program will not stop if sufficient space is not available for flow vector calculation. The program prints a warning message if there is insufficient space for flow vector calculation. If the flow vectors are subsequently requested by the user, a message will be printed indicating that there is insufficient space to process the request, and the flow vectors are not computed.

Command File

The MMSP program reads all the data that are used for running the Modular Model. The MMSP program reads data for the basic package from Fortran unit 5 and writes output to Fortran unit 6. The data written to Fortran unit 6 describes the results of the operations performed by the MMSP program. The MMSP program reads the data for the Modular Model specified in the basic package, and subsequently reads data for other packages which are identified by positive Fortran unit numbers in the IUNIT array (McDonald and Harbaugh, 1988, p. 3-27). Therefore, the job control directives that are used to run the Modular Model are used, with few modifications, to run the MMSP program. Additionally, the MMSP program needs a command file to be open on Fortran unit 7.

The user controls the operation of the MMSP program by supplying a command file. The command file must contain at least four records which comprise two title lines to be displayed in the output from the MMSP program. The title lines may each contain up to 128 characters, of which no more than 80 characters may be entered on the first record of the respective title line in the command file. This is the same format used to enter the title line (HEADNG) for the basic package of the Modular Model. Thus, the first four records in the command file are used for entering two title lines that are printed in the output of the program. The MMSP program prints these two title lines after the title read from the basic package of the Modular Model.

The MMSP program reads and uses the data for the basic, output-control, block-centered flow, well, and recharge packages. Data for other packages are read by the program although these data are not used.

When the MMSP program reads the data that were supplied to the various packages of the Modular Model, the program uses subroutines that are similar to the subroutines that read these data for the Modular Model. Therefore, by default the program will print these data in the same manner as the Modular Model prints the data. The Modular Model uses the value specified for the IPRN variable on the array-control record to determine whether to print these data (McDonald and Harbaugh, 1988, p. 14-5).

Some modelers may want to suppress the printing of all data arrays that are input to the Modular Model when using the MMSP program. The printing can be suppressed by placing a non-blank character in column 80 of the fourth record in the command file. A sample command file is shown below with printing of model-input data suppressed.

```
MMSP TITLE LINE #1, PART 1, 80 CHARACTERS ......................................
...LINE #1, PART 2, 48 CHARACTERS
MMSP TITLE LINE #2, PART 1, 80 CHARACTERS ......................................
...LINE #2, PART 2, 48 CHARACTERS                                              X
```

If no records other than the title lines are supplied in the command file, the MMSP program will perform a set of default commands. The default commands will cause the MMSP program to attempt to perform the following operations. (Some of these operations may not be performed successfully because the data necessary have not been supplied to the Modular Model.)

(1) Compute descriptive statistics on well pumpage rate at active model nodes.
(2) Compute descriptive statistics on recharge flux at all model nodes.
(3) Compute a frequency analysis of starting heads at active model nodes.
(4) Check for well pumpage at inactive or constant-head model nodes.
(5) Check for recharge at inactive or constant-head model nodes.
(6) Check for primary storage coefficient greater than 0.005.
(7) Check for starting heads below the elevation of the aquifer bottom.

The user can override these default operations by specifying one or more commands for the MMSP program that describe the operations to be performed. Commands are appended to the end of the command file, after the fourth title record. The default commands are not performed if more than four records are included in the command file.

Commands consist of column-dependent records, of which the first four columns are always the command abbreviation. Acceptable command abbreviations are shown in table 4. Commands abbreviations and any other character data entered on the command record may be entered in upper- or lower-case letters. A blank column always follows the command abbreviation (column 5). Detailed descriptions of the commands are found in chapter 3.

Table 4.--*Commands accepted by the Modular Model Statistical Processor*

Command abbreviation	Command explanation
****	Comment, not processed.
TITL	Title, defines an optional title line
PRIN	Printing, writes array to print file
WRIT	Write, writes array to disk file
STAT	Statistics, computes descriptive statistics
HIST	Histogram, performs frequency analysis
COMP	Comparison, compares arrays
MATH	Mathematics, computes an array
READ	Read, reads an array
HEAD	Head, writes computed heads for selected nodes
REBO	Reset, updates the model-boundary array
SLIC	Slice, subsets simulation grid with a plane
THIC	Thickness, defines thickness of layers used during computation of flow-vector locations
VECT	Vector, computes flow vectors

Modular Model Data Arrays

The MMSP program reads the same data files that are used by the Modular Model. No reformatting or modification of the Modular Model data files is necessary to produce statistical analyses using the MMSP program. Optionally, the user may use the MMSP program to read data computed by the Modular Model. Again, the MMSP program can read these data without reformatting or modification.

Selected data arrays used by the Modular Model are always read by the MMSP program. These data arrays are shown in table 5. Other data from the Modular Model are read by the MMSP program and are used for calculations and error detection. These data are shown in table 6. The reading of the data shown in tables 5 and 6 is controlled by the IUNIT array in the same manner as the Modular Model. Because the data listed in tables 5 and 6 are automatically read by the MMSP program, the files containing these data must be opened by job control directives prior to running the program. All of the data shown in tables 5 and 6 are in reserved storage space within the MMSP program. Thus, these data are always available for computation and use while the MMSP program is running. This is not so for auxiliary data sets read or computed by the MMSP program, as will be explained in the next section, Auxiliary Data Arrays.

Table 5.--*Modular Model data arrays read and available to a user of the Modular Model Statistical Processor*

Array description	Package	Variable name
Starting head	BAS	STRT
Storage coefficient	BCF	SC1
Layer bottom	BCF	BOT
Layer top	BCF	TOP
Recharge rate	RCH	RECH

Table 6.--*Modular Model data read and used internally by the Modular Model Statistical Processor*

[The following abbreviations are used: BAS is basic, OC is output control, BCF is block-centered flow, RIV is river, RCH is recharge, WEL is well, DRN is drain, EVP is evapotranspiration, GHB is general-head boundary, and UTL is utility]

Data description	Package	Variable name
Simulation title	BAS	HEADNG
Number of layers	BAS	NLAY
Number of rows	BAS	NROW
Number of columns	BAS	NCOL
Number of stress periods	BAS	NPER
Input-unit array	BAS	IUNIT
Model-boundary array	BAS	IBOUND
Head for inactive cells	BAS	HNOFLO
Number of time steps	BAS	NSTP
Computed-head unit number	OC	IHEDUN
Computed-drawdown unit number	OC	IDDNUN
Head/drawdown output code	OC	INCODE
Head/drawdown save flags	OC	IOFLG
Cell-by-cell unit number	BCF	IBCFCB
Layer-type table	BCF	LAYCON
Cell width along rows	BCF	DELR
Cell width along columns	BCF	DELC
Cell-by-cell unit number	RIV	IRIVCB
Recharge-option code	RCH	NRCHOP
Cell-by-cell unit number	RCH	IRCHCB
Recharge-read flag	RCH	INRECH
Recharge-layer-read flag	RCH	INIRCH
Recharge layer	RCH	IRCH
Maximum number of wells	WEL	MXWELL
Cell-by-cell unit number	WEL	IWELCB
Well flag/counter	WEL	ITMP
Well layer	WEL	K
Well row	WEL	I
Well column	WEL	J
Well recharge/discharge rate	WEL	Q
Cell-by-cell unit number	DRN	IDRNCB
Cell-by-cell unit number	EVT	IEVTCB
Cell-by-cell unit number	GHB	IGHBCB
Flag and unit number	UTL	LOCAT
Constant	UTL	CNSTNT
Format	UTL	FMTIN
Flag and print-format code	UTL	IPRN

Three computed data arrays are available to the user of the MMSP program for analysis. These data arrays are recharge flux, well pumpage rate, and cell area. Recharge flux for each node is computed by multiplying the recharge rate, by the width of the cell along the column, and by the width of the cell along the row. The MMSP program uses the recharge option (NRCHOP) to determine which layer receives recharge. However, the model boundary array is not used to exclude recharge from inactive or constant head nodes. Because the thickness of cells may be variable across the areal extent of the model grid, recharge flux is computed as a unit depth. Well pumpage rate data are prepared by setting the well discharge/recharge rate, for the node at the the layer, column, and row location specified for the well. The well pumpage rate is set to zero at all other nodes. Cell area is computed from the width of cells along columns and the width of cells along rows. Because all of the variables needed to prepare well pumpage rate, recharge rate, recharge flux, and cell area are stored in reserved locations, these three computed data arrays are always available to the user of the MMSP program.

The data for recharge rate and well pumpage rate may be altered for each stress period during Modular Model runs. If the well or recharge packages are used in the Modular Model, the MMSP program automatically reads the data input to these packages for the first stress period when the program begins. The user may read pumpage or recharge data for subsequent stress periods using the READ command described in chapter 3.

Auxiliary Data Arrays

Data arrays that are not used by the Modular Model also may be analyzed using the MMSP program. These data arrays must be formatted in a manner compatible with the MMSP program. Some analyses desired by users of the MMSP program may require data that are not entered for the Modular Model. An example of such an analysis is to test if starting heads or computed heads are higher than the land surface at any model nodes. To perform this calculation, the program must be provided with an array of values representing the altitude of the land surface because these data are not supplied to the Modular Model. These data must be read by the MMSP program before performing the test. Then the land-surface array may be compared on a node-by-node basis with the head array.

Another example is the calculation of descriptive statistics for the change in head between time steps. Using the MMSP program, the computed heads may be read for two different time steps, and an array may be calculated on a node-by-node basis that is the difference between the two computed head arrays. After the difference array has been calculated, statistics may be computed on that array.

Whenever auxiliary data arrays are read or computed by the MMSP program, whether these data are output generated by the Modular Model or other data, the data are stored in a temporary storage area referred to as the "stack". The stack is a collection of temporary storage locations for data arrays. The MMSP program maintains two stacks: One stack for the storage of two-dimensional (column and row) data, and the other stack for the storage of three-dimensional (column, row, and layer) data. The logic of the MMSP program determines which stack should contain a given data array, based upon the size of the data array. Whenever an operation is performed by the MMSP program that alters the contents or positions of arrays stored in a stack, the program prints a summary of the stack contents.

The two-dimensional and the three-dimensional stacks each contain space to store four arrays. When a new array is stored in a stack, the new entry is placed at the bottom (fig. 1). The previous entries on the stack are moved upward in storage location. Storage of a fifth entry will cause removal of the oldest entry, which will no longer be available for analysis using the MMSP software.

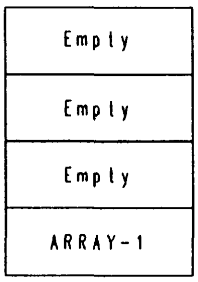
Data-set Names

When a command is given to the MMSP software, the user indicates which data array(s) is to be the object of the command by specifying a data-set name (DSN) for the data array. Data-set names consist of 1-6 characters in either upper- or lower-case letters. Lower-case letters are converted to upper case prior to being processed by the MMSP program. When a data-set name is given that consists only of numeric data, the MMSP program prepares a data array with all nodes equal to the numeric value that was entered for the data-set name.

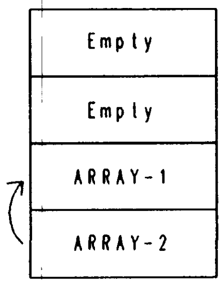
The data arrays input to the Modular Model, as shown in table 5, are assigned data-set names and are available for analysis using the MMSP program. The data arrays for well pumpage rate, recharge rate, recharge flux, and cell area, and the data arrays output by the Modular Model have reserved data-set names. There are also two other reserved data-set names (IBOUND and CLASS), and the usage of these names is explained in chapter 3. The reserved data-set names are shown in table 7.

When the MMSP program requires a data array to perform an operation, the following steps are used to match a data-set name with a data array. The program first determines if the data array must be generated because it is well pumpage rate, recharge flux, or cell area. If the data array is not to be generated, the program determines if the data array is one of the Modular Model data arrays listed in table 5. If the data array is still not defined, the program determines if the two-dimensional stack contains the data-set name. The stack is searched from the bottom position to the top position. Failing to find the data-set name, the program will examine the three-dimensional stack in a similar fashion. Finally, the program will attempt to read the data-set name as a real number. If the data-set name is a real number, then a three-dimensional data array is created with all values in the array equal to the real number.

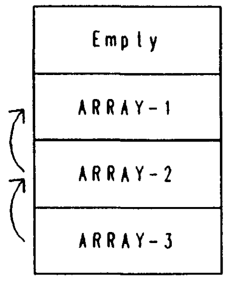
If the data-set name is found during any of the steps listed above, further searching for the data array ceases, and the data array whose DSN matched the requested DSN is used. There is one exception to this rule. Two commands available in the MMSP program can use two data arrays. These two commands are (1) the computation of a data array (MATH), and (2) the



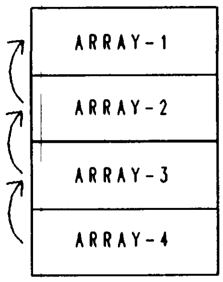
(1) Put ARRAY-1 on stack



(2) Put ARRAY-2 on stack



(3) Put ARRAY-3 on stack



(4) Put ARRAY-4 on stack

Figure 1.--Movement of data arrays on the stack as new data arrays are added.

Table 7.--*Reserved data-set names for the Modular Model Statistical Processor*

Data-set name	Contents of the data array
STRT	Starting head
SC1	Storage coefficient
TOP	Top of layer
BOT	Bottom of layer
WELL	Well-pumpage rate
RECH	Recharge rate
RECHF	Recharge flux
AREA	Area of cells
HEAD	Computed heads
DRAWDN	Computed drawdowns
STORAG	Storage cell-by-cell flow terms from BCF package
CNHEAD	Constant head cell-by-cell flow terms from BCF package
RIFACE	Right-face cell-by-cell flow terms from BCF package
FRFACE	Front-face cell-by-cell flow terms from BCF package
LOFACE	Lower-face cell-by-cell flow terms from BCF package
CBCRIV	Cell-by-cell flow terms from RIV package
CBCRCH	Cell-by-cell flow terms from RCH package
CBCWEL	Cell-by-cell flow terms from WEL package
CBCDRN	Cell-by-cell flow terms from DRN package
CBCEVT	Cell-by-cell flow terms from EVT package
CBCGHB	Cell-by-cell flow terms from GHB package
UBOUND	User boundary
CLASS	Histogram classes

comparison of one data array with another data array (COMP). It is possible to duplicate data-set names on the two- or three-dimensional stacks. When either MATH and COMP operations are performed, and the user requests the same DSN for both input data arrays and the data arrays are stored on the same stack, the following occurs. The first use of the DSN results in the MMSP program using the first occurence of the DSN encountered on the stack; the second use of the DSN in the same operation results in the MMSP program using the second occurence of the DSN on the stack. If the DSN is stored on the stacks in only one location, then the same data array is used for both occurences of the DSN.

The MMSP commands READ and MATH provide the capabilities for reading or computing a data array and giving a data-set name to the corresponding data array. The previously described searching procedure for data-set names causes some limitations in names that can be assigned to new data arrays either read or created using the READ and MATH commands.

Any data-set name that abides by the following rules may be given to a data array. First, the name must not conflict with any of the data-set names that have reserved storage locations within the MMSP program (STRT, SC1, BOT, TOP, and RECH). Second, the name must not conflict with either of the data-set names that identify a data array which will be computed by the MMSP program (WELL, RECHF, and AREA). Third, no new three-dimensional data array should be given the same name as a data array that is already stored on the two-dimensional stack. Until the two-dimensional data array is removed from the stack, the like-named three-dimensional data array will never be used by the MMSP program.

It is permissible, but not advisable, to name a data array with a number (such as "1"). As long as a data array with a numeric name is on the stacks, the MMSP program will use the data array from the stack, and the MMSP program will not create an array with all nodes equal to that numeric value. Alternatively, a data-set name with a number in characters (such as "ONE") causes no problems.

Subsetting Data Arrays

Subsets of data arrays may be created in a number of different ways for analysis by the MMSP program. Often the user may wish to restrict an analysis to a single layer of a three-dimensional data array. The six commands PRIN, WRIT, STAT, HIST, MATH, and COMP allow the user to specify a layer number, which confines the operation to the selected layer. When the layer number is blank, or zero, all layers of the data array are used for the analysis.

If a layer number is specified for a two-dimensional data array, an error message is written to the output file, the specification is ignored, and the operation is attempted using the entire two-dimensional data array. Certain three-dimensional data arrays may not always contain data for every layer in the simulation, therefore these data arrays can be "sparsely layered". The data arrays for layer top (TOP) and layer bottom (BOT) contain data only for layers appropriately identified by the layer index variable (LAYCON, specified in the input to the block-centered flow package of the Modular Model). Likewise, the calculated heads (HEAD) and drawdowns (DRAWDN) for a time step may be sparsely layered by the specifications for the head/drawdown save flag variable (IOFLG), that is specified for the output-control package of the Modular Model. The MMSP program uses the LAYCON and IOFLG variables and will print a message if the user attempts to use a layer that is not available for a sparsely layered data array.

A subset of a data array also may be created by masking some values during an analysis. This subsetting process is called masking because data values are either removed or replaced prior to performing an operation. The operation is performed using data values that remain after application of the mask.

When using a mask with the STAT, HIST, and COMP commands, values that have been masked are removed from the data array prior to performing the command. Thus, with these three commands, application of a mask causes the number of data values to be reduced.

The commands PRIN, WRIT, and HEAD can mask data arrays also. When values are masked during these three commands, the masked data values are replaced by a missing-value indicator. The missing-value indicator is a number indicating that the value has been masked. The default missing-value indicators are shown in attachment A, in the inserted block of statements named MISVAL.INS. Each of the three missing-value indicators is associated with one of three masks that are discussed in the following paragraphs. When a data array is masked during the commands PRIN, WRIT and HEAD, the number of values remaining in the data array is not reduced.

There are three masking procedures available that may be specified individually or in combination. The three masking procedures are named (1) the zero mask, (2) the model-boundary mask, and (3) the user-boundary mask. The simplest mask is the zero mask. When the zero mask is specified, values equal to zero in the data array are masked before performing the requested operation. The other two masks are boundary masks. When applying a boundary mask, a second array is used to mask the contents of the data array that is input to the operation. The value of each element in the masking array, called the boundary array, is used to determine whether to mask or not to mask the corresponding element of the data array.

The first of the boundary masks uses the model-boundary array (IBOUND), that is input to the the basic package of the Modular Model. The IBOUND array identifies model nodes as being constant head, inactive, or active by negative, zero, or positive values, respectively. An operation may be restricted to analysis of any combination of these three groups of model nodes. To identify which group of model nodes are to be included in the analysis using the model-boundary mask, the user specifies a "mask key" value. These values range from -3 to +3 (table 8). For example, to restrict an analysis to active model nodes, and mask the values of inactive and constant head nodes, the mask-key value should be set equal to one.

Table 8.--*Explanation of the model-boundary mask-key field of the Modular Model Statistical Processor*

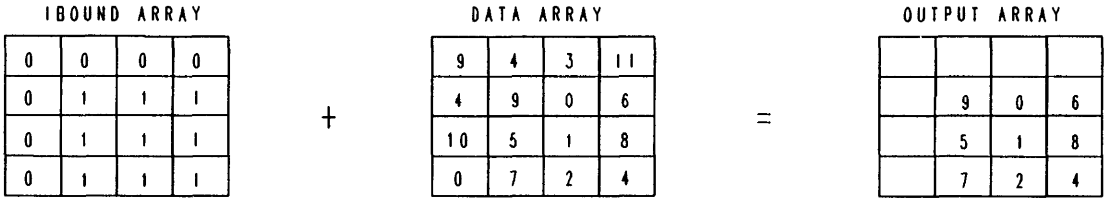
Model nodes remaining after mask application	Mask key
Inactive nodes	-3
Constant head nodes	-2
Inactive and constant head nodes	-1
Inactive, constant head and active nodes	0
Active nodes	1
Constant head and active nodes	2
Inactive and active nodes	3

The second of the boundary masks uses a user-specified boundary array (UBOUND), that may be defined for the MMSP program using either of two methods. To define the UBOUND array by the first method, enter the user-boundary array using the READ command of the MMSP program. The second method for creating a user-boundary array slices the model grid with a plane that defines the user boundary. Using this second method, a user boundary may be created along any row, column, or layer; or, optionally, may slice the model grid obliquely. Details for creating a user-boundary array can be found in chapter 3.

The user-specified boundary array consists of an integer value for each node in the model grid. UBOUND array values greater than zero indicate that the corresponding nodes of the model are "inside" the boundary. UBOUND array values less than or equal to zero indicate that the corresponding nodes of the model are "outside" the boundary. The UBOUND mask may be used to mask data values that are either "inside" or "outside" the user's boundary. A positive value for the UBOUND mask key will cause masking of all values in a data array whose corresponding elements of the UBOUND array that are "outside" the boundary, and all values that are "inside" the boundary are not masked. A negative value for the UBOUND mask key will cause masking of all values in a data array whose corresponding values of the boundary array are inside the boundary.

Application of the three masks by the MMSP program is cumulative. When combining the UBOUND mask with the IBOUND mask and the zero mask, as many as 42 different combinations of composite masks are possible. If a value of a data array is masked by any of the three masking options, the value will be masked for the duration of the operation. Three examples of masking a data array are shown in figure 2.

When a two-dimensional data array is masked using one of the two boundary masks, it is necessary to identify how the mask should be applied. Since the IBOUND and UBOUND boundary arrays are three-dimensional, it is important to indicate which layer from the boundary array should be used for masking the two-dimensional data array. The layer number is used to identify which layer from the boundary array should be used for masking the data array.



IBOUND ARRAY

0	0	0	0
0	1	1	1
0	1	1	1
0	1	1	1

+

DATA ARRAY

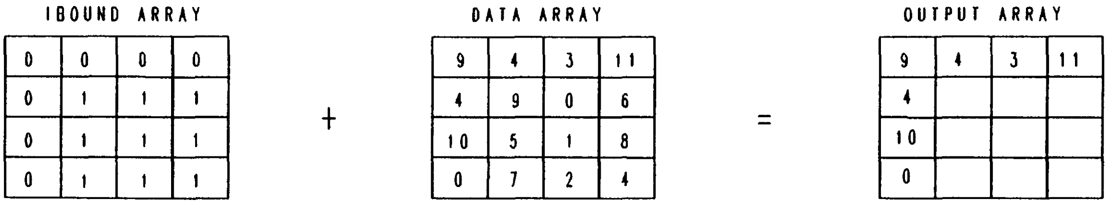
9	4	3	11
4	9	0	6
10	5	1	8
0	7	2	4

=

OUTPUT ARRAY

	9	0	6
	5	1	8
	7	2	4

EXAMPLE ONE: Masking of inactive nodes



IBOUND ARRAY

0	0	0	0
0	1	1	1
0	1	1	1
0	1	1	1

+

DATA ARRAY

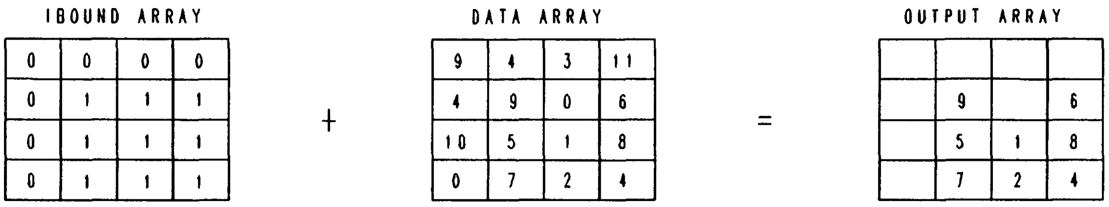
9	4	3	11
4	9	0	6
10	5	1	8
0	7	2	4

=

OUTPUT ARRAY

9	4	3	11
4			
10			
0			

EXAMPLE TWO: Masking of active nodes



IBOUND ARRAY

0	0	0	0
0	1	1	1
0	1	1	1
0	1	1	1

+

DATA ARRAY

9	4	3	11
4	9	0	6
10	5	1	8
0	7	2	4

=

OUTPUT ARRAY

	9		6
	5	1	8
	7	2	4

EXAMPLE THREE: Masking of inactive and zero-value nodes

Figure 2.--Examples of the application of masking to a data array.

CHAPTER 3.
USE OF THE PROGRAM

General Comments

This chapter provides a description of each of the commands that may be used in the MMSP command file for controlling the operations performed by the program. The chapter has a section for each command and each section contains a discussion of the command, input instructions, example input, and example output. The purpose of this chapter is to provide the user with a knowledge of the capabilities and the requirements of the MMSP commands. This chapter can be used to learn about the capabilities of the program and as a reference guide for information about specific commands. Therefore, some information that is applicable to more than one command is repeated in several sections.

The examples in this chapter show how commands are used and formatted. The examples are relatively simple applications of the MMSP program. More complex applications of the MMSP program are presented in chapter 4.

The example input and output shown in this chapter consistently refer to the same ground-water simulation. The simulation presented is based on the sample problem in Appendix D of the Modular Model documentation (McDonald and Harbaugh, 1988, p. D-1 through D-12). The presentation of the simulation in Appendix D has been modified to save unformatted cell-by-cell flow terms and computed head data. The output-control package data for the simulation has been provided by specifying a non-zero Fortran unit in the IUNIT array (in the basic package, McDonald and Harbaugh, 1988, p. 4-11). The contents of the output-control package data (McDonald and Harbaugh, 1988, p. 4-14 through 4-16) are shown below.

```
0      0      29      0     IHEDFM,IDDNFM,IHEDUN,IDDNUN
0      1       0      1     INCODE,IHDDFL,IBUDFL,ICBCFL
0      0       1      0     HDPR,  DDPR,  HDSV,  DDSV
```

Additionally, the cell-by-cell flow terms have been saved to a file by specifying positive values for the cell-by-cell flow-term Fortran unit number in the input data sets for the block-centered flow, recharge, drain, and well packages. These values are specified for the variables: IBCFCB, IRCHCB, IDRNCB, and IWELCB, respectively.

The MMSP program processes commands sequentially in the order that the commands are placed in the command file. Therefore, if data produced by a command are required for processing of a second command, the second command is placed after the first command in the command file.

Command Descriptions

Documenting the Command File

DISCUSSION

When placing commands in the MMSP command file, inserted comments will be ignored by the MMSP program. These comments can be used to document commands or groups of commands in the command file. Two methods may be used for placing comments in the command file. The first method is to use the MMSP comment command that is described in this section. The second method is to place comments in unused columns at the end of other MMSP commands.

INPUT INSTRUCTIONS

Field:	Command	COMMENT
Beginning column:	1	6
Data Format:	****	A75

MMSP program comments are designated by placing asterisks in the first four columns of the command line.

SAMPLE INPUT

Input Command File

```
****
**** ------------------- STAT COMMAND FORMAT --------------------
****               L  O I U       (O = ZERO-MASK)
****               A              (I = MODEL BOUNDARY MASK)
****        DSN    Y  MASKS       (U = USER BOUNDARY MASK)
**** STAT ______  __  ______
```

SAMPLE OUTPUT

There is no printed output from the comment command.

Title Lines

DISCUSSION

There are three different types of title lines printed in the output of the MMSP program. The first type is a single line that is read from the input to the basic package of the Modular Model. The second type consists of two lines read from the beginning of the MMSP command file. All three of these title lines may contain 128 characters each, and are printed on top of pages throughout a single execution of the MMSP program.

The third type of title line is optional, and can be defined with the MMSP TITL command. This title line may contain 75 characters and can be changed as often as desired during an execution of the MMSP program. Titles entered with the TITL command are printed on the fourth title line below the title line from the basic package, and the titles from the beginning of the command file.

Most MMSP commands cause the program to print at least one page of output. The TITL command does not cause any new pages of output. Instead, the TITL command defines the fourth title line to be printed on the next page of output. The definition of the fourth title line remains in effect until another TITL command is read by the MMSP program from the command file.

INPUT INSTRUCTIONS

Field:	Command	TITLE
Beginning column:	1	6
Data Format:	TITL	A75

SAMPLE INPUT

Input Command File

```
MMSP TITLE LINE #2
MMSP TITLE LINE #3
TITL FORGOT TO HAVE THE MODULAR MODEL PRINT THE STARTING HEAD ARRAY
****     1    1    2    2    3    3    4    4    5    5    6    6    7
****5    0    5    0    5    0    5    0    5    0    5    0    5    0
PRIN STRT
```

SAMPLE OUTPUT

Output Print File

```
SAMPLE----3 LAYERS, 15 ROWS, 15 COLUMNS, STEADY STATE, CONSTANT HEADS COLUMN 1, LAYERS 1 AND 2, RECHARGE, WELLS AND DRAINS
MMSP TITLE LINE #2
MMSP TITLE LINE #3

FORGOT TO HAVE THE MODULAR MODEL PRINT THE STARTING HEAD ARRAY


Processing:  PRIN STRT



           PRINTING OF : STRT   - INITIAL HEADS
                                  ALL LAYER(S)
                                  USING FORMAT CODE: 000
```

COMMENTS

In this example, the TITL command was followed by a PRIN command. The PRIN command, discussed in a subsequent section, included in this example shows the effects of the TITL command. The complete results of the PRIN command are not shown here.

Reading Data Arrays

DISCUSSION

The MMSP program will read automatically the recharge and well pumpage-rate data for the first stress period if the Modular Model IUNIT array indicates that the packages are used in the simulation. The MMSP READ command can be used to read thse data for analysis of subsequent stress periods in a transient simulation.

A simulation using the Modular Model can write detailed flow and head information to binary output files. The data arrays in these binary files may be accessed and used by the MMSP program with the READ command.

Other data arrays to be analyzed with the MMSP program can be accessed, if these arrays are prepared in the format specified for the Modular Model array readers U2DREL and U2DINT. An example application of this capability is reading a data array representing the altitude of the land surface, in order to check if head exceeds the land-surface datum at any model nodes.

The READ command may be used to define the user-boundary array (UBOUND), for use in masking. The SLIC command, discussed later in this chapter, also may be used to define the user-boundary array.

A modeler may use the READ command to read a set of class boundaries for use with the HIST command when performing frequency analysis. These class boundaries are called "cut points". When performing a frequency analysis with user-specified cut points, the HIST command uses the cut points as class boundaries. Therefore, the number of frequency classes used by the HIST command will be one greater than the number of cut points provided by the modeler with the READ command.

Modelers can combine binary data of different types in the same file by specifying the same Fortran unit number in different places of the Modular Model input data. When the MMSP program reads cell-by-cell flow terms, computed heads, or computed drawdowns the program reads the record of identifying information in the binary file written by the Modular Model. If the information identifies the data specified by the READ command, the program reads the data, prints a summary, and processes the next command in the command file. However, if the identification record does not match the data requested, the program continues reading from the Fortran unit and a message is printed showing what data were found and indicating that the program is "fast-forwarding". The MMSP program continues in this manner until the requested data are found. If the end of the file is reached before the requested data are found the MMSP program returns to the beginning of the file and prints a message indicating that the program is "rewinding". The program then continues examining identification records, printing messages, and attempting to find the requested data. If the end of the file is reached a second time, the program prints an error message and the READ command aborts.

The costs and the processing time for running a computer program on many computer systems is related to the amount of time the computer uses for reading files. To minimize processing time, data arrays should be read in the same order as the arrays were written. A large number of "fast-forwarding" and "rewinding" messages in the output from MMSP may be an indication that the commands could be reordered for more efficient processing.

INPUT INSTRUCTIONS

Field:	Command	DSN	UNIT	NUMBER OF DIMENSIONS	DATA TYPE	STRESS PERIOD	TIME STEP	ARRAY-NAME
Beginning column:	1	6	13	16	18	20	23	26
Data Format:	READ	A6	I2	I1	A1	I2	I2	A24

When reading frequency analysis cut points:

Field:	Command	DSN	UNIT	NUMBER OF CUT POINTS
Beginning column:	1	6	13	16
Data Format:	READ	CLASS	I2	I2

To read a data array, include the READ command in the command file and specify the DSN of the data array to be read. If the data array is input for the well or recharge package of the Modular Model (WELL or RECH), the only other required information is the stress period desired. If the data array is one of the Modular Model outputs (HEAD, DRAWDN, CNHEAD, STORAG, RIFACE, FRFACE, LOFACE, CBCRIV, CBCRCH, CBCWEL, CBCDRN, CBCEVT, or CBCGHB), the only other needed information is the stress period and time step. For these data arrays, the program will determine the Fortran unit number, the number of dimensions, the data type, and the array name. This information is determined from the data input to the applicable packages of the Modular Model.

The MMSP program can only read computed heads, computed drawdowns, and cell-by-cell flow terms if these data were written by the Modular Model. The specifications for the various packages of the Modular Model determine if these data are written. To read HEAD or DRAWDN, the data for the Output Control package of the Modular Model should specify a unit number for IHEDUN or IDDNUN. Additionally, non-zero values must be specified for each of the time steps to be read for the variables IHDDFL, and HDSV or DDSV. To read cell-by-cell flow terms, data for each time step for the Output Control package should specify a non-zero value for the variable ICBCFL. Additionally, a positive integer should be supplied in the data input for each package identifying the unit number for recording the cell-by-cell flow terms (McDonald and Harbaugh, 1988, pp. 4-14 through 4-16).

Data arrays not mentioned in the previous paragraphs must be described in greater detail to retrieve them using the MMSP READ command. Any data array that is to be read using the READ command must be formatted with an array-control record as described in the Modular Model documentation, Utility Modules chapter. For these data arrays, the READ command must specify the following fields:

(1) The DSN to be given to the data array,
(2) The Fortran unit number that provides the array-control record (opened by job control directives before the beginning the MMSP program),
(3) The number of dimensions in the data array ("2" for row-column data, "3" for row-column-layer data),
(4) The type of data ("R" for real numbers, "I" for integers), and
(5) Optionally, a textual description of the array. The textual description is entered as a 24-character array name. This name is printed whenever the array is used, but the name is not needed for the MMSP program to operate.

Although the READ command allows the reading of integer data, the MMSP program will convert integer arrays to real numbers.

When reading a user-boundary array, the READ command should specify: a DSN of UBOUND, the Fortran unit number of the data file, "3" dimensions, and "I" type.

When reading a data array that was previously written using the WRIT command, use the same Fortran unit number in the READ command, as was used with the WRIT command. The WRIT command places the unit number in the array-control record that is written with the data array.

When reading frequency-analysis cut points, the READ command format differs from the format used for reading two- and three-dimensional data arrays. In this case, the READ command should specify: a DSN of CLASS, the Fortran unit number of the data, and the number of cut points to be read. The format of the data set containing the cut points should meet the specifications of the Modular Model one-dimensional real array reader: U1DREL.

SAMPLE INPUT

Input Command File

```
EXAMPLE OF READ COMMAND, USED TO READ DATA


****    1    1    2    2    3    3    4    4    5    5    6    6    7
****5   0    5    0    5    0    5    0    5    0    5    0    5    0
READ HEAD           1  1
```

SAMPLE OUTPUT

Output Print File

```
SAMPLE----3 LAYERS, 15 ROWS, 15 COLUMNS, STEADY STATE, CONSTANT HEADS COLUMN 1, LAYERS 1 AND 2, RECHARGE, WELLS AND DRAINS
EXAMPLE OF READ COMMAND, USED TO READ DATA



Processing:  READ HEAD          1  1



             READING : HEAD   - COMPUTED HEADS
                                ON UNIT:  29
                                STRESS PERIOD   1
                                TIME STEP       1

      THREE-DIMENSIONAL STACK CONTENTS AFTER READ COMMAND

          STACK   DATA SET STRESS TIME
         POSITION   NAME   PERIOD STEP DESCRIPTION

             4               0     0
             3               0     0
             2               0     0
             1     HEAD      1     1  COMPUTED HEADS
```

COMMENTS

This example shows the printed output from a READ command that is used for subsequent examples. The only information required to read the HEAD array is the stress period and the time step.

SAMPLE INPUT

Input Command File

```
EXAMPLE OF READ COMMAND, USED TO READ DATA


****     1    1    2    2    3    3    4    4    5    5    6    6    7
****5    0    5    0    5    0    5    0    5    0    5    0    5    0
READ CLASS  31 19
```

Input Disk File

```
      31         1. (8F10.4)                    1
0.000000   7.08039   14.1608   21.2412   28.3216   35.4019   42.4823   49.5627
 56.6431   63.7235   70.8039   77.8843   84.9647   92.0451   99.1255   106.206
 113.286   120.367   127.447
```

SAMPLE OUTPUT

Output Print File

```
SAMPLE----3 LAYERS, 15 ROWS, 15 COLUMNS, STEADY STATE, CONSTANT HEADS COLUMN 1, LAYERS 1 AND 2, RECHARGE, WELLS AND DRAINS
EXAMPLE OF READ COMMAND, USED TO READ DATA



Processing:  READ CLASS  31 19



                           HISTOGRAM CUT POINTS     WILL BE READ ON UNIT 31 USING FORMAT: (F10.4)
                           ------------------------------------------------------------------------------

 0.00000       7.0804       14.161       21.241       28.322       35.402       42.482       49.563       56.643       63.724
  70.804       77.884       84.965       92.045       99.126       106.21       113.29       120.37       127.45
```

COMMENTS

In this example, Fortran unit 31 had been opened with job control directives prior to beginning the MMSP program. The READ command was used to read histogram cut points for later use with the HIST command.

Printing Data Arrays

DISCUSSION

The MMSP program can print data arrays in the strip and wrap formats described in the Modular Model documentation (McDonald and Harbaugh, 1988, p. 14-2). An extra Modular Model simulation can be avoided if a modeler wishes to print one of the input or output data arrays. The READ command may be used to read binary-formatted data computed and written to a file by the Modular Model. These data may be printed using the MMSP PRIN command. Additionally, the MMSP MATH command (discussed in this chapter) can be used to compute a data array that can be printed with the PRIN command.

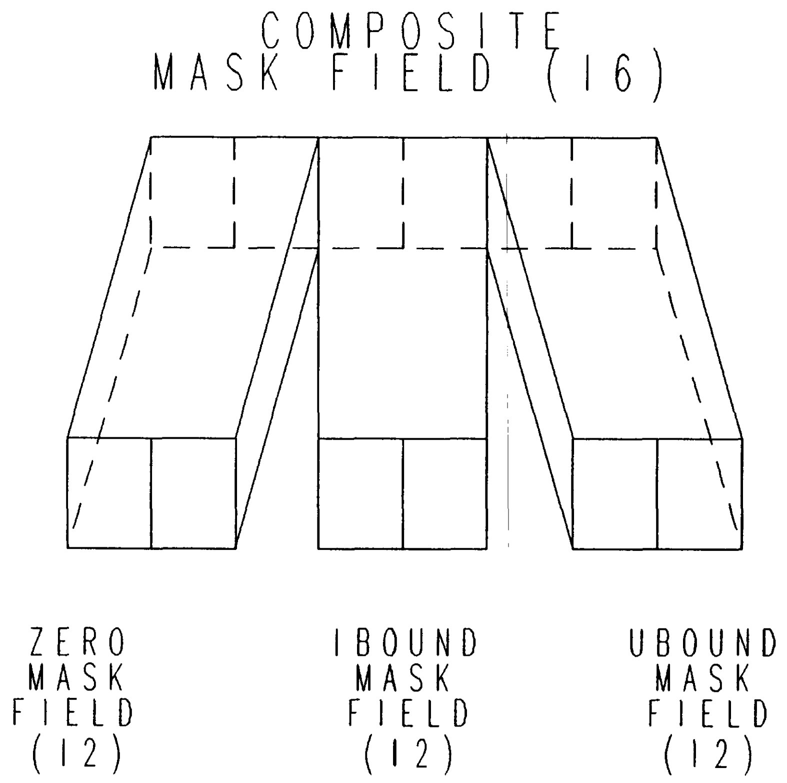
INPUT INSTRUCTIONS

Field:	Command	DSN	LAYER	MASK	FMT-CODE	MISSING-VALUES
Beginning column:	1	6	13	16	23	27
Data Format:	PRIN	A6	I2	I6	I3	3F10.0

To print a data array using the MMSP program, include the PRIN command in the command file and specify the DSN of the array to be printed. The operation of the PRIN command may be controlled by using any combination of the following options.

A specific layer of a three-dimensional data array may be printed by specifying the layer number with the PRIN command. If the layer number is not specified, the layer number defaults to zero, and causes all layers to be printed.

The three masks: zero, IBOUND, and UBOUND, discussed in chapter 2, Subsetting Data Arrays, may be used with the PRIN command. Individual masks are enabled or disabled by placing integer values in the appropriate mask fields (fig. 3). When a value is masked by the PRIN command, it is replaced by one of the three missing-value indicators. The three masks have a one-to-one correspondence with the three missing-value indicators. For example, when a zero-mask is requested by placing a non-zero value in columns 16-17, all zero values in the data array are replaced by the first missing-value indicator during printing. Values masked by the model-boundary array are replaced by the second missing-value indicator during printing. Similarly, values masked by the user-boundary array are replaced by the third missing-value indicator during printing.



NODES REMAINING AFTER EACH MASK APPLICATION

ZERO MASK FIELD

MASK KEY	NODES RETURNED
=0 :	All
>0 :	Non-zero
<0 :	Non-zero

MODEL BOUNDARY MASK FIELD

MASK KEY	NODES RETURNED
=-3 :	Inactive
=-2 :	Constant head
=-1 :	Inactive and Constant head
= 0 :	All
= 1 :	Active
= 2 :	Constant head and Active
= 3 :	Inactive and Active

USER BOUNDARY MASK FIELD

MASK KEY	NODES RETURNED
=0 :	All
>0 :	where UBOUND > 0
<0 :	where UBOUND <= 0

Figure 3.--Modular model statistical processor mask-field components.

By default, the missing-value indicators are defined by MISVAL.INS, as shown in attachment A. The default values for the missing-value indicators may be overridden for the duration of the command by specifying the desired numerical values in the MISSING-VALUES field of the PRIN command.

The data arrays for altitudes of the aquifer top (TOP) and aquifer bottom (BOT) do not necessarily contain data for all layers. The existence of data for a specific layer is determined by the value given in the layer-type array (LAYCON). All of these data are prepared as input to the Modular Model in the block-centered flow package (McDonald and Harbaugh, 1988, p. 5-37 through 5-40).

Similarly, the data arrays for computed head (HEAD) and computed drawdown (DRAWDN) do not necessarily contain data for all layers. The existence of data for a specific layer is determined by the value given in the save-flag array (IOFLG). IOFLG is entered into the Modular Model in the output-control package (McDonald and Harbaugh, 1988, p.4-16 and 4-59 through 4-62).

When a PRIN command specifies a particular layer, and that layer does not exist according to the specifications of LAYCON or IOFLG, the PRIN command issues an error message and stops performing the command. When a PRIN command specifies layer zero, indicating all layers are to be printed, any data layers that do not exist during printing are set equal to the first missing-value indicator.

The data array may be printed in either strip or wrap format using Fortran format specifications in the same manner as the Modular Model utility modules ULAPRS and ULAPRW (McDonald and Harbaugh, 1988, chapter 14). A Fortran format may be chosen from table 9. Format codes 1-12 presented in table 9 are the same formats provided for use with the Modular Model. The utility modules supplied with the MMSP program have the additional format codes 13-20 for use with 80-column output devices.

To select a Fortran format for use with the PRIN command, find the corresponding print-format code in the table, and enter the value in the FMT-CODE field of the PRIN command. A negative value of the print-format code will cause printing of the data array in the strip format. A positive value will cause printing of the data array in the wrap format. If the FMT-CODE field is blank or zero, the data are printed using the wrap format and print-format code 12, Fortran format (10G11.4).

Table 9.--*Print-format codes for WRIT and PRIN commands of the Modular Model Statistical Processor*

[With the 'PRIN' command: print-format code > 0 = wrap format
print-format code < 0 = strip format.
With the 'WRIT' command: print-format code > 0 = wrap format
print-format code < 0 = unformatted.
modified from McDonald and Harbaugh, 1988, p. 14-3.]

Format code	Fortran format
0	(10G11.4)
1	(11G10.3)
2	(9G13.6)
3	(15F 7.1)
4	(15F 7.2)
5	(15F 7.3)
6	(15F 7.4)
7	(20F 5.0)
8	(20F 5.1)
9	(20F 5.2)
10	(20F 5.3)
11	(20F 5.4)
12	(10G11.4)
13	(8G 9.0)
14	(8G 9.1)
15	(8G 9.2)
16	(8G 9.3)
17	(8F 9.0)
18	(8F 9.1)
19	(8F 9.2)
20	(8F 9.3)

SAMPLE INPUT

Input Command File

```
EXAMPLE OF PRINT COMMAND


**** PRINT WELLS IN TOP LAYER FOR STRESS PERIOD 1
**** MASK THE INACTIVE AND CONSTANT HEAD NODES TO 9
****     1    1    2    2    3    3    4    4    5    5    6    6    7
****5    0    5    0    5    0    5    0    5    0    5    0    5    0
PRIN WELL   01 000100 000                 9.0
```

SAMPLE OUTPUT

Output Print File

```
SAMPLE----3 LAYERS, 15 ROWS, 15 COLUMNS, STEADY STATE, CONSTANT HEADS COLUMN 1, LAYERS 1 AND 2, RECHARGE, WELLS AND DRAINS
EXAMPLE OF PRINT COMMAND



Processing:  PRIN WELL   01 000100 000                 9.0



             PRINTING OF :WELL   - WELL PUMPAGE
                                   LAYER    1
                                   STRESS PERIOD   1
                                   TIME STEP       1
                                   USING FORMAT CODE: 000

         Beginning mask from layer  1

         Masking was performed on WELL

      210 values unmasked out of       225

       15 points masked that were inactive or constant head nodes
```

Output Print File--Continued

```
        WELLS          IN LAYER  1 AT END OF TIME STEP  1 IN STRESS PERIOD  1
        ----------------------------------------------------------------------

              1          2          3          4          5          6          7          8          9         10
             11         12         13         14         15
 ............................................................................................................................

    1     9.000     0.0000     0.0000     0.0000     0.0000     0.0000     0.0000     0.0000     0.0000     0.0000
         0.0000     0.0000     0.0000     0.0000     0.0000

    2     9.000     0.0000     0.0000     0.0000     0.0000     0.0000     0.0000     0.0000     0.0000     0.0000
         0.0000     0.0000     0.0000     0.0000     0.0000

    3     9.000     0.0000     0.0000     0.0000     0.0000     0.0000     0.0000     0.0000     0.0000     0.0000
         0.0000     0.0000     0.0000     0.0000     0.0000

    4     9.000     0.0000     0.0000     0.0000     0.0000     0.0000     0.0000     0.0000     0.0000     0.0000
         0.0000     0.0000     0.0000     0.0000     0.0000

    5     9.000     0.0000     0.0000     0.0000     0.0000     0.0000     0.0000     0.0000     0.0000     0.0000
         0.0000     0.0000     0.0000     0.0000     0.0000

    6     9.000     0.0000     0.0000     0.0000     0.0000     0.0000     0.0000     0.0000     0.0000     0.0000
         0.0000     0.0000     0.0000     0.0000     0.0000

    7     9.000     0.0000     0.0000     0.0000     0.0000     0.0000     0.0000     0.0000     0.0000     0.0000
         0.0000     0.0000     0.0000     0.0000     0.0000

    8     9.000     0.0000     0.0000     0.0000     0.0000     0.0000     0.0000     0.0000     0.0000     0.0000
         0.0000     0.0000     0.0000     0.0000     0.0000

    9     9.000     0.0000     0.0000     0.0000     0.0000     0.0000     0.0000     -5.000     0.0000     -5.000
         0.0000     -5.000     0.0000     -5.000     0.0000

   10     9.000     0.0000     0.0000     0.0000     0.0000     0.0000     0.0000     0.0000     0.0000     0.0000
         0.0000     0.0000     0.0000     0.0000     0.0000

   11     9.000     0.0000     0.0000     0.0000     0.0000     0.0000     0.0000     -5.000     0.0000     -5.000
         0.0000     -5.000     0.0000     -5.000     0.0000

   12     9.000     0.0000     0.0000     0.0000     0.0000     0.0000     0.0000     0.0000     0.0000     0.0000
         0.0000     0.0000     0.0000     0.0000     0.0000

   13     9.000     0.0000     0.0000     0.0000     0.0000     0.0000     0.0000     -5.000     0.0000     -5.000
         0.0000     -5.000     0.0000     -5.000     0.0000

   14     9.000     0.0000     0.0000     0.0000     0.0000     0.0000     0.0000     0.0000     0.0000     0.0000
         0.0000     0.0000     0.0000     0.0000     0.0000

   15     9.000     0.0000     0.0000     0.0000     0.0000     0.0000     0.0000     0.0000     0.0000     0.0000
         0.0000     0.0000     0.0000     0.0000     0.0000
```

Writing Data Arrays

DISCUSSION

Data input to or output from the Modular Model can be used in a statistical package, a graphical display program, or similar programs. The MMSP program can write data arrays to a disk file using the WRIT command. Data arrays written by the MMSP program can be used as data input to the Modular Model.

When using the WRIT command, the user must provide a Fortran unit number that can be used for writing the data array. The Fortran unit must have been opened for writing by the user's job control directives before running the MMSP program.

The MMSP WRIT command can write data to a formatted or an unformatted file. When writing to a formatted file, data arrays are written with an array-control record that meets the specifications of the Modular Model real array reader U2DREL. The specifications for U2DREL are shown below (McDonald and Harbaugh, 1988, p. 14-4).

Data:	LOCAT	CNSTNT	FMTIN	IPRN
Format:	I10	F10.0	5A4	I10

LOCAT is a Fortran unit number. When the sign of LOCAT is negative, data are unformatted. The MMSP program will write the same Fortran unit number for LOCAT that is given with the WRIT command. CNSTNT is a multiplier. MMSP will set CNSTNT to zero, indicating that the data are not multiplied by a constant. FMTIN is a Fortran format. MMSP will place a Fortran format on the array-control record that corresponds to the format specified with the WRIT command. IPRN is a code that indicates whether the data array should be printed when it is read by U2DREL. MMSP always sets IPRN to negative one, which specifies that the data array should not be printed.

If unformatted data are written with the WRIT command, no array-control record is written with the data array. An example array-control record that could be used to read the unformatted data will be shown in the printed output from the WRIT command.

INPUT INSTRUCTIONS

Field:	Command	DSN	LAYER	MASK	UNIT	FMT-CODE	MISSING-VALUES
Beginning column:	1	6	13	16	23	26	30
Data Format:	WRIT	A6	I2	I6	I2	I3	3F10.0

To create a disk file containing a data array using the MMSP program, include the WRIT command in the command file, specify the DSN of the data array to be written, and the Fortran unit number to be used for writing the file. The operation of the WRIT command may be controlled by using any combination of the following options.

A specific layer of a three-dimensional data array may be written by specifying the layer number with the WRIT command. If the layer number is blank or zero, all layers are written.

The three masks: zero, IBOUND, and UBOUND, discussed in chapter 2, Subsetting Data Arrays, may be used with the WRIT command. Individual masks are enabled or disabled by placing integer values in the appropriate mask fields (fig. 3). When a value is masked by the WRIT command, it is replaced by one of the three missing-value indicators. The three masks have a one-to-one correspondence with the three missing-value indicators. For example, when a zero mask is requested by placing a non-zero value in columns 16-17, all zero values in the data array are replaced by the first missing--value indicator during printing. By default, the missing-value indicators are defined by MISVAL.INS, as shown in attachment A. The default values for the missing-value indicators may be overridden for the duration of the command by specifying the desired numerical values in the MISSING-VALUES field of the WRIT command.

The Fortran format of the data written to the file can be chosen from table 9. To select a Fortran format, the corresponding print-format code should be entered into the FMT-CODE field of the WRIT command. The default is to use print-format code zero, Fortran format (11G10.4). Care should be taken to select a Fortran format that will display the desired precision without overflowing the output field width. When in doubt about which code to use, print-format codes zero, one, and two are advised; these codes specify the use of the Fortran general format control. If a negative value for the print-format code is entered, the data will be written to an unformatted file.

SAMPLE INPUT

Input Command File

```
EXAMPLE OF WRIT COMMAND
USED TO OUTPUT DATA TO A FORTRAN UNIT NUMBER
TITL READING COMPUTED HEADS FOR THE END OF THE FIRST TIME STEP
****     1    1    2    2    3    3    4    4    5    5    6    6    7
****5    0    5    0    5    0    5    0    5    0    5    0    5    0
READ HEAD          1 1
TITL WRITING COMPUTED HEADS, LAYER 1 FOR USE IN A CONTOURING PROGRAM
WRIT HEAD  01 000000 55
```

SAMPLE OUTPUT

Output Print File

```
SAMPLE----3 LAYERS, 15 ROWS, 15 COLUMNS, STEADY STATE, CONSTANT HEADS COLUMN 1, LAYERS 1 AND 2, RECHARGE, WELLS AND DRAINS
EXAMPLE OF WRIT COMMAND,
USED TO OUTPUT DATA TO A FORTRAN UNIT NUMBER

WRITING COMPUTED HEADS, LAYER 1 FOR USE IN A CONTOURING PROGRAM

Processing:  WRIT HEAD   01 000000 55



        WRITING OF : HEAD   - COMPUTED HEADS
                              LAYER   1
                              STRESS PERIOD   1
                              TIME STEP       1
                              ON UNIT:  55
                              USING FORMAT CODE:
```

Output Disk File

```
      55          0.(10G11.4)                    -1
  0.0000      24.95      44.01      59.26      71.82      82.52      91.91      100.0      106.9      112.6
   117.4      121.3      124.3      126.4      127.4
  0.0000      24.45      43.10      57.98      70.17      80.57      90.12      98.40      105.3      111.0
   115.7      119.6      122.7      124.9      126.1
  0.0000      23.45      41.30      55.43      66.78      76.21      86.51      95.20      102.2      107.6
   112.0      116.1      119.6      122.1      123.4
  0.0000      21.92      38.61      51.75      61.79      68.03      81.34      90.75      97.64      102.5
   106.1      110.7      114.9      117.9      119.4
  0.0000      19.73      34.92      47.32      57.69      66.74      77.09      85.76      92.22      96.15
   97.29      103.1      108.8      112.5      114.3
  0.0000      16.51      29.50      40.90      51.30      61.21      71.19      79.85      86.47      90.82
   93.03      94.23      102.1      106.4      108.4
  0.0000      11.55      21.10      31.21      41.40      51.84      63.08      72.68      79.95      84.92
   88.60      91.66      96.43      99.82      101.8
  0.0000      3.483      6.832      18.25      26.30      36.97      52.59      64.31      72.52      77.25
   81.99      85.00      89.27      91.72      94.33
  0.0000      10.54      19.11      28.12      36.92      45.27      52.95      55.38      65.15      66.07
   73.93      73.79      80.84      80.17      86.49
  0.0000      14.62      25.86      35.38      43.49      50.11      54.93      57.55      62.95      65.55
   70.39      72.44      76.72      78.26      81.79
  0.0000      17.11      29.96      40.01      47.78      53.24      55.81      53.33      60.27      59.29
   66.43      65.45      72.22      71.04      77.62
  0.0000      18.68      32.56      43.07      50.81      55.92      58.33      58.47      61.93      63.18
   67.12      68.50      72.29      73.46      76.85
  0.0000      19.67      34.24      45.14      53.01      58.04      59.91      56.75      62.59      60.91
   67.22      65.75      71.90      70.35      76.48
  0.0000      20.27      35.27      46.48      54.61      60.08      63.17      64.52      67.25      68.79
   71.64      73.18      75.84      77.03      79.09
  0.0000      20.56      35.78      47.16      55.48      61.26      65.02      67.52      69.94      72.01
   74.29      76.22      78.22      79.66      80.82
```

COMMENTS

In this example, Fortran unit 55 had been opened with job control directives prior to beginning the MMSP program. The READ command was used to read the computed heads calculated and written to an unformatted file by the Modular Model. The output of the READ command is in a previous section of this report.

Descriptive Statistics

DISCUSSION

Often a computed statistical description of a data array is desirable. These statistics can be used to provide a quick check of the validity of data that are input to the Modular Model. Also, statistics can be used to summarize data arrays computed and written by the Modular Model.

When statistics are computed by the MMSP program, the following values are written to the print file: arithmetic mean, mean absolute value, geometric mean, harmonic mean, root mean square, variance, minimum value, maximum value, sum of values, standard deviation, mean deviation, the number of non-missing values, moment-skewness coefficient, lower quartile, median, upper quartile, and the non-parametric skewness coefficient.

Printed at the end of the statistical summary are the row, column, and layer locations of the minimum, maximum, and median values in the model grid. When the values for the minimum, maximum, and median are not unique, the program prints the location of the first node in the simulation with the corresponding value. When an even number of values are summarized, the locations of the two nodes whose values are averaged to compute the median are printed. When a two-dimensional data array is summarized using the STAT command, layer one always will be printed for the layer location.

Variance is computed by summing the squares of the deviations of the observations from the mean and dividing by the number of observations minus one. The moment-skewness coefficient is computed by the following formula (Chow, V.T., 1964).

$$\text{SKEWNESS} = \frac{N}{(N-1)\ (N-2)}\ \frac{\sum (x - \bar{x})^3}{S^3}$$

where N = number of observations,
$\bar{x}$ = mean value, and
S = standard deviation.

Non-parametric skewness is computed by the following formula (David, H.A., 1962).

$$\text{SKEWNESS} = \frac{P_{75} + P_{25} - 2\ (P_{50})}{P_{75} - P_{25}}$$

where P_{75} = upper quartile,
P_{50} = median, and
P_{25} = lower quartile.

The MMSP program will not compute some statistics when specific conditions occur. If any of the values in the data array are less than or equal to zero geometric and harmonic means are not computed. If the number of observations is less than 3 or the standard deviation is zero the moment-skewness coefficient is not computed. If the upper and lower quartiles are equal the non-parametric skewness is not computed. When these conditions occur, the MMSP program prints a warning message and sets the value of the statistic to zero.

INPUT INSTRUCTIONS

Field:	Command	DSN	LAYER	MASK
Beginning column:	1	6	13	16
Data Format:	STAT	A6	I2	I6

To produce descriptive statistics of a data array, include the STAT command in the command file and specify the DSN of the data array to be summarized. The operation of the STAT command may be controlled by any combination of the options described in the following paragraphs.

A specific layer of a three-dimensional data array may be described by specifying the layer number with the STAT command. If the layer number is blank or zero all layers are included in the statistical analysis.

The three masks, zero, IBOUND, and UBOUND, discussed in chapter 2, Subsetting Data Arrays, can be used to exclude values from the statistical summary. Individual masks are enabled or disabled by placing integer values in the appropriate mask fields (fig. 3). Unlike the PRIN and WRIT commands, the use of the masks does not cause the missing-value indicators to replace values in the data array. Instead, when a value is masked during the calculation of descriptive statistics, the value is removed from the set of data used in the computation of the statistics. When masking is performed, a summary of the number of values excluded from the analysis by each mask is printed.

SAMPLE INPUT

Input Command File

```
EXAMPLE OF THE STAT COMMAND

USED TO PRODUCE DESCRIPTIVE STATISTICS OF A DATA ARRAY

TITL RECHARGE APPLIED TO INACTIVE AND CONSTANT HEAD NODES
****     1    1    2    2    3    3    4    4    5    5    6    6    7
****5    0    5    0    5    0    5    0    5    0    5    0    5    0
STAT RECH   00 00-100
```

SAMPLE OUTPUT

Output Print File

```
SAMPLE----3 LAYERS, 15 ROWS, 15 COLUMNS, STEADY STATE, CONSTANT HEADS COLUMN 1, LAYERS 1 AND 2, RECHARGE, WELLS AND DRAINS
EXAMPLE OF THE STAT COMMAND
USED TO PRODUCE DESCRIPTIVE STATISTICS OF A DATA ARRAY

RECHARGE APPLIED TO INACTIVE AND CONSTANT HEAD NODES

Processing:  STAT RECH   00 00-100



        STATISTICS FOR : RECH   - RECHARGE RATE
                                  ALL LAYER(S)
                                  STRESS PERIOD   1
                                  TIME STEP       1



        Masking was performed on RECH

      30 points remain out of      675

     645 points excluded which were active nodes



        Zero or negative values present in matrix, therefore geometric and harmonic means were not computed


        ARITHMETIC     ABSOLUTE VALUE      GEOMETRIC          HARMONIC          ROOT MEAN
           MEAN             MEAN                MEAN               MEAN              SQUARE            VARIANCE
         0.150000E-07     0.150000E-07      0.000000           0.000000           0.212132E-07     0.232759E-15

                                               SUM OF            STANDARD             MEAN           NUMBER OF
           MINIMUM          MAXIMUM            VALUES           DEVIATION          DEVIATION           VALUES
         0.000000         0.300000E-07      0.450000E-06       0.152564E-07       0.150000E-07          30

        COEFFICIENT          LOWER                                  UPPER       NON-PARAMETRIC
        OF SKEWNESS         QUARTILE            MEDIAN           QUARTILE            SKEWNESS
         -0.381576E-13     0.000000          0.150000E-07       0.300000E-07       0.000000


                                                   LOCATION
                                      STATISTIC   ROW COLUMN  LAYER

                                        MINIMUM    15      1      2

                                        MAXIMUM    13      1      1

                                         MEDIAN     1      1      2

                                         MEDIAN    15      1      1
```

Frequency Analysis

DISCUSSION

Frequency analysis can be used to check the distribution of values in a data array. Frequency analysis is useful for making a quick check of data arrays that are input to or computed and written by the Modular Model.

The method employed by the MMSP program to perform a frequency analysis sorts the data array in ascending order of magnitude, organizes the values into classes bounded by an upper and lower magnitude, counts and prints a table of the number of values in each of the classes, and prints a histogram of the distribution of data in each of the classes.

The boundaries of the classes into which data values are grouped can be defined using one of three methods with the MMSP program. By the first method, the program computes an arithmetic progression for the class boundaries beginning with the minimum data value. By the second method, the program computes a logarithmic progression of class boundaries. With the third method, the user specifies the class boundaries that are to be used in the frequency analysis. Regardless of which method is selected, the MMSP program requires that at least three and no more than 20 classes be used for tallying the data values.

INPUT INSTRUCTIONS

Field:	Command	DSN	LAYER	MASK	NUMBER OF CLASSES
Beginning column:	1	6	13	16	23
Data Format:	HIST	A6	I2	I6	I3

To produce a frequency analysis of a data array, include the HIST command in the command file and specify the DSN of the data array to be summarized. The operation of the HIST command may be controlled by any combination of the following options.

A specific layer of a three-dimensional data array may be analyzed by specifying the layer number with the HIST command. If the layer number is blank or zero all layers are included in the frequency analysis.

The three masks, zero, IBOUND, and UBOUND, discussed in chapter 2, Subsetting Data Arrays, can be used to exclude values from the frequency analysis. Individual masks are enabled or disabled by placing integer values in the appropriate mask fields (fig. 3). As with the STAT command, when a value is masked during the frequency analysis, the value is removed from the set of data used in the computations. When masking is performed, a summary of the number of values excluded from the analysis by each mask is printed.

From 3 to 20 classes can be selected. If an arithmetical progression of class boundaries is desired, the number of classes should be specified as a positive integer. The MMSP program will compute the class boundaries for the number of classes entered. The class boundaries will be equally spaced between the maximum and minimum values.

If a logarithmic progression of class boundaries is desired, the number of classes should be specified as a negative integer. In this case, the program computes the number of class boundaries and the value of the negative integer entered with the HIST command has no effect on the class boundary computations.

If specific class boundaries are desired, the class boundaries first must be defined using the READ command for the DSN CLASS. (Refer to the discussion of the READ command in this chapter.) To use these specific classes in the frequency analysis, the number of classes should be specified as a number between zero and three. Any of the values: 0, 1, or 2 may be used to indicate that user-specific class boundaries are to be used. When one of these values is entered, the number of classes used for the frequency analysis will be one greater than the number of cut points entered with the READ command.

If the number-of-classes field is blank or zero, and if user-specified cut points have not been defined using the READ command, the MMSP program will print a message and compute 20 arithmetically spaced classes.

SAMPLE INPUT

Input Command File

```
EXAMPLE OF HIST COMMAND
USED TO PERFORM FREQUENCY ANALYSIS ON A DATA ARRAY
****    1    1    2    2    3    3    4    4    5    5    6    6    7
****5   0    5    0    5    0    5    0    5    0    5    0    5    0
READ HEAD               1 1
TITL FREQUENCY DISTRIBUTION OF COMPUTED HEADS: ALL LAYERS, ACTIVE NODES
HIST HEAD    00 000100
```

SAMPLE OUTPUT

Output Print File

```
SAMPLE----3 LAYERS, 15 ROWS, 15 COLUMNS, STEADY STATE, CONSTANT HEADS COLUMN 1, LAYERS 1 AND 2, RECHARGE, WELLS AND DRAINS
EXAMPLE OF HIST COMMAND
USED TO PERFORM FREQUENCY ANALYSIS ON A DATA ARRAY

FREQUENCY DISTRIBUTION OF COMPUTED HEADS: ALL LAYERS, ACTIVE NODES

Processing:  HIST HEAD  00 000100



          HISTOGRAM FOR : HEAD   - COMPUTED HEADS
                                   ALL LAYER(S)
                                   STRESS PERIOD   1
                                   TIME STEP       1



         Masking was performed on HEAD

      645 points remain out of      675

       30 points excluded that were inactive or constant head nodes
         User-specified classes requested, but have not  been read; using 20 arithmetic computed classes


         CLASS      GREATER THAN  LESS THAN OR EQUAL TO         POPULATION

           1                          0.433095                       1.
           2          0.433095         7.48942                      18.
           3           7.48942        14.5457                       10.
           4          14.5457         21.6021                       32.
           5          21.6021         28.6584                       19.
           6          28.6584         35.7147                       27.
           7          35.7147         42.7711                       24.
           8          42.7711         49.8274                       29.
           9          49.8274         56.8837                       50.
          10          56.8837         63.9400                       68.
          11          63.9400         70.9964                       64.
          12          70.9964         78.0527                       81.
          13          78.0527         85.1090                       46.
          14          85.1090         92.1653                       43.
          15          92.1653         99.2217                       22.
          16          99.2217        106.278                        29.
          17         106.278         113.334                        28.
          18         113.334         120.391                        27.
          19         120.391         127.447                        27.
          20         127.447                                         0.
```

Output Print File--Continued

```
FREQUENCY------------------------------------------------------------------------------------------------------
   81  *                                                           I                                          *
   79  *                                                           I                                          *
   77  *                                                           I                                          *
   75  *                                                           I                                          *
   73  *                                                           I                                          *
   71  *                                                           I                                          *
   69  *                                                           I                                          *
   67  *                                                 I         I                                          *
   65  *                                                 I         I                                          *
   63  *                                                 I    I    I                                          *
   61  *                                                 I    I    I                                          *
   59  *                                                 I    I    I                                          *
   57  *                                                 I    I    I                                          *
   55  *                                                 I    I    I                                          *
   53  *                                                 I    I    I                                          *
   51  *                                                 I    I    I                                          *
   49  *                                            I    I    I    I                                          *
   47  *                                            I    I    I    I                                          *
   45  *                                            I    I    I    I    I                                     *
   43  *                                            I    I    I    I    I    I                                *
   41  *                                            I    I    I    I    I    I                                *
   39  *                                            I    I    I    I    I    I                                *
   37  *                                            I    I    I    I    I    I                                *
   35  *                                            I    I    I    I    I    I                                *
   33  *                                            I    I    I    I    I    I                                *
   31  *                   I                        I    I    I    I    I    I                                *
   29  *                   I                   I    I    I    I    I    I    I         I                      *
   27  *                   I         I         I    I    I    I    I    I    I         I    I    I    I       *
   25  *                   I         I         I    I    I    I    I    I    I         I    I    I    I       *
   23  *                   I         I    I    I    I    I    I    I    I    I         I    I    I    I       *
   21  *                   I         I    I    I    I    I    I    I    I    I    I    I    I    I    I       *
   19  *                   I    I    I    I    I    I    I    I    I    I    I    I    I    I    I    I       *
   17  *         I         I    I    I    I    I    I    I    I    I    I    I    I    I    I    I    I       *
   15  *         I         I    I    I    I    I    I    I    I    I    I    I    I    I    I    I    I       *
   13  *         I         I    I    I    I    I    I    I    I    I    I    I    I    I    I    I    I       *
   11  *         I         I    I    I    I    I    I    I    I    I    I    I    I    I    I    I    I       *
    9  *         I    I    I    I    I    I    I    I    I    I    I    I    I    I    I    I    I    I       *
    7  *         I    I    I    I    I    I    I    I    I    I    I    I    I    I    I    I    I    I       *
    5  *         I    I    I    I    I    I    I    I    I    I    I    I    I    I    I    I    I    I       *
    3  *         I    I    I    I    I    I    I    I    I    I    I    I    I    I    I    I    I    I       *
    1  *    I    I    I    I    I    I    I    I    I    I    I    I    I    I    I    I    I    I    I       *
 ---------------------------------------------------------------------------------------------------------------
 CLASS      1    2    3    4    5    6    7    8    9   10   11   12   13   14   15   16   17   18   19   20


           HISTOGRAM FOR : HEAD   - COMPUTED HEADS
                                    ALL LAYER(S)
                                    STRESS PERIOD   1
                                    TIME STEP       1
```

COMMENTS

In this example, the READ command was used to read the binary computed heads calculated and written by the Modular Model. The printed output of the MMSP READ command was shown previously, and is not repeated here. The HIST command did not specify the number of classes to be used, the program read this field as a zero, and found that user-specified cut points had not been read. Therefore, 20 arithmetically-spaced classes were used.

Comparison of Data Arrays

DISCUSSION

When checking for errors in Modular Model input data, it can be useful to determine if values in a data array exceed some value. For example, when simulating a confined aquifer, a modeler might wish to ensure that none of the values for the storage coefficient are greater than 0.005. Another example of validating the Modular Model input data is to check if the altitudes of the aquifer bottom is greater than the altitudes of the aquifer top at any model nodes.

Likewise, a modeler can test hypotheses regarding the results of the simulation by performing logical comparisons between data arrays. An example is to compare the computed heads for consecutive time steps to determine where heads have decreased.

The COMP command performs a node-by-node comparison of the values in two data arrays and produces a tabular listing of the row-column-layer locations and values of model nodes that pass the logical comparison test.

INPUT INSTRUCTIONS

Field:	Command	DSN1	LAYER	OPERATOR	DSN2	LAYER	MASK	LIMIT
Beginning column:	1	6	13	16	19	26	29	36
Data Format:	COMP	A6	I2	A2	A6	I2	I6	I4

To compare two data arrays, include the COMP command in the command file and specify the DSN for each of the two data arrays in the DSN1 and DSN2 fields. Specify a logical operator for the comparison in the OPERATOR field of the COMP command. Valid logical operators available for use with the COMP command are shown in the following list. Each operator may be expressed by any of two or three syntaxes in the COMP command.

LOGICAL OPERATOR	OPERATION
= or EQ	equal to
^= or NE	not equal to
< or LT	less than
> or GT	greater than
<= or =< or LE	less than or equal to
>= or => or GE	greater than or equal to

When both data-set names specified in a COMP command are the same, and the data arrays are stored in either the two-dimensional or three-dimensional stack, the program will search the stacks to find whether there are two data arrays with the same DSN stored in the same stack. When two arrays with same DSN are requested and found in a stack, the program will use both data arrays for the comparison. Time-series analysis of data arrays output by the Modular Model may be performed by using this capability. This searching procedure is described in chapter 2, Data-set Names.

For example, a modeler might want to determine where water levels decreased after a stress was applied during a transient simulation. Two READ commands would be used to read the computed heads. The first READ command would read HEAD for the final time step of the stress period before the stress was applied, and the second READ command would read HEAD for last time step of the stress period when the stress was applied. Two data arrays named HEAD would be on the three-dimensional stack. The comparision command:

```
COMP HEAD   00  < HEAD   00 000100
```

would list all active nodes where heads had decreased.

The operation of the COMP command may be controlled by any combination of the following options.

The comparision may be restricted to a specific layer of a three-dimensional data array by specifying the layer number adjacent to the corresponding DSN field of the COMP command. If the layer number is blank or zero all layers are included in the comparison. To compare two data arrays, both arrays must contain the same number of values. Thus, two two-dimensional data arrays, two three-dimensional data arrays, or two layers may be compared on a node-by-node basis.

All layers in the model simulation may not exist for the data arrays: altitude of the aquifer top (TOP), altitude of the aquifer bottom (BOT), computed heads (HEAD), and computed drawdown (DRAWDN); depending upon the specifications of the layer type array (LAYCON), and the save-flag array (IOFLG). When a COMP command specifies layer zero for both data arrays, and a layer does not exist for one of the data arrays, the program skips the layer and resumes the comparison for the next layer that is present in both data arrays.

If the COMP command specifies a specific layer, and the layer is not present for the corresponding data array, the command prints an error message and the comparison is not performed.

The two masks: IBOUND and UBOUND, discussed in chapter 2, Subsetting Data Arrays, can be used to exclude values from a comparison. Individual masks are enabled or disabled by placing integer values in the appropriate mask fields (fig. 3). When a value is masked during the comparison, the logical testing of the values of the two data arrays is skipped. When masking is performed, a summary of the number of values excluded from the comparison is printed. The zero mask is not available for use with the COMP command, because there are two data arrays. If the zero mask is requested, a warning message will be printed and the comparison will continue without using a zero mask.

Care should be taken with the specifications of the COMP command for simulations with a large number of model nodes. An inappropriately prepared comparison command can produce a huge print file. The LIMIT field of the COMP command may be used to control the maximum number of nodes that will be printed by the program when the comparision condition is true. The number of nodes printed by the COMP command is not limited if blanks or zero is entered for the LIMIT field.

SAMPLE INPUT

Input Command File

```
EXAMPLE OF COMP COMMAND, USED TO COMPARE ONE DATA ARRAY WITH ANOTHER.
EXAMINING LAYER ONE, ACTIVE NODES
****    1    1    2    2    3    3    4    4    5    5    6    6    7
****5   0    5    0    5    0    5    0    5    0    5    0    5    0
READ RIFACE          1  1
READ LOFACE          1  1
TITL WHERE IS THE LEFT->RIGHT FLOW PREDOMINANT OVER DOWNWARD FLOW?
COMP RIFACE 01 GT LOFACE 01 000100
```

SAMPLE OUTPUT

Output Print File

```
SAMPLE----3 LAYERS, 15 ROWS, 15 COLUMNS, STEADY STATE, CONSTANT HEADS COLUMN 1, LAYERS 1 AND 2, RECHARGE, WELLS AND DRAINS
EXAMPLE OF COMP COMMAND, USED TO COMPARE ONE DATA ARRAY WITH ANOTHER.
EXAMINING LAYER ONE, ACTIVE NODES

WHERE IS THE LEFT->RIGHT FLOW PREDOMINANT OVER DOWNWARD FLOW?

Processing:  COMP RIFACE 01 GT LOFACE 01 000100



         COMPARISON OF : RIFACE - RIGHT-FACE CBC FLOW
                                  LAYER   1
                                  STRESS PERIOD   1
                                  TIME STEP       1


               AGAINST : LOFACE - LOWER-FACE CBC FLOW
                                  LAYER   1
                                  STRESS PERIOD   1
                                  TIME STEP       1

                 WHERE : RIFACE GT LOFACE

        COMPARISON TEST BYPASS CONDITIONS:

        when model boundary nodes are inactive or constant head


                         --------- RIFACE ---------      --------- LOFACE ---------
    ROW     COLUMN      LAYER               VALUE       LAYER                VALUE

      9         13          1          0.153772             1        0.338018E-001
     11          7          1          0.506731             1        0.633933E-001
     11          9          1          0.204340             1        0.309618E-001
     11         11          1          0.212355             1        0.302935E-001
     11         13          1          0.262021             1        0.296655E-001
     13          7          1          0.657696             1        0.631353E-001
     13          9          1          0.355808             1        0.317633E-001
     13         11          1          0.317563             1        0.309513E-001
     13         13          1          0.343022             1        0.301595E-001

                  9 nodes identified
                 15 nodes ignored via mask options
                675 nodes in data set
```

COMMENTS

In this example, the READ command was used to read the binary cell-by-cell flow terms computed and written by the Modular Model. The printed output of the MMSP READ command is not shown here.

Mathematical Operations

DISCUSSION

Some applications of the MMSP program may require the computation of a data array by a mathematical operation. This capability is provided with the MATH command. Using the MATH command, the drawdown may be calculated between any two time steps where heads have been saved during the simulation. With another application of the MATH command, the saturated thickness in an unconfined aquifer may be calculated by subtracting the altitude of the head from the altitude of the aquifer bottom.

The MATH command uses one or two data arrays and a mathematical operator to compute a new data array. The resultant data array is placed on either the two-dimensional or the three-dimensional stack, depending upon the size of the array. The computed data array may be used in subsequent MMSP commands for further analysis. The MATH command does not perform any analysis or printing of the computed data array.

Data arrays are computed by the MATH command by performing the desired calculation on a node-by-node basis. Therefore, when a new data array is calculated from two data arrays, both input arrays must contain the same total number of model nodes.

The MATH command can be used to copy well pumpage rates (WELL), recharge rates (RECH), or recharge fluxes (RECHF) from reserved storage locations to the stack storage locations. (Refer to chapter 2, Modular Model Data Arrays and Auxiliary Data Arrays for a discussion of storage locations in the MMSP program.) For example, it would be necessary to copy RECH and give it a different DSN, in order to compare recharge rates from one stress period to another. The MATH command can copy the data array onto the stack by multiplying the desired data array by one, or by adding zero. Then the READ command can be used to read recharge rates for another stress period and the two data arrays can be compared.

INPUT INSTRUCTIONS

Fields:	Command	DSN1	LAYER	OPERATOR	DSN2	LAYER	DSN3	ARRAY-NAME
Beginning column:	1	6	13	16	19	26	29	36
Data format:	MATH	A6	I2	A2	A6	I2	A6	A24

To calculate a data array, include the MATH command in the command file and specify the DSN for each data array needed in the DSN1 and DSN2 fields. Specify an arithmetic operator for the computation in the OPERATOR field of the MATH command. Valid arithmetic operators for use with the MATH command are shown below. Each operator may be expressed by either of the two syntaxes shown.

ARITHMETIC OPERATOR	OPERATION
+ or AD	addition
- or SU	subtraction
* or MU	multiplication
/ or DI	division
** or EX	exponentiation
|| or AB	absolute value

Only the absolute value operator does not require two data-set names. When the absolute value operator is used, the DSN of the array to be operated upon should be placed in the DSN1 field.

The DSN3 field must be used with all operators to name the new data array that will be placed on a stack. This DSN will be used in subsequent commands when referring to the calculated data array. Optionally, a textual description of the calculated data array may be supplied in the ARRAY-NAME field. This textual description will be printed with subsequent commands that use the calculated data array. The ARRAY-NAME is printed when the calculated data array is used during an operation and is not used by the program for identifying the array. The program uses the DSN3 field for identifying the array.

When both data-set names input to the MATH command are the same, and the data arrays are stored in either the two-dimensional or three-dimensional stack, the program will search the stacks to find whether there are two data arrays with the same DSN stored in the same stack. When two arrays with same DSN are requested and found in a stack, the program will use both data arrays for the computation. This searching procedure is described in chapter 2, Data-set Names.

To compute a two-dimensional data array from a layer of one or two three-dimensional data arrays, specify the layer number adjacent to the corresponding DSN field of the MATH command. If the layer number is blank or zero all layers are included in the computation.

All layers in the model simulation may not exist for the data arrays: altitude of the aquifer top (TOP), altitude of the aquifer bottom (BOT), computed heads (HEAD), and computed drawdown (DRAWDN); depending upon the specifications of the layer type array (LAYCON), and the save-flag array (IOFLG). When a MATH command specifies layer zero for both data arrays, and a layer does not exist for one of the data arrays, the program fills the layer in the calculated data array with zeroes, prints a message, and resumes the computation for the next layer that exists in both data arrays.

If the MATH command specifies a layer that does not exist for the corresponding data array, the command prints an error message and the computation is not performed.

SAMPLE INPUT

Input Command File

```
EXAMPLE OF MATH COMMAND, USED TO CALCULATE A DATA ARRAY:

VERTICAL VELOCITY = LOWER FACE FLOW / CELL AREA

****    1    1    2    2    3    3    4    4    5    5    6    6    7
****5   0    5    0    5    0    5    0    5    0    5    0    5    0
READ LOFACE         1  1
TITL CALCULATING VERTICAL VELOCITY FOR LAYER 1:  SPEED
MATH LOFACE 01   / AREA    01 SPEED   VERTICAL VELOCITY
STAT SPEED  01     1
```

SAMPLE OUTPUT

Output Print File

```
SAMPLE----3 LAYERS, 15 ROWS, 15 COLUMNS, STEADY STATE, CONSTANT HEADS COLUMN 1, LAYERS 1 AND 2, RECHARGE, WELLS AND DRAINS
EXAMPLE OF MATH COMMAND, USED TO CALCULATE A DATA ARRAY:
VERTICAL VELOCITY = LOWER FACE FLOW / CELL AREA

CALCULATING VERTICAL VELOCITY FOR LAYER 1:  SPEED

Processing:  MATH LOFACE 01  / AREA   01 SPEED  VERTICAL VELOCITY



        COMPUTATION OF : SPEED  - VERTICAL VELOCITY


                 FROM : LOFACE - LOWER-FACE CBC FLOW
                                 LAYER   1
                                 STRESS PERIOD   1
                                 TIME STEP       1


            DIVIDED BY : AREA   - AREA OF CELLS
                                 LAYER   1



       TWO-DIMENSIONAL STACK CONTENTS AFTER MATH COMMAND

         STACK   DATA SET STRESS TIME
        POSITION   NAME   PERIOD STEP DESCRIPTION

            4               --    --
            3               --    --
            2               --    --
            1     SPEED     --    --  VERTICAL VELOCITY
```

COMMENTS

This example shows the use of the READ, MATH, and STAT commands to examine of vertical flow velocities between layers 1 and 2. The printed output from the READ and STAT commands are not shown. The vertical velocity is computed by dividing the lower-face flow by the area for each cell. Units of velocity are determined by the units used in the simulation. In the sample simulation, distance is expressed in feet and time is expressed in seconds. Therefore the computed velocities have units of feet per second.

Subsetting Computed Heads

DISCUSSION

When analyzing computed heads, it is frequently useful to extract the simulated values of head at selected model nodes for every time step when heads were saved. These values can be entered into a statistical package for verification analysis or into a graphical display program for hydrographs of observed versus computed head.

The MMSP HEAD command will write computed heads to a Fortran unit for selected model nodes from the binary output file created by the Modular Model. Up to six model nodes may be specified with each HEAD command, and the command may be repeated as many times as needed.

When using the HEAD command, the modeler must provide the number of a Fortran unit that can be used for writing the data. The Fortran unit must have been opened for writing by job control directives.

The HEAD command will write a record to the output file unit for each model node specified, for every time step. Each record consists of the row, column, layer, stress period, time step, time since the beginning of the simulation, time since the beginning of the stress period, and the computed head. For some time steps, computed heads may not be written by the Modular Model, due to the specifications given for IOFLG in the output-control package. In this situation, the HEAD command writes a record for each requested model node, with the value for computed head set equal to the first missing value indicator, and writes a message to the print file.

Several steps should be taken when preparing the specifications for the Modular Model, if the MMSP HEAD command will be used with the results of the simulation. Job control directives must open a file for saving unformatted head data and the Fortran file unit number of this file should be provided for the variable IHEDUN in the output-control data for the Modular Model. Also, values for the variables INCODE, IHDDFL, HDSV must be set properly for each time step in the output-control data.

INPUT INSTRUCTIONS

Field:	Command	UNIT	up to six node locations (ROW-COLUMN-LAYER)
Beginning column:	1	6	9, 21, 33, 45, 57, 69
Data Format:	HEAD	I2	6(I3,1X,I3,1X,I3,1X)

To create a disk file containing computed heads at selected model nodes for all time steps, include the HEAD command in the command file, and specify the number of a Fortran unit to be used for writing the data. Specify the row-column-layer locations of the model nodes for which computed heads are to be recorded.

SAMPLE INPUT

Input Command File

```
EXAMPLE OF HEAD COMMAND

USED TO OUTPUT SELECTED HEAD DATA TO A FORTRAN UNIT NUMBER

TITL EXTRACTING COMPUTED HEADS FOR THE FIRST TIME STEP
****     1    1    2    2    3    3    4    4    5    5    6    6    7
****5    0    5    0    5    0    5    0    5    0    5    0    5    0
HEAD 55 001 001 001    5   11    3   12 12    1
```

SAMPLE OUTPUT

Output Print File

```
SAMPLE----3 LAYERS, 15 ROWS, 15 COLUMNS, STEADY STATE, CONSTANT HEADS COLUMN 1, LAYERS 1 AND 2, RECHARGE, WELLS AND DRAINS
EXAMPLE OF HEAD COMMAND
USED TO OUTPUT SELECTED HEAD DATA TO A FORTRAN UNIT NUMBER

EXTRACTING COMPUTED HEADS FOR THE FIRST TIME STEP

Processing:  HEAD 55 001 001 001    5  11    3  12 12    1



          WRITING OF : COMPUTED HEADS
             ON UNIT : 55
        USING FORMAT : (5I5,3G15.6E2)
                     1 TIME STEP(S)
                     3 NODES

          ROW    COLUMN     LAYER
            1        1         1
            5       11         3
           12       12         1
```

Output Disk File

```
    1    1    1    1    1    86400.0        86400.0        0.000000
    5   11    3    1    1    86400.0        86400.0         77.4623
   12   12    1    1    1    86400.0        86400.0         68.4991
```

COMMENTS

In this example, Fortran unit 55 had been opened with job control directives prior to beginning the program. The example problem is a steady-state simulation, therefore head data for a single time step were written by the HEAD command.

Resetting the Model-boundary Array

DISCUSSION

During a simulation, the Modular Model changes an active model node to an inactive node when the saturated thickness at the node becomes zero. When the model deactivates a model node, it sets the head at the node equal to 10^{30} (McDonald and Harbaugh, 1988, p. 5-75). During post-processing, this large value for head can cause erroneous calculations, unless it is excluded from the analysis using an updated IBOUND mask to indicate where nodes have been deactivated.

The MMSP program provides the REBO command for updating the IBOUND array by resetting the values for deactivated model nodes to zero. To perform this operation, computed heads must be saved to an unformatted file by the Modular Model. Several steps should be taken when preparing the specifications for the Modular Model, if the MMSP REBO command will be used with the results of the simulation. Job control directives must open a file for saving unformatted head data and the Fortran file unit number of this file should be provided for the variable IHEDUN in the output-control package of the Modular Model. Also, values for the variables INCODE, IHDDFL, HDSV must be set properly for each time step in the output-control package.

The REBO command can be specified for any stress period and time step in the simulation when computed heads were saved. The REBO command will write a table of all node locations that were deactivated by the specified stress period and time step. Because the Modular Model does not re-wet nodes during a simulation, the listing of nodes produced by the REBO command indicates all nodes that have gone dry during any time step before or during the specified time step.

INPUT INSTRUCTIONS

		STRESS	TIME
Field:	Command	PERIOD	STEP
Beginning column:	1	6	9
Data Format:	REBO	I2	I2

To reset the model-boundary array, include the REBO command in the command file, and specify the stress period and time step to be used.

SAMPLE INPUT

Input Command File

```
EXAMPLE OF REBO COMMAND
USED TO UPDATE THE MODEL-BOUNDARY ARRAY (IBOUND)
****    1   1   2   2   3   3   4   4   5   5   6   6   7
****5   0   5   0   5   0   5   0   5   0   5   0   5   0
REBO  1  1
```

SAMPLE OUTPUT

Output Print File

```
SAMPLE----3 LAYERS, 15 ROWS, 15 COLUMNS, STEADY STATE, CONSTANT HEADS COLUMN 1, LAYERS 1 AND 2, RECHARGE, WELLS AND DRAINS
EXAMPLE OF REBO COMMAND
USED TO UPDATE THE MODEL-BOUNDARY ARRAY (IBOUND)



Processing:  REBO  1  1



            RESETTING OF : MODEL-BOUNDARY ARRAY
        FOR STRESS PERIOD :  1
                TIME STEP :  1

      THE FOLLOWING NODES HAVE GONE DRY

            ROW     COLUMN     LAYER

  0 NODE(S) WENT DRY
```

COMMENTS

In this example, no nodes were deactivated during the steady-state simulation. Therefore, the REBO command had no effect on the model-boundary array.

Defining Layer Thickness

DISCUSSION

The THIC command defines the thickness of model layers. Layer thickness is defined in a manner analogous to the cell width along rows (DELR) and columns (DELC) in the block-centered flow package of the Modular Model. Layer thickness is given for each layer in the simulation. Accordingly, each model node is located in the center of the cell defined by the corresponding cell widths and layer thickness.

The VECT command of the MMSP program calculates flow vectors for nodes in the simulation grid that intersect a plane defined with the SLIC command. (The VECT and SLIC commands are discussed in subsequent sections of this chapter.) Before defining a plane that does not intersect the simulation grid horizontally, it may be useful to define the thickness of the model layers. Only the SLIC and the VECT commands of the program use layer thicknesses for computations. The program uses the thicknesses of model layers provided with the THIC command for determining the vertical distance between nodes when calculating the positions of flow vectors. It is not necessary to enter the "actual" thickness of the layers being simulated with the THIC command. In some cases it may be useful to alter the vertical positioning of flow vectors within a cross-section using the THIC command.

Although the Modular Model does not explicitly use a value for layer thickness and layer thickness can be variable throughout the simulation grid, the MMSP program uses a single value of layer thickness for each layer, which is used for every node in the layer. This inconsistency should be of no concern to the modeler. The layer thickness data are used only by the MMSP program for placement of flow vectors in cross-sections that slice through more than one layer of the model grid. The alternative would be to require the modeler to enter a layer-thickness value for every node in the model grid, which could be extremely tedious for a large simulation.

Layer thickness values should be specified in the same units that are used for DELR and DELC.

If the THIC command is not used to define the thickness of layers, the thicknesses of all layers will be given a default value equal to the average width of cells.

INPUT INSTRUCTIONS

Field:	Command	up to seven LAYER-THICKNESSES
Beginning column:	1	6, 16, 26, 36, 46, 56, 66
Data Format:	THIC	7(F10.0)

(repeat as necessary until all layers have been given a thickness value)

To define the thickness of layers, include the THIC command in the command file and specify the thickness of each layer in the simulation grid. Ten columns are provided for the entry of each layer thickness value. Up to seven layers may be defined on a single THIC command line. If there are more than seven layers in the simulation, continue specifying layer thicknesses with another THIC command until all layers have been assigned a thickness value. The THIC and the comment (****) commands are the only MMSP commands that may be continued on a subsequent command line.

SAMPLE INPUT

Input Command File

```
EXAMPLE OF THIC COMMAND
USED TO DEFINE LAYER THICKNESSES
****      1    1    2    2    3    3    4    4    5    5    6    6    7
****5     0    5    0    5    0    5    0    5    0    5    0    5    0
THIC        5000      5000      5000
```

SAMPLE OUTPUT

Output Print File

```
SAMPLE----3 LAYERS, 15 ROWS, 15 COLUMNS, STEADY STATE, CONSTANT HEADS COLUMN 1, LAYERS 1 AND 2, RECHARGE, WELLS AND DRAINS
EXAMPLE OF THIC COMMAND
USED TO DEFINE LAYER THICKNESSES


Processing:  THIC       5000       5000       5000


         LAYER     LAYER
         NUMBER  THICKNESS
         -----------------
            1    0.500E+04
            2    0.500E+04
            3    0.500E+04
```

Slicing Data Arrays

DISCUSSION

A slice of a data array is a plane that intersects the model grid. A plane is defined by any three non-colinear points. A modeler might wish to define a slice of a model grid for two reasons. First, to compute flow vectors with the MMSP progam it is necessary to define the plane that determines which nodes are to be included in the computations. Second, it can be useful to define a cross-sectional user boundary to be used as a mask with other MMSP commands. One application of this user-boundary mask is to calculate the flow through a cross-section.

Both of these objectives can be met with the SLIC command. The SLIC command will define a slice of the model grid along a column, row, or layer, or define a slice that intersects the model grid obliquely.

When performing the SLIC command, the program first checks if the three points lie along a column, row, or layer. If so, then the equation defining the plane is calculated and the user boundary is determined. Each value in the user-boundary array corresponding to nodes lying on the plane is set to one; for all other nodes, the value in the user-boundary array is set to zero.

If the three points specified with the SLIC command do not lie along any column, row, or layer, the slice is oblique to the model grid. To prepare a slice in this case, the program calculates the equations of two vectors lying in the plane and the equation of a vector normal to the plane. The equation of a plane is computed from these vectors. The user boundary is prepared by checking columns along each row on every layer to determine which nodes are closest to the slicing plane. Thus, on any layer-row combination, the column closest to the slicing plane is defined within the user boundary and the corresponding node of the user-boundary array are set equal to one. All other nodes on the column are outside the user boundary and are set to zero. The slice of the simulation grid prepared in this manner may or may not be a good representation of a cross-section of the study area depending on the fineness of the discretization of space in the simulation.

When the SLIC command is used, the MMSP program prints the points used to define the plane and the equation of the slicing plane. The coefficients of the equation are in the same units that were used to define the width of cells (such as feet or meters).

If an oblique slice is desired, but the three points specified with the SLIC command are colinear, an equation for the plane is indeterminate, an error message is printed, and a slice is not prepared. Similarly, if a plane is defined that does not intersect the simulation grid, an error message is written and a slice is not prepared.

INPUT INSTRUCTIONS

Field:	Command	three node locations ROW-COLUMN-LAYER
Beginning column:	1	6, 18, 30
Data Format:	SLIC	3(I3,1X,I3,1X,I3,1X)

To compute the equation of a slicing plane and prepare a user-boundary array that slices the model grid, include the SLIC command in the command file and specify the grid cell locations of three points that define the plane.

When defining a plane that intersects the simulation grid along a column, row, or layer an abbreviated method for specifying the slice may be used. In this case, only one coordinate needs to be specified. The desired column, row, or layer is entered in one of the fields provided in the SLIC command for the first node location. The other two fields for the first node location and the six fields for the remaining two nodes can be entered as zeroes or left blank.

SAMPLE INPUT

Input Command File

```
EXAMPLE OF SLIC COMMAND
USED TO DEFINE A USER-BOUNDARY WHICH SLICES THE MODEL GRID WITH A PLANE
****    1    1    2    2    3    3    4    4    5    5    6    6    7
****5   0    5    0    5    0    5    0    5    0    5    0    5    0
SLIC 000 000 001                                     LAYER ONE
```

SAMPLE OUTPUT

Output Print File

```
SAMPLE----3 LAYERS, 15 ROWS, 15 COLUMNS, STEADY STATE, CONSTANT HEADS COLUMN 1, LAYERS 1 AND 2, RECHARGE, WELLS AND DRAINS
EXAMPLE OF SLIC COMMAND
USED TO DEFINE A USER-BOUNDARY WHICH SLICES THE MODEL GRID WITH A PLANE



Processing:  SLIC 000 000 001                              LAYER ONE


         COMPUTING USER BOUNDARY FROM THE GRID COORDINATES:
           ROW     COLUMN    LAYER
         ------+-----------+-------
            0         0        1
            0         0        0
            0         0        0


         User boundary defined by layer  1


         The equation of the slicing plane is:

          0.000     * Y  +  0.000     * X  +    1.00     * Z  =   0.250E+04
```

COMMENTS

In this example, the SLIC command includes a comment, "LAYER ONE", after the specification of the locations of the points. The program ignores all data written beyond the defined fields of any command. These unused columns may be used for comments.

Computation of Flow Vectors

DISCUSSION

The display of flow-vector diagrams in two dimensions is an extremely useful technique for illustrating the results of a ground-water simulation, for checking hypotheses regarding the magnitude and direction of water movement, and for identifying the potential ground-water flow paths. The VECT command is provided to calculate flow vectors from data output by the Modular Model.

To compute flow vectors using the MMSP program, several preparatory steps are required. When preparing the data that are input to the Modular Model, it is essential to save the cell-by-cell flow terms from the block-centered flow package. This is done by specifying a non-zero value for ICBCFL in the output-control data for each time step to be used for flow-vector calculation. Additionally, a positive value for IBCFCB must be entered for the block-centered flow package, identifying the Fortran unit number for writing the cell-by-cell flow terms. The job control directives that are used to run the Modular Model and the MMSP programs must open the file unit specified by IBCFCB.

At least four MMSP commands must be performed successfully prior to the VECT command. The cell-by-cell flow terms quantifying the flow through the lower, right, and front faces of model cells must be placed onto the three-dimensional stack using the MMSP READ command. Three READ commands specifying the data-set names LOFACE, RIFACE, and FRFACE, and the desired stress period and time step should be used to get these data. A SLIC command must be used to define the plane that is the two-dimensional slice of the simulation grid for which vectors will be computed. If the slice intersects more than one layer of the simulation grid, it may be desirable to specify the thickness of model layers using the THIC command, to control the vertical distance between nodes on different layers.

Flow vectors can be viewed from one of three orientations. If one imagines the three-dimensional simulation grid as a box, the three viewing orientations correspond to looking at the box perpendicular to any two of the orthogonal axes. Thus, one may look at the box from the front, the top, or the side. The VECT command provides an option for specifying the orientation for viewing oblique cross-sections.

Figure 4 illustrates the three possible orientations and the origin of coordinates used for each one. The Modular Model uses the node on the top layer, back row, and left column for the origin of coordinates used for numbering columns, rows, and layers. The MMSP programs writes flow vectors in length units to correctly account for irregularly spaced simulation grids. The origin of coordinates used when writing flow vectors on a two-dimension plane is always in the lower-left corner of the image. Since most graphical display programs assume the origin of coordinates to be in the lower-left corner of the illustration, the MMSP program uses the edge of the bottom-most layer, the left-most column, and the last row of the Modular Model simulation grid as the origin of the coordinate system.

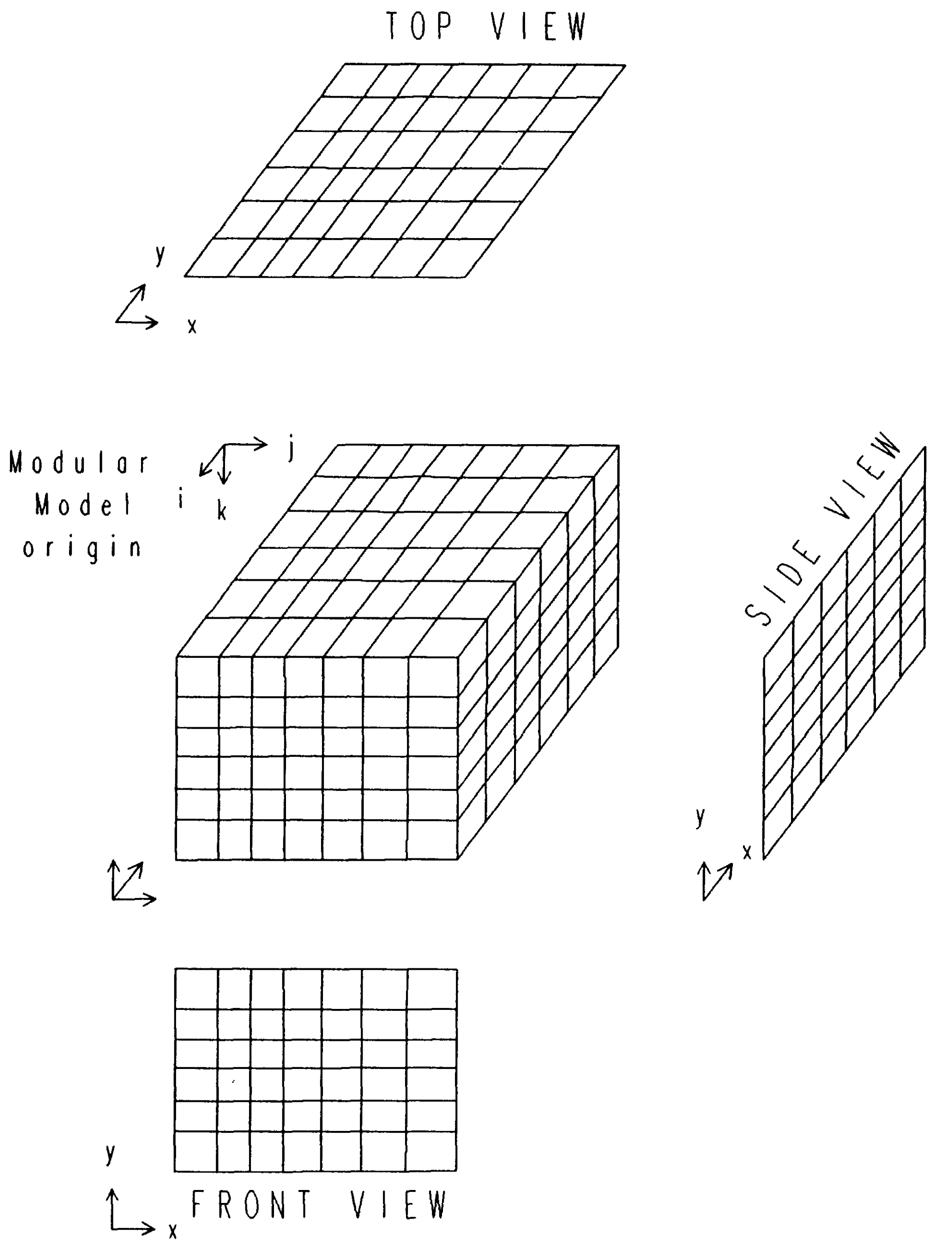


Figure 4.--Cartesian reference frames of the three viewing orientations.

An example oblique slice and two possible viewing orientations of the cross-section are shown in figure 5. When flow vectors for oblique cross-sections are created, the program uses the user-specified viewing orientation to determine which two orthogonal components of flow to use for the calculations. The equation of the slicing plane is used for selecting cells from the simulation grid. An example oblique slice and the simulation grid cells included on the slicing plane are shown in figure 6.

Irrespective of the viewing orientation and the angle of the slicing plane, the program computes the orthogonal components of flow using discharges through all six faces of each grid cell on the slicing plane. The discharge is averaged for each of the three pairs of parallel grid-cell faces (fig. 7). Zero discharge is computed and written by the Modular Model for the right-face flow in the last column, for the lower-face flow in the bottom layer, and the front-face flow in the last row of the simulation grid (McDonald and Harbaugh, 1988, p. 5-83 through 5-84). Since the MMSP program computes flow vectors by averaging the discharges through parallel grid-cell faces, the resultant flow vectors at these locations will often be non-zero. For example, if flow is simulated into the bottom layer, the MMSP program will compute a downward vertical component of flow for nodes in the bottom layer.

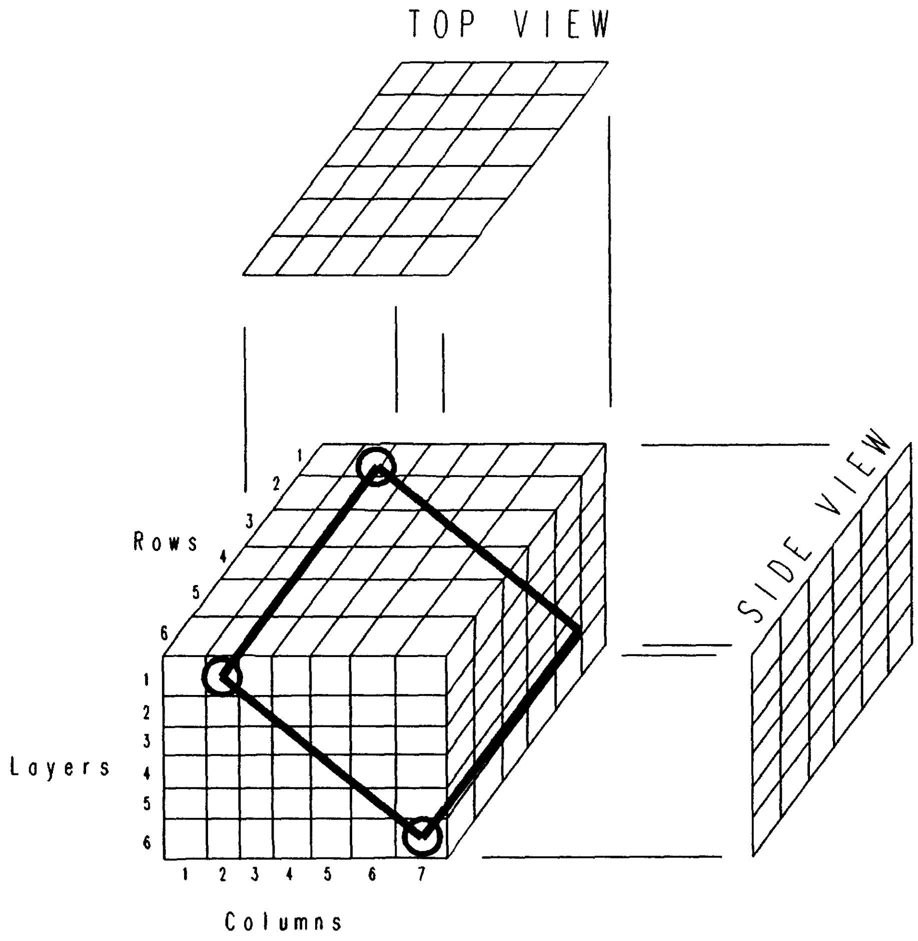
The length of the flow vectors may be scaled either arithmetically or logarithmically. When arithmetic scaling is selected, the length of each flow vector is multiplied by a user-specified constant. When logarithmic scaling is selected, the MMSP program computes the minimum power-of-ten multiplier needed to increase the minimum non-zero vector length to a value greater than one. The computation of vector lengths is made by multiplying all flow-vector lengths by the computed power-of-ten, taking the base-10 logarithm of the product, and multiplying by the absolute value of the user-specified scaling factor.

The VECT command will compute and write flow vectors for model nodes that are on the plane defined by the SLIC command. A vector is written for each node on the slicing plane. For each vector, four records are written to the output file. The first record is an identification number that is unique to each node in the model grid. The identification number is calculated by the following formula.

$$ID = J + ((I-1) + (K-1) * NROW) * NCOL$$

where ID = node identification number,
J = node column,
I = node row,
K = node layer,
NROW = number of rows in the simulation, and
NCOL = number of columns in the simulation.

The second record consists of the model node location relative to the origin of the coordinate system, in length units. The origin of the coordinate system used for calculating flow vectors is transformed from the coordinate system used by the Modular Model, as described previously.



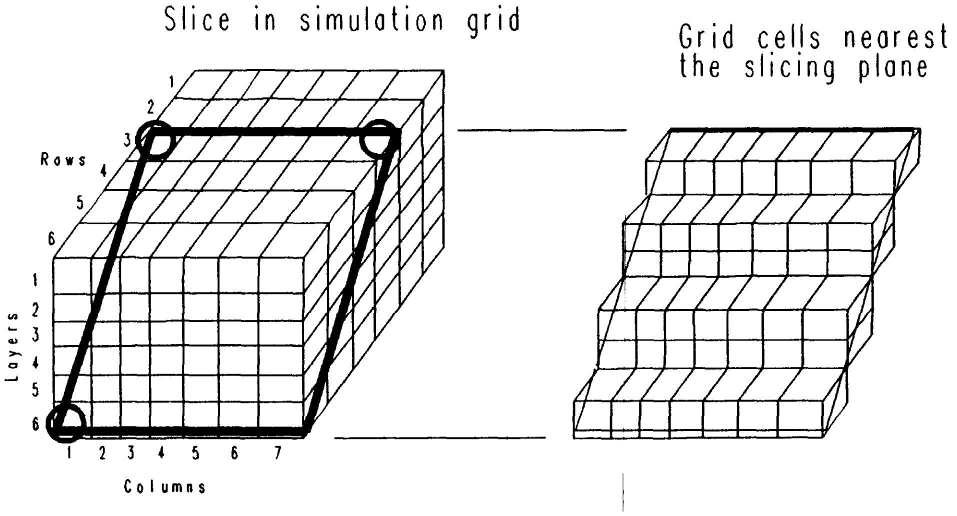
Oblique slice defined by:

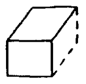
Row	Column	Layer
1	2	1
6	2	1
6	7	6

Figure 5.--Two possible viewing orientations for a sample oblique slice.

Oblique slice defined by:

Row	Column	Layer
3	1	1
3	7	1
6	1	6





Top viewing orientation shows flow through right and front cell faces.

Front viewing orientation shows flow through right and lower cell faces.

Figure 6.--Grid cells and vector components used for the calculation of flow vectors on a sample oblique slice.

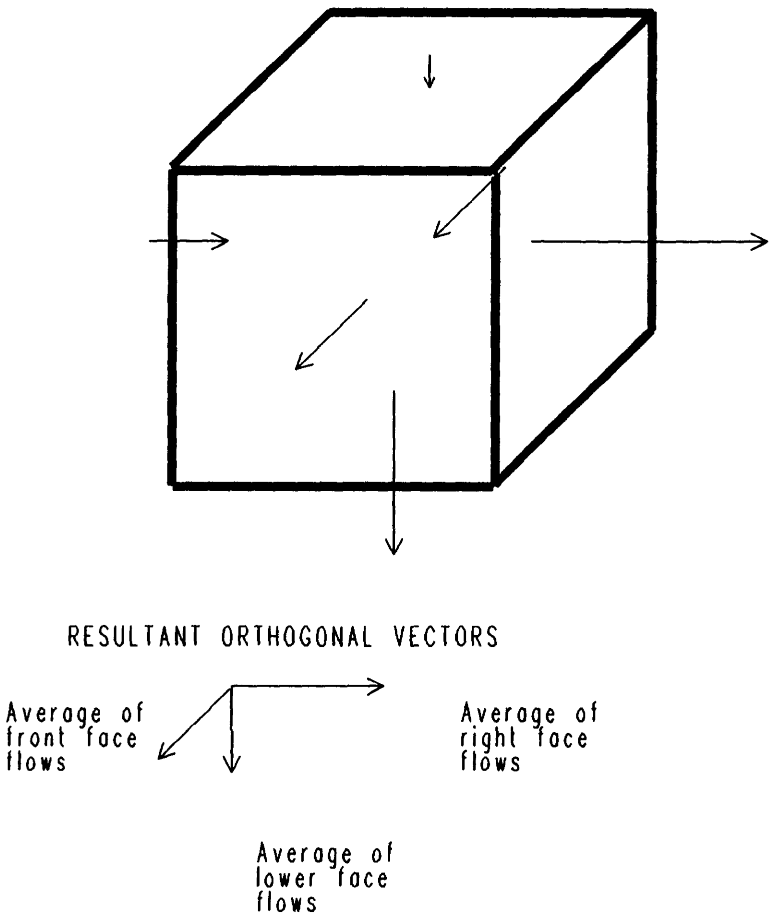

RESULTANT ORTHOGONAL VECTORS
Average of front face flows
Average of right face flows
Average of lower face flows

Figure 7.--Computation of orthogonal vector components for a grid cell.

The third record consists of the flow vector end-point location relative to the origin of the coordinate system, in length units. The length of the vector is proportional to the magnitude of the discharge in that direction.

The fourth record written for each flow vector is the three characters "END". After the fourth record of the last flow vector has been written, an additional "END" record is written.

The flow vector format described herein is compatible with the requirements for entry into the ARC/INFO geographic information system (Environmental Systems Research Institute, Inc., 1987a, p. GENERATE 1-10). A supplementary program, VECTOR.AML, for using the flow-vector data with ARC/INFO is included in attachment C. This format for flow-vector data also can be used easily with customized graphical display programs.

INPUT INSTRUCTIONS

Field:	Command	UNIT	ORIENTATION	SCALE-FACTOR
Beginning column:	1	6	9	15
Data Format:	VECT	I2	A5	F10.0

In order to compute and write flow vectors, include the necessary READ and SLIC commands in the command file, followed by the VECT command. The VECT command should specify the Fortran unit number to be used for writing flow-vector data. The Fortran unit must be opened by the job control directives used to run the MMSP program.

If the plane defined by the SLIC command is oblique, the VECT command must specify the desired orientation for viewing the plane. The choices for orientation are: "TOP", "FRONT", and "SIDE". These orientations are illustrated in figure 4. If the plane is defined along a column, row, or layer of the model grid, the orientation of the viewing plane is determined by the MMSP program, and need not be specified.

The length of the flow vectors may be scaled arithmetically or logarithmically. If no scale factor is entered with the VECT command, the default is to use no scaling (scale factor equal to one). If a positive number is entered for a scaling factor, all vector lengths are multiplied by that constant. If a negative scaling factor is entered with the VECT command, the length of the flow vectors computed by the MMSP command will be proportional to the base-10 logarithms of the flow magnitudes.

SAMPLE INPUT

Input Command File

```
EXAMPLE OF VECT COMMAND

USED TO COMPUTE FLOW VECTORS PROJECTED ONTO A PLANE

****      1    1    2    2    3    3    4    4    5    5    6    6    7
****5     0    5    0    5    0    5    0    5    0    5    0    5    0
READ RIFACE        01 01
READ FRFACE        01 01
READ LOFACE        01 01
SLIC 000 000 001                            LAYER ONE
VECT 56       1500.
```

SAMPLE OUTPUT

Output Print File

```
SAMPLE----3 LAYERS, 15 ROWS, 15 COLUMNS, STEADY STATE, CONSTANT HEADS COLUMN 1, LAYERS 1 AND 2, RECHARGE, WELLS AND DRAINS
EXAMPLE OF VECT COMMAND
USED TO COMPUTE FLOW VECTORS PROJECTED ONTO A PLANE



Processing:  VECT 56       1500.



         FLOW VECTORS COMPUTED FROM CELL BY CELL FLOW TERMS
                                     ON THE PLANE:   0.000     * Y +  0.000     * X +   1.00     * Z =  0.250E+04
                                     WRITTEN TO UNIT:  56
                                     VECTOR SCALE FACTOR =  0.150E+04
                                     VECTORS SHOWN IN ARITHMETIC UNITS
                                     HORIZONTAL AXIS REPRESENTS COLUMNS
                                      VERTICAL  AXIS REPRESENTS ROWS
                                     THE RIGHT  EDGE OF THE GRID IS AT   75000.000
                                     THE FRONT  EDGE OF THE GRID IS AT   75000.000
                                     THE BOTTOM EDGE OF THE GRID IS AT   15000.000
                                       MINIMUM VECTOR LENGTH  =  0.254
                                       MAXIMUM VECTOR LENGTH  =   4.61

                                    NUMBER OF VECTORS WRITTEN =    225
                                       MINIMUM SCALED LENGTH  =   381.
                                       MAXIMUM SCALED LENGTH  =  0.691E+04
```

Output Disk File

```
         1
           2500.0000          72500.0000
          -3543.4893          72500.0000
END
         2
           7500.0000          72500.0000
           1847.9209          72369.5000
END
         3
          12500.0000          72500.0000
           7566.8486          72237.7812
END
```

.... data for flow vectors 4 through 222 omitted ....

```
       223
          62500.0000           2500.0000
          61911.3281           2904.8550
END
       224
          67500.0000           2500.0000
          67051.8594           2950.9443
END
       225
          72500.0000           2500.0000
          72299.9062           2797.9414
END
END
```

COMMENTS

Fortran unit 56 was opened with job control directives prior to beginning the MMSP program. Three READ commands and one SLIC command were performed by the program prior to VECT command. Only the output of the VECT command is shown here. The processing of these data with the supplementary program, VECTOR.AML, is shown in chapter 4, Displaying Flow Vectors.

CHAPTER 4. SAMPLE APPLICATIONS OF THE PROGRAM

General Comments

In chapters 2 and 3 of this report, numerous potential applications of the MMSP program are suggested. In chapter 3, example input and output are presented for each MMSP command. These examples consistently refer to the sample simulation documented in appendix D of the Modular Model report with modifications as described at the beginning of chapter 3 of this report.

In this chapter, some other sample simulations are discussed to show how the MMSP program can be used to investigate ground-water flow modeling problems. The applications are not an exhaustive listing of the capabilities of the MMSP program for analyzing ground-water simulations, rather these examples show some of the problems that can be analyzed, and further demonstrate how the program can be used.

Checking Data Input to the Model

The Modular Model does not verify, in any substantial way, data that are input for a simulation. It is the modeler's responsibility to ensure that the data adequately describe the problem and that the various data elements are logically consistent. The MMSP program can be used as part of the verification process. Examples of using the MMSP program to check data input to the Modular Model follow.

Comparing Starting Head to Land Surface Altitude

Starting heads for the simulation (STRT) are specified in the basic package of the Modular Model. It may be appropriate to check starting head values using several different tests. In a water-table aquifer, values of head are greater than or equal to the altitude of the land surface only at seeps, springs, streams, and swamps. In a confined aquifer, it is possible for values of head to exceed the altitude of the land surface at many model nodes. Therefore, it may be inappropriate to perform this check on a confined aquifer layer.

To determine where starting heads exceed the altitude of the land surface, a data array that represents the altitude of the land surface at model nodes on the top layer must first be read by the MMSP program. If the values for head are measured from a different datum plane than the values for altitude, one data array must be adjusted so that both arrays are measured from the same datum plane. The values of starting head then can be compared to the land-surface altitude for each layer in the simulation.

The first step in performing this test is to prepare a data array that represents the land-surface altitude. For this example, there are three layers, ten columns, and ten rows. Layer one is strictly unconfined (LAYCON(1)=0), and layers two and three are fully convertible between confined and unconfined (LAYCON(2)=3 and LAYCON(3)=3).

Altitude data in this example have been picked from a topographic map and entered into a file using the format needed for the two-dimensional real array reader, U2DREL. The array control record specifies that the data array is to be read on Fortran unit 41, according to the format '(8F10.0)', and is not to be printed.

```
      41         1          (8F10.0)        -1    LOCAT,CNSTNT,FMTIN,IPRN
    2113.     2120.     2132.     2135.     2147.     2159.     2174.     2188.
    2190.     2195.
    2110.     2112.     2125.     2128.     2137.     2148.     2166.     2174.
    2183.     2190.
    2109.     2113.     2121.     2125.     2133.     2146.     2156.     2169.
    2177.     2186.
    2105.     2103.     2115.     2122.     2129.     2134.     2146.     2158.
    2167.     2178.
    2100.     2100.     2111.     2119.     2122.     2130.     2139.     2149.
    2157.     2168.
    2094.     2098.     2105.     2112.     2117.     2123.     2130.     2139.
    2148.     2157.
    2103.     2108.     2113.     2120.     2126.     2132.     2137.     2145.
    2151.     2158.
    2109.     2114.     2118.     2126.     2135.     2141.     2147.     2154.
    2159.     2162.
    2111.     2118.     2124.     2132.     2140.     2149.     2156.     2163.
    2168.     2175.
    2114.     2120.     2128.     2137.     2146.     2157.     2168.     2179.
    2180.     2190.
```

For this example, the starting heads were measured from a datum plane 2,000 feet above sea level. Therefore, the MATH command is used to adjust the starting head data array to the same datum plane as the altitude data array. The following commands would be given to the MMSP program to enter the land-surface data array, adjust the starting head data array, and compare the two arrays.

```
****    1    1    2    2    3    3                     COLUMN
****5   0    5    0    5    0    5                   INDICATORS
READ LSD    41 2 R       LAND SURFACE
MATH STRT      +  2000.     HEAD+D HEAD @ SEA LEVEL
TITL CHECKING IF HEADS GREATER THAN LAND-SURFACE ALTITUDE
COMP HEAD+D  1  > LSD
```

The READ command retrieves the land-surface data from Fortran unit 41, as a two-dimensional, real array, and provides the textual description "LAND SURFACE". The MATH command creates a new data array by adding 2,000 to every value for starting head. Since no layer numbers are specified, the new data array is three-dimensional. The new data array is named HEAD+D, and has the textual description "HEAD @ SEA LEVEL". The next command provides a title for the next page of printed output. Finally, the COMP command compares the top layer of adjusted starting heads to the land-surface altitude data array.

Comparing the Aquifer Top and Bottom Elevations

A similar logical test can be performed with the aquifer top (TOP) and aquifer bottom (BOT) arrays. To test if the elevation of the top is less than the aquifer bottom of any confined layer, only one MMSP command is necessary. The MMSP program will perform this test only where the layer-type code (LAYCON) is 3, indicating a layer that is fully convertible between confined and unconfined. Therefore, it is not necessary to identify which layers should be used for performing the comparison test. The following commands can be used to perform this test.

```
****    1    1    2    2    3    3                              COLUMN
****5   0    5    0    5    0    5                            INDICATORS
TITL CHECKING IF AQUIFER TOP IS LESS THAN AQUIFER BOTTOM ALTITUDE
COMP TOP           < BOT
```

Comparing Aquifer Top and Bottom to Land Surface Altitude

Similarly, each of the data arrays TOP and BOT may be tested against the land-surface altitude to determine if any values exceed the land-surface altitude.

```
****    1    1    2    2    3    3                              COLUMN
****5   0    5    0    5    0    5                            INDICATORS
READ LSD     41 2 R           LAND SURFACE
MATH LSD          -   2000. 01 LSD-2K LSD - 2000 FEET
TITL CHECKING OF AQUIFER TOP GREATER THAN LAND-SURFACE ALTITUDE
COMP TOP       1   > LSD-2K
COMP TOP       2   > LSD-2K
COMP TOP       3   > LSD-2K
TITL CHECKING IF AQUIFER BOTTOM GREATER THAN LAND-SURFACE ALTITUDE
COMP BOT       1   > LSD-2K
COMP BOT       2   > LSD-2K
COMP BOT       3   > LSD-2K
```

As before, the land-surface altitude data are read from Fortran unit 41 and given a data-set name of LSD. In this example, the LSD data array is adjusted to a datum plane 2,000 feet above sea level by subtracting 2,000 from each element. The new data array is named LSD-2K. A layer number is needed for the data array because a three-dimensional array is prepared when the data-set name is a number. The LSD data array is two-dimensional, and when a two-dimensional data array is used with a three-dimensional data array in a COMP or MATH command, a layer number should be supplied with the three-dimensional data-set name.

For each of the three layers, the elevation of the layer top and bottom is compared to the LSD-2K data. Six COMP commands are used because the LSD-2K data array is two-dimensional. When TOP or BOT is unavailable for any layer, (the layer-type array LAYCON indicates that these data are not present), then the corresponding MMSP COMP command causes an error message to be printed, the comparison is not performed, and the MMSP program processes the next command. In this example, layer 1 is unconfined (LAYCON(1)=1), therefore the first COMP command produces an error message because there is no TOP data array for layer 1.

Finding Pumpage or Recharge at Inactive and Constant Head Nodes

The Modular Model will allow a modeler to specify recharge or pumpage data for inactive or constant-head nodes. However, the results of the simulation will not be affected by recharge or pumpage at these locations. Therefore, such specifications may be an error in the data that are input to the Modular Model. It is simple is check for these conditions using the MMSP program. A modeler can determine at which model nodes these conditions occur, and compute the recharge flux applied to nodes where the water does not enter the simulation.

```
****     1    1    2    2    3                           COLUMN
****5    0    5    0    5    0                         INDICATORS
TITL LISTING OF INACTIVE OR CONSTANT HEAD NODES RECEIVING RECHARGE
COMP RECH        > 0.0         -1
TITL LISTING OF INACTIVE OR CONSTANT HEAD NODES WITH PUMPAGE
COMP WELL        > 0.0         -1
TITL RECHARGE FLUX APPLIED TO INACTIVE AND CONSTANT HEAD NODES
STAT RECHF           -1
```

In this example, a listing of nodes is written where recharge and pumpage are applied to inactive and constant-head nodes. The last command prepares a statistical summary of recharge flux at inactive and constant head nodes. The recharge flux (volume/time) applied to inactive and constant-head nodes is printed in the statistical summary under the heading **SUM OF VALUES**. All of these commands use a model-boundary mask to restrict the analyses to inactive and constant-head nodes.

Checking the Range of the Storage Coefficient Values

The storage coefficient of an aquifer is the volume of water released from a unit area of the aquifer by a unit change in head. The storage coefficient for a transient simulation of a confined aquifer is specified for the Modular Model in the input to the block-centered flow package. Typically, values for the storage coefficient of a confined aquifer are in the range between 0.00005 and .005 (Todd, 1964, p. 13-4).

In an unconfined aquifer, the storage coefficient is equal to the specific yield, which is the volume of water yielded from an saturated medium by gravity divided by the total volume of the medium. Specific yield of an unconfined aquifer is normally significantly greater than the storage coefficient of a confined aquifer, and can be as large as 0.35 (Johnson, 1967, p. D70). The MMSP program can be used to verify that the values specified for the storage coefficient and specific yield arrays in an isotropic, homogeneous aquifer are within a reasonable range.

```
****     1    1    2    2                                COLUMN
****5    0    5    0    5                              INDICATORS
TITL CHECKING SPECIFIC YIELD, LAYER ONE IS UNCONFINED
COMP SC1      1  > 0.35    1
TITL CHECKING STORAGE COEFFICIENT, LAYERS 2-3 ARE CONFINED
COMP SC1      2  > 0.005   2
COMP SC1      3  > 0.005   3
COMP SC1      2  < 0.5E-4  2
COMP SC1      3  < 0.5E-4  3
```

Analyzing the Results of Simulations

The MMSP program is especially useful for analyzing the results of ground-water simulations made using the Modular Model. In the past, many ground-water modelers have written computer programs to assist in the interpretation of a particular simulation. The MMSP program is a general-purpose computer program that eliminates the need for many of these programs. Furthermore, the MMSP program provides the capabilities for detailed analyses of ground-water simulations that previously were difficult to accomplish.

Summarizing Drawdown

The Modular Model computes and stores drawdown by subtracting heads at any specified time step from starting heads. The MMSP program can compute the drawdown between any pair of time steps when heads were saved. Therefore, when using the MMSP program, it is unnecessary to have the Modular Model compute and save drawdowns.

Commonly, a ground-water model is used to simulate the effects of various ground-water management strategies for an aquifer system. By computing the drawdown from the start of the simulation to the final time step, analysis of the effects of a given strategy on head in the aquifer is easy.

In the following example applications of the MMSP program, a transient simulation that consists of three stress periods and five time-steps in each stress period is analyzed. There are 15 columns, 15 rows, and 3 layers in the model grid. At the second stress period, a new field well is placed into operation in the vicinity of another well field. Heads and cell-by-cell flow terms are written to a file at the end of each time step.

To summarize the results of the simulation, a statistical analysis is prepared for the drawdown during the simulation period. Computed heads are read for the final time step of the third stress period from the unformatted file written by the Modular Model. A data array of drawdown is computed by subtracting the computed heads from the starting heads. The new data array is three-dimensional, and is named "CHANGE". Summary statistics and a frequency analysis for active model nodes are written to the print file.

```
****     1    1    2    2    3                                 COLUMN
****5    0    5    0    5    0                               INDICATORS
READ HEAD          3    5
MATH STRT       - HEAD       CHANGE CHANGE IN HEAD
TITL STATISTICAL SUMMARY OF DRAWDOWN AT ACTIVE NODES
STAT CHANGE        1
HIST CHANGE        1
```

Summarizing Drawdown Within a Specific Area

The modeler wants to evaluate the effects that a new well field will have on water levels in the vicinity of an existing well field. The existing well field is within rows 4 through 13, columns 6 through 14, in layer 2 of the model grid. To accomplish this task, a user-boundary array is prepared and entered into a file using the format for the two-dimensional integer array reader, U2DINT. The first and third array control records identify all nodes in layers 1 and 3 as outside the user boundary, and suppresses printing of these data. The second array control record specifies that the data array is to be read on Fortran unit 56, according to the format '(15I3)', and is not to be printed.

```
          0        -1(15I3)                 -3   UBOUND-1
         56         1(15I3)                 -3   UBOUND-2
 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
 -1 -1 -1 -1 -1  1  1  1  1  1  1  1  1  1 -1
 -1 -1 -1 -1 -1  1  1  1  1  1  1  1  1  1 -1
 -1 -1 -1 -1 -1  1  1  1  1  1  1  1  1  1 -1
 -1 -1 -1 -1 -1  1  1  1  1  1  1  1  1  1 -1
 -1 -1 -1 -1 -1  1  1  1  1  1  1  1  1  1 -1
 -1 -1 -1 -1 -1  1  1  1  1  1  1  1  1  1 -1
 -1 -1 -1 -1 -1  1  1  1  1  1  1  1  1  1 -1
 -1 -1 -1 -1 -1  1  1  1  1  1  1  1  1  1 -1
 -1 -1 -1 -1 -1  1  1  1  1  1  1  1  1  1 -1
 -1 -1 -1 -1 -1  1  1  1  1  1  1  1  1  1 -1
 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
          0        -1(15I3)                 -3   UBOUND-1
```

As in the previous example, a statistical analysis is prepared for drawdown during the simulation period. Computed heads are read for the final time step of the third stress period, and a data array of drawdown is prepared by subtracting the computed heads from the starting heads. The user-boundary array is read by the program and used to limit the data included in the summary statistics and the frequency analysis.

```
****     1    1    2    2    3                          COLUMN
****5    0    5    0    5    0                        INDICATORS
READ HEAD           3    5
MATH STRT         - HEAD       CHANGE CHANGE IN HEAD
READ UBOUND 56 3 I
TITL SUMMARY OF DRAWDOWN AT ACTIVE NODES IN EXISTING WELL FIELD
STAT CHANGE         1 1
HIST CHANGE         1 1
```

Both of the previous example analyses could have been accomplished in a single execution of the MMSP program. To combine both analyses, the TITL, STAT, and HIST commands from the prior example could be inserted after the MATH command in the second example.

Comparing the Results of Several Simulations

Suppose the ground-water modeler also must compare the effects of several pumpage rates (50, 100, and 150 cubic feet per second). To perform this task, the modeler must make three simulations of the ground-water system, setting the pumpage rates specified for the Modular Model in the well package before each simulation. After the simulation of each of the first two pumpage rates, the MMSP program is used as described in the previous example. However, an additional command is added to the command file to write the final computed heads to a file, so that the data array can be used as an auxiliary data array in a later execution of the MMSP program. After simulation of pumpage of 50 cubic feet per second, the command file would contain the following commands.

```
****     1    1    2    2    3                             COLUMN
****5    0    5    0    5    0                           INDICATORS
READ HEAD          3    5
MATH STRT        - HEAD        CHANGE CHANGE IN HEAD
READ UBOUND 56 3 I
TITL DRAWDOWN AT ACTIVE NODES IN EXISTING WELL FIELD: 50 CFS
STAT CHANGE        1 1
HIST CHANGE        1 1
TITL COMPUTED HEADS FOR 50 CFS PUMPAGE
WRIT HEAD         31    0
```

The last command causes the MMSP program to write a formatted file of computed heads on Fortran unit 31, using the print-format code zero (Fortran format '(10G11.4)', refer to table 9). Similarly, for the simulation of pumpage of 100 cubic feet per second, the corresponding command file would contain the following commands.

```
****     1    1    2    2    3                             COLUMN
****5    0    5    0    5    0                           INDICATORS
READ HEAD          3    5
MATH STRT        - HEAD        CHANGE CHANGE IN HEAD
READ UBOUND 56 3 I
TITL DRAWDOWN AT ACTIVE NODES IN EXISTING WELL FIELD: 100 CFS
STAT CHANGE        1 1
HIST CHANGE        1 1
TITL COMPUTED HEADS FOR 100 CFS PUMPAGE
WRIT HEAD         32    0
```

After simulating pumpage of 150 cubic feet per second, the results of all three simulations can be read and manipulated. The following example command file causes the same analysis of computed heads as previously demonstrated. In addition, the computed heads from the simulations of other pumpage rates are accessed with the READ command, placed on the three-dimensional stack, and given the data-set names: WL50 and WL100. The subsequent MATH command is given in order to move the HEAD data array to the bottom of the stack, so that it is not lost after the following MATH commands. These MATH commands compute the differences between computed heads for each of the pumpage rates. The difference data arrays are named CHG1-2, CHG1-3, and CHG2-3. Statistical summaries are then printed for each of the three arrays for all active nodes, as well as active nodes within the existing well field.

```
****      1    1    2    2    3                                 COLUMN
****5     0    5    0    5    0                                 INDICATORS
****
**** Analyzing the simulation of pumpage at 150 cfs
TITL SUMMARY OF DRAWDOWN AT ACTIVE NODES IN EXISTING WELL FIELD
READ HEAD             3    5
MATH STRT          - HEAD        CHANGE DRAWDOWN: 150 CFS
READ UBOUND 56 3 I
TITL DRAWDOWN, 150-CFS PUMPAGE AT ACTIVE NODES
STAT CHANGE          1
HIST CHANGE          1
TITL DRAWDOWN, 150-CFS PUMPAGE AT ACTIVE NODES, EXISTING WELL FIELD
STAT CHANGE          1 1
HIST CHANGE          1 1
****
**** Reading head data from the simulations using other pumpages
**** These data were saved using the WRIT command in previous runs
READ WL50    31 3 R         WATER LEVEL:  50 CFS
READ WL100   32 3 R         WATER LEVEL: 100 CFS
****
**** Computing the head-difference arrays between pumpage scenarios
**** 3-D stack:  HEAD, CHANGE, WL50, WL100
MATH HEAD          + 0.0         HEAD    150 CFS SURFACE
MATH WL50          - WL100       CHG1-2   50-100 CFS SURFACE
MATH WL50          - HEAD        CHG1-3   50-150 CFS SURFACE
MATH WL100         - HEAD        CHG2-3  100-150 CFS SURFACE
****
**** Preparing statistics for the head-difference arrays
TITL HEAD DIFFERENCE BETWEEN PUMPAGE AT 50 AND 100 CFS
STAT CHG1-2          1
STAT CHG1-2          1 1
TITL HEAD DIFFERENCE BETWEEN PUMPAGE AT 50 AND 150 CFS
STAT CHG1-3          1
STAT CHG1-3          1 1
TITL HEAD DIFFERENCE BETWEEN PUMPAGE AT 100 AND 150 CFS
STAT CHG2-3          1
STAT CHG2-3          1 1
```

Computing Change of Volume of Water in Storage

Another common task performed by modelers after performing a transient simulation of a ground-water system is to compute the volume of water in storage. In a water-table aquifer, the volume of water in storage is computed by multiplying the saturated thickness by the specific yield and the area of the aquifer. For a block-centered finite-difference grid the volume of water stored in a unconfined layer is computed by the following formula.

$$\sum_{i}\sum_{j} \mathrm{DELR}_j \bullet \mathrm{DELC}_i \bullet S_{ij} \bullet \left(\mathrm{HEAD}_{ij} - \mathrm{BOT}_{ij}\right)$$

where	i	= column number,
	j	= row number,
	DELR	= width of cells along columns,
	DELC	= width of cells along rows,
	S	= specific yield,
	HEAD	= elevation of the water surface in an unconfined unit, and
	BOT	= elevation of the bottom of an unconfined unit.

The volume of water stored in a Modular Model simulation where the first layer of the grid is unconfined could be computed by the MMSP program using the following commands.

```
TITL COMPUTING VOLUME OF WATER IN STORAGE IN LAYER 1, SP 3, TS 5
REBO  3  5
READ HEAD           3 5
MATH HEAD    01   - BOT     01 SATHIK SATURATED THICKNESS
MATH SATHIK       * AREA    01 VOLUME VOLUME LAYER 1
MATH VOLUME       * SC1     01 STORE1 STORAGE LAYER 1
STAT STORE1       000200
```

The REBO command is given to update the model-boundary array (IBOUND) by inactivating any model nodes that may have gone dry during the simulation. Head for the fifth time step of stress period three is read onto the three-dimensional stack. Saturated thickness is computed using the MATH command to subtract the bottom elevation from the head in layer one. The resultant data array is named SATHIK and is placed on the two-dimensional stack. The volume of the saturated zone is computed by the next MATH command, by multiplying saturated thickness by the area of cells, and placing the results on the two-dimensional stack. The third MATH command computes the volume of water in storage in the layer, by multiplying the volume of the saturated zone with the primary storage factor.

Finally, a STAT command is used to calculate descriptive statistics for the volume of water in storage. The mask field of the STAT command restricts the computation to active and constant head nodes. It is essential to mask the inactive nodes, because the Modular Model sets the head at all inactivated nodes to 10^{30}.

If the saturated thickness or the area of cells is large, it may be desirable to scale the volume data array by using the MATH command to divide by some convenient power of 10. The STAT command may cause the MMSP program to fail to compute geometric and harmonic means if the size of a data value exceeds the limit that can be stored. This size limitation is determined by the scheme used to represent floating point data on a computer system, and this information is made available to the MMSP program in the inserted code block named TINY.INS (attachment A).

Computing the Effects of Well Pumpage on Stream Discharge

Cell-by-cell flow terms from the Modular Model can be analyzed to answer questions about the results of a simulation. For example, suppose an intermittent stream were included in the simulation of the previous problem, where the modeler is analyzing the potential effects of a new well field. The stream makes a minimal, if any, contribution to the ground-water system, therefore it is simulated using the drain package of the Modular Model. The question to be answered is: "What decrease in contribution of ground water to stream discharge will occur if the proposed well field is constructed?"

The first step in solving this problem is to perform a steady-state simulation of conditions prior to the development of the well field. The result is used to determine the base flow of the stream caused by contribution from the ground-water system. To use these data in a later analysis by the MMSP program, the data array of cell-by-cell flow terms from the drain package is read and reformatted, as was done in the example application where the effects of different pumpage rates on computed head were compared.

```
TITL SAVING CBC FLOW TERMS FROM THE DRAIN PACKAGE FOR LATER USE
READ CBCDRN          1 1
WRIT CBCDRN            33
```

Next, a transient simulation is performed, and the MMSP program is used to analyze the different contributions to streamflow between the two simulations.

```
TITL STREAMFLOW CONTRIBUTION AT STRESS PERIOD 3, TIME STEP 5
READ CBCDRN          3 5
READ DRNSS   33 3 R        CBCDRN FOR STEADY-STATE
MATH CBCDRN       - DRNSS       DIFF    CHANGE IN BASE-FLOW
STAT DRNSS       010000
STAT CBCDRN      010000
STAT DIFF        010000
```

The first READ command reads the cell-by-cell flow terms from the drain package created during the transient simulation, and puts the data array on the three-dimensional stack. The second READ command reads, from the steady-state simulation, the cell-by-cell flow terms that were written by the MMSP program after the first model run. The MATH command places a new data array named DIFF on the three-dimensional stack, by subtracting one cell-by-cell flow term data array from the other.

Because flow from a cell is computed as a negative value by the Modular Model, a negative value in the DIFF data array indicates a decrease in streamflow contribution at the corresponding node in the model. Three STAT commands are used to summarize the two cell-by-cell flow-term data arrays and the difference data array. A zero mask is used to exclude nodes where there is no flow from the aquifer to the stream. If any model nodes stopped contributing to streamflow during the transient simulation, an increase in the number of values excluded from the STAT command analysis by the zero mask will be seen.

Graphing the Results of Simulations

The MMSP program does not produce graphical displays. However, the MMSP HEAD and VECT commands produce data files that can be used easily with graphical display programs.

Graphing Computed and Observed Heads

Commonly, hydrographs are prepared to show changes in water level with time. The HEAD command can be used to extract simulated values of water level for selected model nodes and write the values to a formatted file. This file may be used subsequently with a graphical display program to produce hydrographs.

For this example, consider a transient simulation with 16 stress periods, each with a 1-day duration. There is one time step for each stress period, and heads are saved for stress periods: 1, 2, 4, 8, and 16. Time in the simulation is measured in seconds. For the period covering the simulation, water levels at an observation well were measured daily in the field. The location of the observation well corresponds to the node in the simulation grid at row 17, column 17, and layer 10.

After completion of the Modular Model transient simulation, the following command is used with the MMSP program.

```
HEAD 44  17  17  10
```

This command caused computed head values for the node corresponding to the observation well to be written to a file open on Fortran unit 44. Job control directives have opened a file named HEADOUT on Fortran unit 44. The file of computed heads is shown below.

```
17   17   10    1    1         86400.          86400.      98.233842
17   17   10    2    1         86400.         172800.      90.144520
17   17   10    3    1  -0.123456E+20   -0.123456E+20  -0.123456E+20
17   17   10    4    1         86400.         345600.      85.145238
17   17   10    5    1  -0.123456E+20   -0.123456E+20  -0.123456E+20
17   17   10    6    1  -0.123456E+20   -0.123456E+20  -0.123456E+20
17   17   10    7    1  -0.123456E+20   -0.123456E+20  -0.123456E+20
17   17   10    8    1         86400.         691200.      82.732234
17   17   10    9    1  -0.123456E+20   -0.123456E+20  -0.123456E+20
17   17   10   10    1  -0.123456E+20   -0.123456E+20  -0.123456E+20
17   17   10   11    1  -0.123456E+20   -0.123456E+20  -0.123456E+20
17   17   10   12    1  -0.123456E+20   -0.123456E+20  -0.123456E+20
17   17   10   13    1  -0.123456E+20   -0.123456E+20  -0.123456E+20
17   17   10   14    1  -0.123456E+20   -0.123456E+20  -0.123456E+20
17   17   10   15    1  -0.123456E+20   -0.123456E+20  -0.123456E+20
17   17   10   16    1         86400.        1382400.      79.852342
```

The HEAD command produced 16 records in the file, one for each time step in the simulation. Each record contains the row, column, and layer of the model node; the stress period and time step number; the elapsed time since the beginning of the stress period and since the beginning of the simulation; and the computed head. The Fortran format for each record is: '(5I5,3G15.6)'. When heads were not saved for a time step, the elapsed time and computed head fields contain the first missing-value indicator.

The observed water level data have been written to a formatted file that is shown below. The name of this file is GW.SUBF.

```
35430209610210119820311   3.74
35430209610210119820312  11.57
35430209610210119820313  14.50
35430209610210119820314  15.15
35430209610210119820315  15.22
35430209610210119820316  15.31
35430209610210119820317  15.38
35430209610210119820318  15.48
35430209610210119820319  15.59
35430209610210119820320  15.71
35430209610210119820321  15.80
35430209610210119820322  15.89
35430209610210119820323  16.11
35430209610210119820324  16.41
35430209610210119820325  16.88
35430209610210119820326  16.97
```

The first 15 columns of data on each record are a site identification number. The next 8 columns are the calender date that the water level was measured. The remaining columns are the depth to the water surface measured from the land surface, in feet.

The data elements of the computed and observed water levels are not expressed in equivalent units. (1) The location of the head data from the Modular Model are expressed in coordinates of the simulation grid cells. The location of the observed water-level data is given by the siteidentification number. (2) The elapsed-time data from the Modular Model are in seconds since the beginning of the simulation. The times of the observed water levels are given as calendar dates. (3) The computed head data from the Modular Model represent the water level above a datum plane 100 feet below the land-surface datum. The observed water-level data are measured as the distance below the land-surface datum.

A computer program may be used to adjust these data elements and produce a hydrograph. For a quick analysis of observed and computed heads, use of a statistical package that has some data manipulation and graphical display capabilities is an alternative. One such statistical package is P-STAT (Buhler, 1986). A variety of statistical and graphical display packages may be used, and several different approaches to solving the problem may be used with each package. Shown below is an example of one approach using the P-STAT package.

```
BUILD CH, FIXED, FILE HEADOUT;
          VARS ROW     1-5
               COL     6-10
               TOTIM  41-55 (M1 '  -0.123456E+20')
               CLEVEL 56-70 (M1 '  -0.123456E+20')$
MODIFY CH (IF ROW = 17 AND COL = 17, GENERATE SITEID:C15 = '354302096102101')
          (GENERATE DAY = TOTIM / 86400) (KEEP DAY, CLEVEL, SITEID), OUT MCH$
PURGE CH$
BUILD OH, FIXED, FILE GW.SUBF;
          VARS SITEID:C15
               DAY    22-23
               DEPTH  24-30 $
MODIFY OH (SET DAY = DAY - 10) (GENERATE OLEVEL = 100 - DEPTH)
          (KEEP DAY, OLEVEL, SITEID),   OUT MOH$
PURGE OH$
JOIN MCH MOH, OUT BH, FILL, NO CHECKS$
PLOT BH, BY SITEID;
          PLOT CLEVEL * DAY;
          OVERLAY OLEVEL * DAY$
```

Explanation of the details of P-STAT commands is not feasible in this report. For an in-depth analysis of this example, readers should consult the P-STAT User's Manual (Buhler, 1986). The following discussion presents an overview of the P-STAT commands used in this example. Each P-STAT command is free-format, and continues for one or more lines. The end of each command is identified by a dollar-sign ($).

The first command, BUILD, reads the data from the file created by the HEAD command into a P-STAT data set named CH. The next command, MODIFY, creates a new data set named MCH from CH, by adding the variables for day number (DAY), and site-identification number (SITEID). The CH data set is then discarded using the PURGE command.

Next, the observed water-level data are read into a P-STAT data set named OH using another BUILD command. The OH data are altered with another MODIFY command to create a new P-STAT data set, named MOH, with the new variable OLEVEL for water level measured from the same datum plane as the computed head data (CLEVEL), and the recomputed variable DAY for day number. The PURGE command is used to discard the OH data set. The two modified data sets, MCH and MOH, are merged using the JOIN command to create the data set BH. The merged data set BH contains the variables SITEID, DAY, OLEVEL, and CLEVEL. Finally the plot command is used to produce the following line-printer plots.

```
BY:  SITEID = 354302096102101

                 Plot of CLEVEL by DAY (Legend: *=1, 2=2, ..., $=10+)

        100 +
            |*
         95 +
 C          |
 L       90 +    *
 E          |
 V       85 +            *
 E          |                            *
 L       80 +                                                            *
            |
         75 +
            |
         70 +
            +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
             1   2   3   4   5   6   7   8   9   10  11  12  13  14  15  16
                                            DAY
```

```
BY:  SITEID = 354302096102101

              Plot of CLEVEL by DAY (Legend: *=1, 2=2, ..., $=10+)
                 Overlay OLEVEL by DAY (Legend: A =1 or more)

      100 +
          |*
       95 +
 C        |
 L     90 +   *
 E        |   A
 V     85 +       A   *   A   A   A   A   A   A   A   A   A
 E        |                           *                       A   A   A
 L     80 +                                                           *
          |
       75 +
          |
       70 +
          +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
          1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16
                                       DAY
```

As an additional benefit, a statistical package may be used to analyze simulated data using correlation and regression analysis. Continuing the previous example, the following P-STAT commands perform some exploratory analysis of the observed and computed heads.

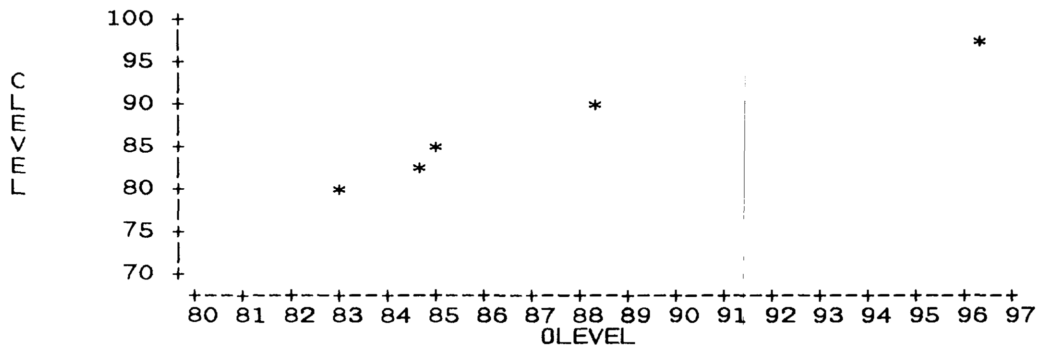
```
CORRELATE BH (KEEP OLEVEL CLEVEL), OUT CORR.BH$
LIST CORR.BH$
REGRESSION BH (KEEP OLEVEL CLEVEL);
           SUMMARY, OUT REGR.BH,
           INDEPENDENT OLEVEL,
           DEPENDENT   CLEVEL$
PLOT REGR.BH;
           PLOT     CLEVEL * OLEVEL;
           PLOT PRE.CLEVEL * CLEVEL;
           PLOT RES.CLEVEL * CLEVEL$
```

These commands provide a cursory examination of the relation between the observed and simulated heads. A complete analysis is beyond the scope of this report. This example shows some of the capabilities provided by using the MMSP program to write computed head data to a file and use these data in a statistical package.

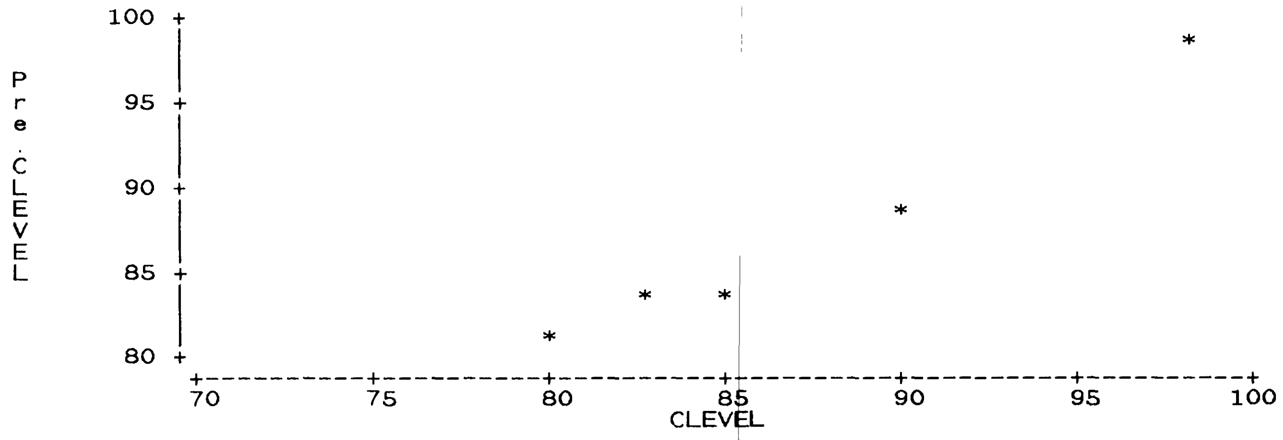
The CORRELATE command prepares a correlation matrix between observed and computed heads. The REGRESSION command computes a simple linear regression equation for predicting computed head from observed head. The REGRESSION command creates a new P-STAT data set named REGR.BH that contains the original variables CLEVEL and OLEVEL, as well as heads computed from the regression equation PRE.CLEVEL and the residuals RES.CLEVEL (RES.CLEVEL = CLEVEL - PRE.CLEVEL). The results of appending these example P-STAT commands to the previous P-STAT commands follow.

```
Correlate completed.
16 cases were read.
5 is the smallest good N for any pair of variables.
The variables are OLEVEL and CLEVEL .

. . . . . . . . . . . . . . .

                            row
    OLEVEL        CLEVEL    label

  1.000000     0.981631   OLEVEL
  0.981631     1.000000   CLEVEL
11 cases were dropped because of missing data.

All variables are now in.

. . . . . . . . . . . . . . .

********************************************************************************
File is BH.

Final summary of Regression on dependent variable CLEVEL:

Multiple R                        0.9816
Multiple R squared                0.9636
Adjusted R squared                0.9515
Std. Error of Est.                1.5909
Constant                        -29.1327


Analysis of                       Sum of            Mean          F       Prob.
  Variance         DF             Squares           Square        Ratio   Level
Regression          1             200.992           200.992       79.414  0.003
Residual            3               7.593             2.531

Adj. Total          4             208.585


  S   Num    Mult    Mult   Change           Variable          B,      Stand
  T  vars     R        R    in RSq            entered          raw     error
  E   now             Sq.                    (* shows         coef-      of
  P    in                                    deleted)        ficient      B

  1     1  0.982   0.964   0.964              OLEVEL          1.3310   0.1494


  S           Variable     BETA,      Final    F when     Simple   Partial
  T            entered     Stand.       F      entered     cor.    cor. in
  E           (* shows     coef-       to         or       with     final
  P           deleted)    ficient    delete    deleted     dep.      step

  1            OLEVEL      0.9816    79.414    79.414     0.9816    0.9816

Regression Equation:

CLEVEL (pred)  =  1.33101 OLEVEL  +  -29.1327

********************************************************************************
 File REGR.BH has 16 rows and 4 variables.
```

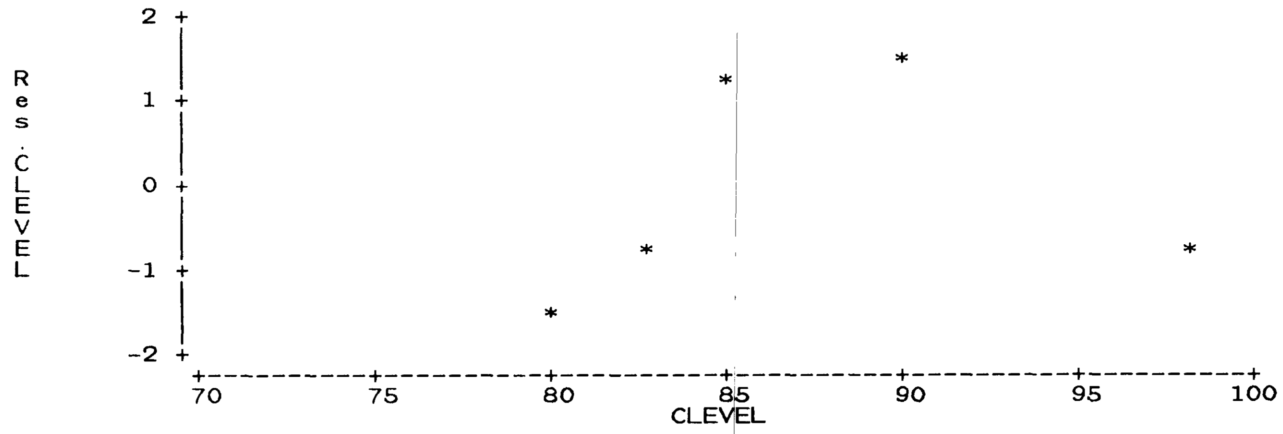
Plot of CLEVEL by OLEVEL (Legend: *=1, 2=2, ..., $=10+)



Plot of Pre.CLEVEL by CLEVEL (Legend: *=1, 2=2, ..., $=10+)



Plot of Res.CLEVEL by CLEVEL (Legend: *=1, 2=2, ..., $=10+)



Displaying Flow Vectors

The MMSP VECT command writes data that can be used easily with the ARC/INFO geographic information system (GIS). In this GIS, spatial data for points, lines, and polygons are grouped into logical sets referred to as "coverages". The VECT command produces a data file that is easily used to create a "line coverage", which can be displayed and analyzed with other coverages such as rivers, wells, or water-level contours.

The data produced by the VECT command can be used by a custom user-supplied program. Information about the contents and formatting of the data is given in chapter 3. In attachment C, VECTOR.AML and ARCNUM.F77, programs for creating an ARC/INFO line coverage from the output of the VECT command, are presented. VECTOR is an ARC/INFO Macro-Language (AML) program that runs ARCNUM, a Fortran 77 program.

The AML program should be run from within the ARC/INFO software package. After invoking ARC/INFO, the AML program is run by typing the following command:

```
&RUN VECTOR VECT_FILE COVER_NAME NROW NCOL
```

```
where  VECT_FILE  = name of the file created by the MMSP VECT command,
       COVER_NAME = name of the new coverage to be created,
          NROW    = number of rows in the model simulation grid, and
          NCOL    = number of columns in the model simulation grid.
```

The VECTOR program will first perform some tests on the data that are supplied on the command line. These tests include checking that all four variables were entered, checking if the VECT_FILE exists, and checking that the COVER_NAME does not exist. If any of these tests fail, the VECTOR program prints a message, and stops. Otherwise, the program creates a new line coverage and adds three new variables to the coverage. These variables are referred to as "items" or "attributes" in ARC/INFO terminology. The three new variables are used for identifying the row, column, and layer of each flow vector. Finally, a Fortran program, ARCNUM, is run to define the row, column, and layer coordinates for each vector using the vector identification number that was written by the MMSP VECT command.

To install the VECTOR package on a computer system, the VECTOR.AML program may need to be modified to correctly identify how the ARCNUM program can be run. The line of the VECTOR.AML program shown below determines the method for running the program, and the location of the ARCNUM program on the computer system.

```
&S OS.COMMAND := RESUME ARCNUM.RUN
```

This line may need to be modified for each computer system. As shown, the OS.COMMAND AML variable is set to run a executable load module on a Prime computer system (RESUME) that is named ARCNUM.RUN in the current disk-storage area. When the ARCNUM program is compiled and linked, the linking process must include a reference to the ARC/INFO binary object code.

To illustrate the use of the VECT command and the VECTOR package, the modified sample simulation from appendix D of the Modular Model report was used (McDonald and Harbaugh, 1988). The example input for the VECT command in chapter 3 shows the commands needed to write flow-vector data to Fortran unit 56, for nodes on layer one of the model simulation grid. For this example, a disk file named VECT.OUT was opened on Fortran unit 56 by job control directives prior to running the MMSP program. After running the MMSP program to produce the flow-vector data, the VECTOR program was run from within ARC/INFO with the following command to create a new ARC/INFO coverage named LAYER1.

```
&RUN VECTOR VECT.OUT LAYER1 15 15
```

When the VECTOR program runs, it produces the following output.

```
[VECTOR Revision 1.0]

(C) 1987 Environmental Systems Research Institute, Inc.
    All Rights Reserved Worldwide

[GENERATE Version 4.0 (24 June 1987)]

Creating Lines with coordinates loaded from VECT.OUT

Externalling BND and TIC...

 Building lines...
Creating attribute file for LAYER1
Adding ROW to LAYER1.AAT to produce LAYER1.AAT.
Adding COLUMN to LAYER1.AAT to produce LAYER1.AAT.
Adding LAYER to LAYER1.AAT to produce LAYER1.AAT.
Submitting command RESUME ARCNUM.RUN
MOD>RBD>INFO
LAYER1
15
15
Number of nodes defined =  225
**** STOP
```

The creation of a flow-vector coverage is complete. The lines in the coverage represent the direction and the scaled magnitude of the discharge at each node. Each flow vector is identified by three attributes for the ROW, COLUMN, and LAYER number of its location in the simulation grid. However, the spatial locations of the nodes in the coverage are represented in the units of the simulation (for example: feet), with the origin of coordinates in the lower-left corner of the plane that slices the simulation grid.

It is possible to use the ARC/INFO ARCPLOT (Environmental Systems Research Institute, Inc., 1987b) sub-system to display the flow vectors. This is performed easily using the ARCS and the ARCARROWS commands of the ARCPLOT sub-system. To display the flow vector coverage created in this example, the graphical display device should be defined using the ARCPLOT DISPLAY command, then the following commands should be given to produce the flow vectors shown in figure 8.

```
MAPEXTENT LAYER1
ARCS LAYER1
ARCARROWS LAYER1
```

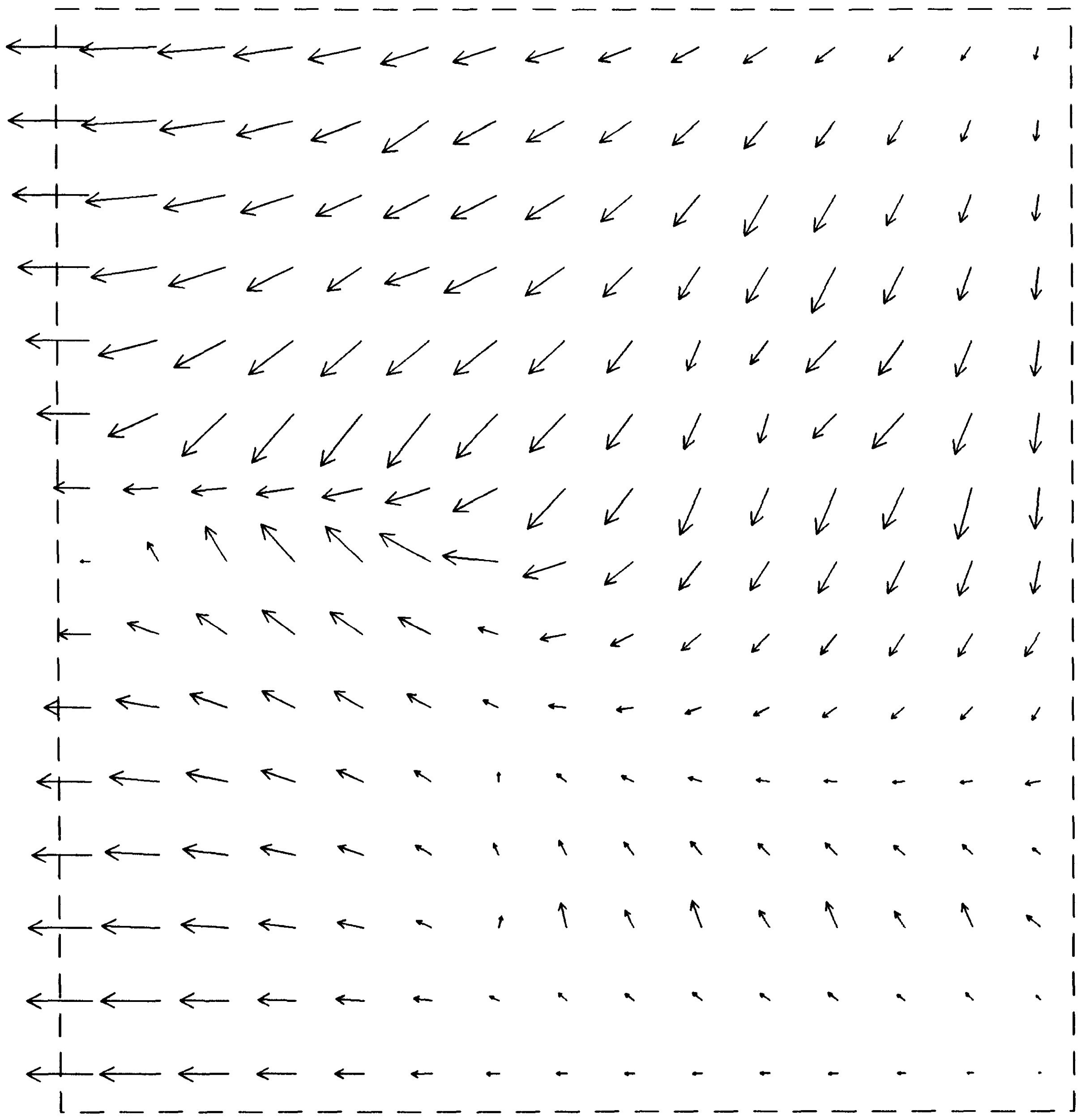


Figure 8.--Sample flow vectors computed by the modular model statistical processor.

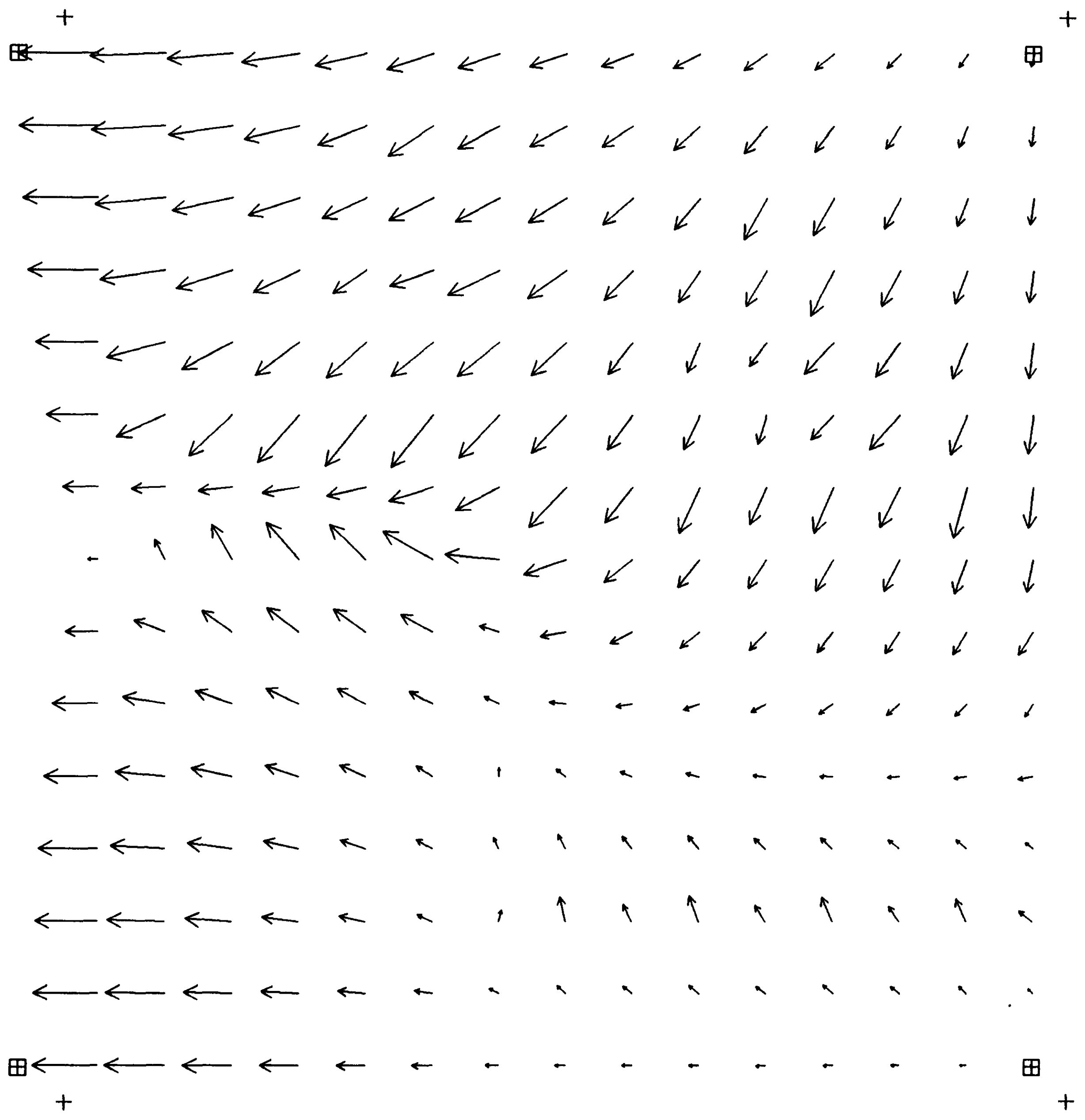
Flow-vector coverages that are derived from a horizontal slice for a specific layer of the simulation grid can be displayed along with other coverages. However, until the coordinate system of the flow-vector coverage is transformed to a system equivalent to the one used with other ancillary coverages, the coverages cannot be shown on the same display.

The ARC/INFO GIS maintains a set of registration points that are used to transform a coverage from one coordinate system to another. These registration points are referred to as "tics". To create a coverage of flow vectors that can be displayed with other geographic data, the values for the tics must be set to known coordinates in the simulation grid, using the same units that were used in the Modular Model simulation. These same tics also must be identified in the coordinate system used for the registration of the ancillary geographic data. Finally, all of the coordinates in the flow-vector coverage from the VECTOR program are converted to equivalent coordinates using a relation defined by the coordinates of the tics in the two references frames.

The simplest points to use for tics in the flow vector coverage are the four corners of the simulation grid. The first step in transforming the flow-vector coverage is to determine the areal extent of the simulation grid. The total horizontal distance along a row is the sum of the cell widths along rows (DELR). Similarly, the total horizontal distance along a column is the sum of the cell widths along columns (DELC). In this example, the total distance in both dimensions is 75,000 (15 x 5,000) since the simulation grid is square. The sums of the cell widths are printed in the output of the VECT command.

Initially, after running the VECTOR program, the four tics will be defined with their coordinates set equal to the minimum and maximum values of coordinates for any of the flow vectors (fig. 9). Since flow vectors may extent beyond the simulation grid (flow toward the edge of the modeled area), these values for registration points are of little utility. However, the values for the tics are easily modified using the INFO sub-system of the ARC/INFO GIS. A example session showing how to reset tics to the areal limits of the simulation follows.

```
ENTER COMMAND >SELECT LAYER1.TIC
      4 RECORD(S) SELECTED

ENTER COMMAND >UPDATE
RECNO?>1
                  1
IDTIC             =     1
XTIC              =  72,500.000
YTIC              =   2,500.000
?>XTIC = 75000.
?>YTIC = 0.
?>
RECNO?>2
                  2
IDTIC             =     4
XTIC              =  72,500.000
YTIC              =  72,500.000
?>IDTIC = 2
?>XTIC = 75000.
?>YTIC = 75000.
?>
```



⊞ Registration points after VECTOR.AML
+ Registration points at simulation grid corners

Figure 9.--Locations of registration points prior to conversion to a map coordinate system.

```
RECNO?>3
                       3
IDTIC                  =     2
XTIC                   =   -3,543.489
YTIC                   =    2,500.000
?>IDTIC = 3
?>XTIC = 0.
?>YTIC = 0.
?>
RECNO?>4
                       4
IDTIC                  =     3
XTIC                   =   -3,543.489
YTIC                   =   72,500.000
?>IDTIC = 4
?>XTIC = 0.
?>YTIC = 75000.
?>
RECNO?>

ENTER COMMAND >QUIT STOP
```

In this example session, the tic file for flow vectors is accessed with the SELECT command. The tic file-name is the same as the name of the coverage with a suffix of ".TIC". The UPDATE command is used to change the coordinate values of the tics. The user is prompted for the record number by UPDATE, and after the record number is entered, the variables are displayed. The coordinates are updated for each tic, in counterclockwise order, starting in the lower-right corner. The order is arbitrary, but must be consistent with a subsequent step. A value is updated by typing the name of the variable (such as XTIC), an equals sign, and the desired value.

After entering the XTIC and YTIC values for each tic, a carriage return is given in response to the "?>" prompt. After all tics have been updated, a carriage return is given in response to the "RECNO?>" prompt. Finally, the commands QUIT STOP are given to the INFO sub-system. Additional information about these commands can be found in the INFO documentation (Henco, 1983).

To transform the flow-vector coverage, it is necessary to have detailed information about the desired coordinate system. For example, the coordinate system might be the Universal Transverse Mercator (UTM) map projection (Synder, 1983). Values for the tics can be interpolated by locating the tics on a map. Alternatively, if the map does not show control points for the desired coordinate system, the coordinates of the tics can be interpolated from the coordinate system used on the map and projected to the desired coordinate system using the ARC/INFO PROJECT command.

To use the latter method, pick coordinates of the tics from the map, enter the coordinates into a file, and use the PROJECT command to create a file of tic coordinates in the chosen coordinate system. There are many possible ways to use the PROJECT command (Environmental Systems Research Institute, Inc., 1987a, p. PROJECT-1 through PROJECT-27). A example session of the PROJECT command follows, where a file of latitude and longitude data (named "LLS") is projected to a file of UTM coordinates (named "UTMS"). The values for latitude-longitude in this example were picked arbitrarily from a map for a square with 75,000-foot sides.

```
Arc: PROJECT FILE LLS UTMS

Please define the input and output map projections.
Use INPUT to define the input projection, OUTPUT
to define the output projection, and END to finish.

Project: INPUT
Project: PROJECTION GEOGRAPHIC
Project: UNITS DMS
Project: QUADRANT NW
Project: FORMAT '(3I2,1X,I3,2I2)'
Project: FLIP YES
Project: PARAMETERS
Project: OUTPUT
Project: PROJECTION UTM
Project: UNITS METERS
Project: ZONE 14
Project: PARAMETERS
Project: END
```

The ARC/INFO command: CREATE is used to generate an empty coverage. The new coverage to be created will be the transformed flow-vector coverage. The CREATE command requires only the name of the coverage to be created, for example:

```
CREATE VECTORS.L1
```

After creating the empty coverage, the INFO sub-system is used to enter the values for the tics in the chosen coordinate system that have either been interpolated from a map or computed by the PROJECT command. The INFO ADD command is used below to enter the values generated by the PROJECT command, from the file named UTMS. It is important to enter the coordinate values in the same order as the corresponding values were entered in the previous session with the INFO sub-system.

```
ENTER COMMAND >SELECT VECTORS.L1.TIC
     0 RECORD(S) SELECTED

ENTER COMMAND >ADD
    1
IDTIC>1
XTIC>523085.9353
YTIC>4039237.1064
    2
IDTIC>2
XTIC>523023.9398
YTIC>4062346.1676
    3
IDTIC>3
XTIC>500000.0001
YTIC>4062315.2085
    4
IDTIC>4
XTIC>500000.0001
YTIC>4039206.2156
    5
IDTIC>
      4 RECORD(S) ADDED

 ENTER COMMAND >QUIT STOP
```

The final step in the process of converting the flow-vector coverage to map coordinates is to transform the coverage using the UTM registration points. The ARC/INFO TRANSFORM command is used to accomplish this conversion, as shown in the following example.

```
Arc: TRANSFORM LAYER1 VECTORS.L1
Transforming coordinates for coverage LAYER1

 Transformation scale: 0.308
            RMS error:  50.3

 tic id      input x      input y      output x     output y       delta x      delta y
 ------  -----------  -----------  -----------  -----------  -----------  -----------
      1  0.750E+05    0.000        0.523E+06    0.404E+07    -15.6         -0.500
      2  0.750E+05    0.750E+05    0.523E+06    0.406E+07     15.4         -0.500
      3  0.000        0.750E+05    0.500E+06    0.406E+07    -15.6         -0.500
      4  0.000        0.000        0.500E+06    0.404E+07     15.4          0.000
```

The coverage named VECTORS.L1 now contains flow vectors in UTM coordinates, and can be displayed with other coverages of the same geographic area that are stored in the UTM coordinate system.

If another coverage of flow vectors in model units is to be converted to UTM coordinates, this process can be shortened. The abbreviated process for converting another coverage is outlined below.

(1) Reset the values for the registration points using the INFO UPDATE command in the unconverted coverage to the coordinates of the corners of the simulation grid.

(2) Create a new coverage using the ARC/INFO CREATE command, but, in this instance, use an optional feature that causes the new coverage to contain the registration points from an existing coverage. When a second coverage name is specified with the CREATE command, the registration points in the second coverage are copied into the newly created coverage. For example, suppose that the coverage LAYER2 has been created using the VECTOR program, from data prepared by an MMSP VECT command, and values for the tics have been reset as described in step one. To create a new coverage for the transformed flow vectors using the UTM registration points in an existing coverage, the CREATE command shown below would be used.

```
CREATE VECTORS.L2 VECTORS.L1
```

(3) Use the ARC/INFO TRANSFORM command to convert the coverage in model coordinates to a coverage in UTM coordinates.

CHAPTER 5. PROGRAM DOCUMENTATION

Discussion

This report does not contain a detailed discussion of the internal construction of the MMSP program. Variable listings, flow charts, and descriptions of the subroutines may be published in a separate report at a later date. However, this chapter presents an overview of the program construction.

The main program is similar to the main program of the Modular Model. A series of subroutines are called to allocate space in the "Z" array for the various time- and space-dependent data. The names of each of these subroutines end with the letters "AL", as in the Modular Model. Subsequently, a series of subroutines are called to read and prepare data that are input to the Modular Model simulation. Again, as in the Modular Model, the names of each of these subroutines ends with the letters "RP".

The final function of the main program is to iteratively call the command-processing subroutine, CTL1CM, to process the MMSP command file. The main program transfers control to the default-command processing subroutine named DFT1CM, if end-of-file is reached immediately after reading the title lines from the command file.

A computer programmer can change the default commands. This is done by changing the variables NDFTCM and DFTCMD in subroutine DFT1CM. After making the desired changes the subroutine should be compiled and linked.

The command-processing subroutine consists largely of a single Fortran 77 BLOCK-IF construction. The operations pertinent to performing a particular MMSP command are contained within a branch of the BLOCK-IF structure.

The remaining subroutines and functions can be classified into two groups: those that pertain to performing a particular MMSP command, and those that perform an operation common to two or more MMSP commands.

The six blocks of programming statments inserted into the program (table 1 and attachment A) specify machine-specific constants and problem-specific constants. The size of the "Z-array" is modified by altering the value of the variable LENZ. The size of the two-dimen'sional and three-dimen'sional stacks is modified by changing the value of the variable ISTKSZ. Machine-specific constants are stored in the variables TINY, BIG, SMALL, and HUGE. The maximum number of layers is set by the dimension specification of the variable LAYCON. The default missing-value indicators are assigned to the three-element array MISVAL. A programmer can modify these assignments to accomodate different computer systems or different modeling requirements.

Either of two versions of the proprietary IMSL subroutine library can be used with the program. The MMSP code contains two subroutine-call statements for every use of an IMSL subroutine in the program. The MMSP subroutines that contain subroutine-call statements for IMSL subroutines are listed in table 2. Only one of these subroutine-call statements should be used for each pair of subroutine-call statements in the MMSP program. The other subroutine-call statement should be disabled by placing the letter "C" in column one of the statement. The choice of which call statement should be enabled depends on which version of the IMSL library is used on the computer system.

Informative and Error Messages

Messages from the MMSP program are written to the print file. Messages can be grouped into two classes: informative messages and error messages. Informative messages provide information about how the program is processing data and are intended to be self-explanatory. Informative messages are not discussed in this chapter. Error messages provide an indication that something may be wrong with the specifications given to the program.

Error messages are sub-divided into three groups. The first group is called "run-fatal" error messages because when these errors occur, the MMSP programs stops, and no processing can be performed until the error is found and corrected. The second group of error messages is called "command-fatal" error messages because when these messages occur, the MMSP program stops processing the current command. Processing resumes with the next command in the MMSP command file. Command-fatal errors may cause subsequent commands in the MMSP command file to fail, if these subsequent commands require the results of a previous command that has generated a command-fatal error. The third group is called "warning" error messages because the MMSP program will perform the command that generated the warning error message. However, the user should be aware that the results of the command may be incorrect, therefore a warning error message is printed.

The remainder of this chapter lists the error messages that may be printed by the MMSP program. The error messages are listed alphabetically, within each error-message group. The text of each message is shown, followed by the name of the subroutine or subroutines that cause the message to be printed, and one or more potential corrective actions that the user may wish to perform before running the MMSP program again with the same command file.

Run-Fatal Errors

Two run-fatal errors are generic. These errors occur when the MMSP program attempts and fails to read data. The subroutines listed in table 10 read data from the Fortran units specified for input and output for the Modular Model and MMSP commands. Each of these subroutines may issue an error message indicating that the expected data type was not found. The text of the generic error message is shown on the next page of this report.

Table 10.--*Subroutines of the Modular Model Statistical Processor that can print generic input error and end-of-file messages*

Group of data read by subroutine	Subroutine name
Basic package	BASDDF
	BASDAL
	BASDRP
	BASDST
Block-centered flow package	BCFDAL
	BCFDRP
River package	RIVDAL
	RIVDRP
Recharge package	RCHDAL
	RCHDRP
Well package	WELDAL
	WELDRP
Drain package	DRNDAL
	DRNDRP
Evapotranspiration package	EVTDAL
	EVTDRP
General-head boundary package	GHBDAL
	GHBDRP
Strongly implicit procedure package	SIPDAL
	SIPDRP
Slice-successive overrelaxation package	SORDAL
	SORDRP
Output control package	UI01FG
Control file	CTL1DF
Modular Model unformatted output	U3DREL
	ULYREL

```
CCCCCC:  Error reading from unit:  NN

 where   CCCCCC is the name of the subroutine
               that generated the message, and
            NN is the Fortran unit number.
```

Each of the subroutines listed in table 10 may print an error message if it attempts to read data on a Fortran unit and no more data are present. The text of the generic end-of-file error message is shown below.

```
CCCCCC:  Unexpected E-O-F on unit:  NN
```

When either of these two generic error messages are printed, the user should review the job control directives, and the unit numbers specified for the Modular Model and the MMSP program. If no mistake is found, the data provided on the specified unit number should be reviewed and edited to correct the error. The remainder of this chapter lists messages in a common format.

```
TEXT:    At row __ column __ __ is an illegal layer index
SOURCE:  RCHDRP
ACTION:  Correct IRCH value in recharge package input.
```

```
TEXT:    Aquifer type 1 is only allowed in top layer
SOURCE:  BCFDAL
ACTION:  Correct LAYCON value in block-centered flow package input.
```

```
TEXT:    DYNAMIC STORAGE EXCEEDED
         __ ELEMENTS ALLOCATED
         __ ELEMENTS AVAILABLE
SOURCE:  CTL1CK
ACTION:  Increase Z array size in ZARRAY.COMMON.INS.
```

```
TEXT:    Illegal recharge option: __...stopping
SOURCE:  RCHDAL
ACTION:  Correct NRCHOP value in recharge package input.
```

```
TEXT:    IRCH must be read for 1st stress period
SOURCE:  RCHDRP
ACTION:  Add IRCH for first stress period in recharge package input.
```

```
TEXT:    Number of layers specified: __ greater than 80...
         stopping due to insufficient space
SOURCE:  BCFDAL
ACTION:  Increase size of LAYCON array in FLWCOM.INS
         or correct NLAY value in basic package input.
```

```
TEXT:    NWELLS __ is greater than MXWELL __
SOURCE:  WELDRP
ACTION:  Increase MXWELL or decrease NWELLS in well package input.
```

```
---------------------------------------------------------------------------

TEXT:    Preceding WELL is outside model limits.
SOURCE:  WELDRP
ACTION:  Correct coordinates of well in well package input.

---------------------------------------------------------------------------

TEXT:    WELLS must be read for the first stress period
SOURCE:  WELDRP
ACTION:  Add wells for first stress period in well package input.

---------------------------------------------------------------------------
```

Command-Fatal Errors

```
---------------------------------------------------------------------------

TEXT:    All __ values in the array are the same: __
SOURCE:  HIS1EX
ACTION:  None required.  Histogram is uninformative.  May need to
         correct data-set name.

---------------------------------------------------------------------------

TEXT:    Array-size error occurred reading __ on unit __
             SIZE READ        SIZE EXPECTED
          NROW NCOL NLAY    NROW NCOL NLAY

          ____ ____ ____    ____ ____ ____
SOURCE:  ULYREL or U3DREL
ACTION:  Correct unit number in READ command, job control directives,
         or array control record.

---------------------------------------------------------------------------

TEXT:    Attempted to READ __ cut points; minimum = 3
SOURCE:  REA1CL
ACTION:  Correct NUCLAS specification in READ command,
         increase number of cut points, or use computed classes.

---------------------------------------------------------------------------

TEXT:    Attempted to READ __ cut points; maximum = 20
SOURCE:  REA1CL
ACTION:  Correct NUCLAS specification in READ command,
         decrease number of cut points, or use computed classes.

---------------------------------------------------------------------------
```

```
--------------------------------------------------------------------

TEXT:     Comparison limit is non-numeric
SOURCE:   CTL1CM
ACTION:   Correct the limit field on the comparison command.

--------------------------------------------------------------------

TEXT:     Data set __ layer __ is too large
          Maximum number of layers = __
SOURCE:   ULC1LY
ACTION:   Correct the layer number on the corresponding command.

--------------------------------------------------------------------

TEXT:     Data set: __ not available.  IUNIT from basic package for
          data set = __
SOURCE:   ULC1SP
ACTION:   Correct job control directive for basic package input, or
          correct data-set name for the corresponding command.

--------------------------------------------------------------------

TEXT:     Data-set type for __ is unknown: __
SOURCE:   REA1EX
ACTION:   Correct number of dimensions or array type on READ command.

--------------------------------------------------------------------

TEXT:     Data set __ was requested, but not found
SOURCE:   ULC1DS
ACTION:   Correct data-set name on corresponding command, reorder commands,
          or add MATH/READ command.

--------------------------------------------------------------------

TEXT:     Error occurred reading __ on unit __
             FOUND:  stress period __ time step __
          EXPECTED:  stress period __ time step __
SOURCE:   U3DREL or ULYREL
ACTION:   Correct model-output controls and re-run model, or
          correct data-set name or unit number on READ command.

--------------------------------------------------------------------

TEXT:     File unit specified for __ not an integer:  __
SOURCE:   CTL1CM
ACTION:   Correct file unit number on corresponding  WRIT, READ, HEAD,
          or VECT command.

--------------------------------------------------------------------
```

TEXT: Graphic output unit or vector factor is non-numeric
SOURCE: CTL1CM
ACTION: Correct file unit or vector scaling factor on VECT command.

TEXT: HEAD and DRAWDOWN are not available with default output control
SOURCE: UI01FG
ACTION: Correct model-output controls and re-run model, or
correct data-set name on the corresponding command.

TEXT: Illegal layer specified: __ for masking
SOURCE: MAS1EX or MAS1MV
ACTION: Correct layer number on corresponding command.

TEXT: Insufficient data to compute statistics, number of
observations = __
SOURCE: STA1EX
ACTION: None required. May need to correct mask field or layer number.

TEXT: Insufficient graphic storage available
SOURCE: VEC1EX
ACTION: Increase Z array size in ZARRAY.COMMON.INS.
Look for the message near the beginning of the print file with
the following text:

Insufficient space available for vector calculation
Need __ more elements of dynamic storage.

TEXT: Invalid format code: __
SOURCE: PRT1EX or WRT1EX
ACTION: Correct format code in WRIT or PRIN command.

TEXT: __ is not available for reading for stress period __
time-step __ because save flags are not set
SOURCE: REA1EX
ACTION: Correct model-output controls and re-run model, or
correct data-set name or stress period and time step
on READ command.

TEXT: Layer specified for __ not an integer: __
SOURCE: CTL1CM
ACTION: Correct the layer number for the corresponding data array.

TEXT: Logarithmic scaling inappropriate. All data within the same
power-of-ten
SOURCE: HIS1EX
ACTION: Change the sign of the number of classes specified with the HIST
command, or use a READ command to supply user-specified cutpoints
and set the number of classes on the HIST command to zero.

TEXT: Mask specifiers for __ not all integers: __
SOURCE: CTL1CM
ACTION: Correct the mask fields on the corresponding command.

TEXT: No (BOT or TOP) layer __ exists because LAYCON(__) = __
SOURCE: ULC1LY
ACTION: Correct layer number on corresponding command.

TEXT: No (HEAD or DRAWDN) layer __ exists because IOFLG (__) = __
SOURCE: ULC1LY
ACTION: Correct model-output controls and re-run model, or
correct layer number on corresponding command.

TEXT: No model coordinates were within the slicing plane
SOURCE: SLI1EX
ACTION: Correct slicing coordinates on SLIC command.

```
--------------------------------------------------------------------------

TEXT:    Non-numeric row, column, or layer for node number __
SOURCE:  HEA1EX
ACTION:  Correct coordinates on HEAD command.

--------------------------------------------------------------------------

TEXT:    Number of histogram cut points __ is not an integer: __
SOURCE:  REA1CL
ACTION:  Correct number of cut points on READ command.

--------------------------------------------------------------------------

TEXT:    Number of histogram classes for __ not an integer:  __
SOURCE:  CTL1CM
ACTION:  Correct number of classes on HIST command.

--------------------------------------------------------------------------

TEXT:    Orientation of __ is incompatible with the slice
SOURCE:  VEC1EX
ACTION:  Correct orientation on VECT command or change the
         coordinates given with the SLIC command.

--------------------------------------------------------------------------

TEXT:    READ command for __ Stress period __ Time step __ is beyond
         the defined limit.  Number of time steps for stress period = __
SOURCE:  CTL1CM
ACTION:  Correct the time step on the READ command.

--------------------------------------------------------------------------

TEXT:    READ command for __ Stress period __ Time step __ is beyond
         the defined limit.  Number of stress periods = __
SOURCE:  CTL1CM
ACTION:  Correct the stress period on the READ command.

--------------------------------------------------------------------------

TEXT:    (Row, column or layer) number __ is less than 0 or greater than __
SOURCE:  HEA1EX
ACTION:  Correct the coordinate on the HEAD command.

--------------------------------------------------------------------------

TEXT:    Some layer thickness is non-numeric
SOURCE:  CTL1CM
ACTION:  Correct the thickness value on the THIC command.

--------------------------------------------------------------------------
```

```
TEXT:    Some slicing coordinate is non-numeric: __
SOURCE:  CTL1CM
ACTION:  Correct the coordinate on the SLIC command.
```

```
TEXT:    Specified (row, column, or layer) is outside the range of model
         coordinates
SOURCE:  SSLI1Y
ACTION:  Correct the coordinate on the SLIC command.
```

```
TEXT:    Stress period and time step not integer: __
SOURCE:  CTL1CM
ACTION:  Correct the stress period and time step on the corresponding
         READ command.
```

```
TEXT:    Unable to reset model-boundary array because computed heads
         were not saved
SOURCE:  REB1EX
ACTION:  Correct model-output controls and re-run model.
```

```
TEXT:    Unable to compare, data sets not same size
         Length of __ = __
         Length of __ = __
SOURCE:  CTL1CM
ACTION:  Add a layer number for the three-dimensional data array, and
         compare the arrays by layers, or correct data-set name.
```

```
TEXT:    Unable to compute, data sets not same size
         Length of __ = __
         Length of __ = __
SOURCE:  CTL1CM
ACTION:  Add a layer number for the three-dimensional data array, and
         compute a data array for each layer, or correct data-set name.
```

```
TEXT:    Unable to create user-boundary mask, coordinates are colinear
SOURCE:  SSLI1Q
ACTION:  Correct slicing coordinates on the SLIC command.
```

```
------------------------------------------------------------------------

TEXT:    Unable to read command line
SOURCE:  CTL1CM
ACTION:  Correct job control directives.

------------------------------------------------------------------------

TEXT:    Unit number for reading CLASS is not an integer: __
SOURCE:  REA1CL
ACTION:  Correct unit number on READ command.

------------------------------------------------------------------------

TEXT:    Unknown command: __ refer to table 4
SOURCE:  CTL1CM
ACTION:  Correct command name.  Valid commands are listed in table 4.

------------------------------------------------------------------------

TEXT:    Unknown operator: __
SOURCE:  COM1EX OR MTH1EX
ACTION:  Correct operator on COMP or MATH command.

------------------------------------------------------------------------

TEXT:    Unknown orientation specified: __
SOURCE:  VEC1EX
ACTION:  Correct orientation on VECT command.

------------------------------------------------------------------------

TEXT:    Value too large: __ deactivated nodes may not have been masked
SOURCE:  HIS1EX
ACTION:  Use the REBO command to update the model-boundary array, and use
         a model-boundary mask to mask inactive nodes on the HIST command.

------------------------------------------------------------------------

TEXT:    Vectors requested but no slice has been defined
SOURCE:  CTL1CM
ACTION:  Add a SLIC command prior to the VECT command.

------------------------------------------------------------------------

TEXT:    Vectors requested but (RIFACE, LOFACE, FRFACE) has not been read
SOURCE:  CTL1CM
ACTION:  Read RIFACE, FLFACE, and FRFACE for the desired stress period and
         time step using the READ command prior to the VECT command.

------------------------------------------------------------------------
```

Warnings

TEXT: __ nodes were set to missing because their layer(s) was not saved for stress period __, time step __
SOURCE: HEA1EX
ACTION: Correct model-output controls and re-run model.

TEXT: Computed heads were not saved for layer: __, stress period: __, time step __
SOURCE: REB1EX
ACTION: Correct model-output controls and re-run model.

TEXT: Illegal missing-value indicator #__: __
Set to default indicator: __
SOURCE: MAS1MV
ACTION: Correct missing-value indicator on the WRIT or PRIN command.

TEXT: Invalid value for model-boundary mask: __ ....Mask ignored
SOURCE: MAS1EX or MAS1MV
ACTION: Correct the model-boundary mask key for the corresponding command.

TEXT: Number of observations < 3, therefore skewness was not computed
SOURCE: SSTA1E
ACTION: None required. Some masking specification may be erroneous.

TEXT: Overflow or underflow occurred, therefore geometric and harmonic means were not computed
SOURCE: SSTA1E
ACTION: None required. May need to mask some data values, or create a new data array scaled by some convenient power of ten.

TEXT: Overflow or underflow occurred, therefore skewness was not computed
SOURCE: SSTA1E
ACTION: None required. May need to mask some data values, or
create a new data array scaled by some convenient power of ten.

TEXT: Populating layer __ with zeroes because __ has no corresponding
layer
SOURCE: MTH1EX
ACTION: If data array is HEAD or DRAWDN: correct model-output controls
and re-run model. Otherwise, no action required.

TEXT: Unable to find data set __ layer __ two-dimensional;
Retaining complete data set
SOURCE: ULC1LY
ACTION: If using the layer-number field to specify a masking layer: no
action is required. Otherwise, set layer number to zero or
correct data-set name.

TEXT: Standard deviation = 0, therefore skewness was not computed
SOURCE: SSTA1E
ACTION: None required.

TEXT: Upper and lower quartiles equal, therefore non-parametric
skewness was not computed
SOURCE: SSTA1E
ACTION: None required.

TEXT: User-boundary mask requested, but user boundary has not been
defined....Mask ignored
SOURCE: COM1EX, MAS1EX or MAS1MV
ACTION: Use the SLIC or READ commands to define a user boundary prior to
the command using the user-boundary mask, or disable the user-
boundary mask.

```
---------------------------------------------------------------------------

TEXT:    User-specified classes requested, but have not been read;
         using 20 arithmetic computed classes
SOURCE:  HIS1EX
ACTION:  Use READ command to enter user-specified cut points, or
         specify the number of computed classes desired on the HIST commanc

---------------------------------------------------------------------------

TEXT:    Zero or negative values present in matrix, therefore geometric anc
         harmonic means were not computed
SOURCE:  SSTA1E
ACTION:  None required.  May need to mask some data values.

---------------------------------------------------------------------------

TEXT:    Zero-mask not allowed during comparison command
SOURCE:  COM1EX
ACTION:  Set the zero-mask key to zero on the COMP command.

---------------------------------------------------------------------------
```

REFERENCES

American National Standards Institute, Inc., 1978, Programming Language FORTRAN: New York, American National Standards Institute.

Buhler, Shirrell, 1986, P-STAT User's manual (Version 8): Boston, Duxbury, 852 p.

Chow, V.T., 1964, Frequency analysis *in* Chow, V.T., 1964, Handbook of applied hydrology: New York, McGraw-Hill, p. 8-8.

David, H.A., 1962, Special problems in testing hypotheses *in* Sarhan, A.E., and Greenberg, B.G., Contributions to order statistics: New York, Wiley, p. 113.

Environmental Systems Research Institute, Inc., 1987a, Users guide ARC/INFO, v. 2 *of* 2: Redlands, Calif., Environmental Systems Research Institute.

----- 1987b, Users guide ARCPLOT: Redlands, Calif., Environmental Systems Research Institute.

Henco Software, Inc., 1983, INFO Reference manual, Revision 9.0: Waltham, Mass., Henco Software, Inc.

IMSL Inc., 1982, IMSL Library reference manual, Ed. 9, (4 volumes): Houston, Tex., IMSL, Inc.

----- 1987, User's manual STAT/LIBRARY FORTRAN subroutines for statistical analysis, Version 1.0: Houston, Tex., IMSL, Inc., 1232 p.

Johnson, A.I., 1967, Specific yield--Compilation of specific yields for various materials: U.S. Geological Survey Water-Supply Paper 1662-D, 74 p.

Johnson, E.C., 1983, FORTRAN 77 Reference guide (third ed.): Framingham, Mass., Prime Computer Inc.

Kernodle, J.M., and Philip, R.D., 1988, Using a geographic information system to develop a ground-water flow model: American Water Resources Association Monograph, series no. 14, pp.191-202.

McDonald, M.G., and Harbaugh, A.W., 1988, A modular three-dimensional finite-difference ground-water flow model: Techniques of Water-Resources Investigations of the U.S. Geological Survey, Book 6, chap. A1, 528 p.

Synder, J.P., 1983, Map projections used by the U.S. Geological Survey: U.S. Geological Survey Bulletin 1532 (second ed.), 313 p.

Todd, D.K., 1964, Groundwater *in* Chow, V.T., 1964, Handbook of applied hydrology: New York, McGraw-Hill, p. 13-5.

U.S. Geological Survey, 1980, U.S. Geological Survey Yearbook, Fiscal Year 1979: p. 92.

Attachment A.--Blocks of code inserted into the Modular Model Statistical Processor Program

---

```
C ZARRAY.COMMON.INS - SET SIZE OF THE Z-ARRAY AND ALLOCATE SPACE FOR IT
      PARAMETER (LENZ=500000)
      COMMON /ZARRAY/ Z(LENZ)
```

```
C STKSIZE.INS - SET SIZE OF TWO- AND THREE-DIMENSIONAL STACKS
      PARAMETER (ISTKSZ=4)
```

```
C STKDEF.INS - ALLOCATE SPACE FOR STACK-SIZE DEPENDENT VARIABLES
      INTEGER LCU2DS(ISTKSZ)
      INTEGER LCU3DS(ISTKSZ)
      INTEGER ISKSP (ISTKSZ)
      INTEGER ISKTS (ISTKSZ)
      CHARACTER U2DDSN(ISTKSZ)*6
      CHARACTER U3DDSN(ISTKSZ)*6
      CHARACTER U2DANM(ISTKSZ)*24
      CHARACTER U3DANM(ISTKSZ)*24
```

```
C TINY.INS - DEFINE MACHINE-DEPENDENT CONSTANTS
      DOUBLE PRECISION HUGE, SMALL
      PARAMETER (TINY =1.0E-36)
      PARAMETER (BIG  =1.0E+36)
      PARAMETER (SMALL=-1.0D+9700)
      PARAMETER (HUGE =+1.0D+9700)
```

```
C FLWCOM.INS - ALLOCATE SPACE FOR MAXIMUM NUMBER OF LAYERS
      COMMON /FLWCOM/ LAYCON(80)
```

```
C
C MISVAL.INS - DEFINES THE DEFAULT MISSING-VALUE INDICATORS
      REAL      MISING(3)
      CHARACTER MISVAL(3)*10
      DATA MISVAL(1) /'-123456E20'/
      DATA MISVAL(2) /'-123457E20'/
      DATA MISVAL(3) /'-123458E20'/
```

Attachment B.--Modular model statistical processor program listing.

```
      PROGRAM MMSP
C
C U.S. Geological Survey Program MMSP is used for performing statistical
C  analysis of simulations made using the modular, three-dimensional
C  finite-difference, ground-water flow model by McDonald and Harbaugh.
C  Use of this program is described in U.S. Geological Water-Resources
C  Investigations Report 89-4159, by Jonathon C. Scott.  This program is
C  written in the Fortran 77.  The program was last modified and run on
C  a Prime 9955-II minicomputer running revision 21 of the PRIMOS operating
C  system on June 12, 1989.
C
C Although this computer software has been used by the U.S. Geological Survey,
C  no warranty, expressed or implied, is made by the USGS as to the accuracy
C  and functioning of the program and related program material nor shall the
C  fact of distribution constitute any such warranty, and no responsibility
C  is assumed by the USGS in connection therewith.
C
C ======================================================================
C     MAIN PROGRAM FOR MODULAR MODEL STATISTICAL PROCESSOR (MMSP)
C     SPECIFICATIONS:
C ======================================================================
$INSERT ZARRAY.COMMON.INS
$INSERT STKSIZE.INS
$INSERT STKDEF.INS
      DIMENSION IUNIT(24),CLASES(20),IPKGSP(12),PLANE(4)
      CHARACTER TITLE(4)*128, LINE*80
      LOGICAL EXCMD, EOF, UBIN
      INTEGER OUNIT,LCWDS(2)
C
C1 -- ASSIGN BASIC PACKAGE UNIT, OUTPUT FILE UNIT, AND CONTROL FILE UNIT
      INBAS = 5
      OUNIT = 6
      INCTL = 7
C
C2 -- INITIALIZE THE SIZE OF THE SIMULATION & MMSP VARIABLES
      CALL BASDDF (TITLE,NCOL,NROW,NLAY,NODES,INBAS,OUNIT,IUNIT,NPER)
      CALL CTL1DF (INCTL,OUNIT,TITLE,UBIN,PLANE,EOF,IUNIT,NUCLAS,
     1  IPKGSP,NCMDRD,NCMDEX,NOPRT)
C
C3 -- ALLOCATE SPACE FOR THE MMSP & THE MODEL PACKAGE DATA ARRAYS
      CALL CTL1AL (ISUMX,ISUMZ,NCOL,NROW,NLAY,NPER,LCNTS,LCUBOU,LCU2DS,
    1 LCU3DS,LCWDS,U2DDSN,U2DANM,U3DDSN,U3DANM,ISKSP,ISKTS,LCWELL,
    2 LCIRCH,LCRECH,LCDELL)
      IF (INBAS .GT. 0) CALL BASDAL
    1 (ISUMX,ISUMZ,NCOL,NROW,NLAY,INBAS,OUNIT,LCIBOU,LCSTRT,LCIOFG)
      IF (IUNIT(1) .GT. 0)CALL BCFDAL (ISUMX,ISUMZ,IUNIT(1),ISS,NCOL,
    1 NROW,NLAY,OUNIT,IBCFCB,LCSC1,LCBOT,LCTOP,LCDELC,LCDELR,NOPRT)
      IF (IUNIT(2) .GT. 0)CALL WELDAL
    1 (ISUMX,ISUMZ,MXWELL,IUNIT(2),OUNIT,IWELCB,LCWELL)
      IF (IUNIT(3) .GT. 0)CALL DRNDAL(ISUMX,IUNIT(3),OUNIT,IDRNCB)
      IF (IUNIT(8) .GT. 0)CALL RCHDAL(ISUMX,ISUMZ,NRCHOP,IUNIT(8),
    1 OUNIT,IRCHCB,NCOL,NROW,LCIRCH,LCRECH)
      IF (IUNIT(5) .GT. 0)CALL EVTDAL(ISUMX,IUNIT(5),OUNIT,IEVTCB)
      IF (IUNIT(4) .GT. 0)CALL RIVDAL(ISUMX,IUNIT(4),OUNIT,IRIVCB)
      IF (IUNIT(7) .GT. 0)CALL GHBDAL(ISUMX,IUNIT(7),OUNIT,IGHBCB)
      IF (IUNIT(9) .GT. 0)CALL SIPDAL(ISUMX,IUNIT(9),OUNIT)
      IF (IUNIT(11).GT. 0)CALL SORDAL(ISUMX,IUNIT(11),OUNIT)
C
C4 -- CHECK THE REMAINING SPACE IN THE "Z" ARRAY AND ALLOCATE IT
      CALL CTL1CK (ISUMX,ISUMZ,LENZ,NLAY,NODES,LCGRAF,LNGRAF,OUNIT)
```

```
C
C5 -- READ AND PREPARE DATA FOR THE ENTIRE SIMULATION
      IF (INBAS .GT. 0) CALL BASDRP (Z(LCIBOU),Z(LCSTRT),INBAS,NCOL,
     1 NROW,NLAY,NODES,IUNIT(12),IHEDUN,IDDNUN,OUNIT,NOPRT)
      IF (IUNIT(1) .GT. 0)CALL BCFDRP (Z(LCIBOU),Z(LCSC1),Z(LCBOT),
     1 Z(LCTOP),Z(LCDELR),Z(LCDELC),Z(LCDELL),IUNIT(1),ISS,NCOL,NROW,
     2 NLAY,NODES,OUNIT,Z(LCU3DS(1)),NOPRT)
      IF (IUNIT(9) .GT. 0) CALL SIPDRP (IUNIT(9),OUNIT)
      IF (IUNIT(11).GT. 0)CALL SORDRP (IUNIT(11),OUNIT)
      IF (INBAS .GT. 0) CALL BASDST (INBAS,NPER,Z(LCNTS),KSP,KTS,NLAY,
     1  Z(LCIOFG),OUNIT)
C
C6 -- ADVANCE TO FIRST STRESS PERIOD AND TIME STEP OF MODEL INPUT DATA
      IF (IUNIT(2) .GT. 0)CALL WELDRP (NROW,NCOL,NLAY,Z(LCWELL),
     1 NWELLS,MXWELL,IUNIT(2),IPKGSP(2),OUNIT,.TRUE.,NOPRT)
      IF (IUNIT(3) .GT. 0)CALL DRNDRP (IUNIT(3),OUNIT)
      IF (IUNIT(8) .GT. 0)CALL RCHDRP (NRCHOP,Z(LCIRCH),Z(LCRECH),
     1 NROW,NCOL,NLAY,IUNIT(8),IPKGSP(8),OUNIT,.TRUE.,NOPRT)
      IF (IUNIT(5) .GT. 0)CALL EVTDRP (NCOL,NROW,IUNIT(5),OUNIT,
     1  Z(LCU2DS(1)),NOPRT)
      IF (IUNIT(4) .GT. 0)CALL RIVDRP (IUNIT(4),OUNIT)
      IF (IUNIT(7) .GT. 0)CALL GHBDRP (IUNIT(7),OUNIT)
C
C7 -- READ AND PROCESS USER COMMANDS
 1000 CALL CTL1CM (LINE,NROW,NCOL,NLAY,INCTL,OUNIT,EXCMD,EOF,UBIN,
     1  U2DDSN,U3DDSN,U2DANM,U3DANM,ISKSP,ISKTS,LCSTRT,LCIBOU,LCSC1,
     2  LCTOP,LCIOFG,LCBOT,LCDELR,LCDELC,LCWELL,NWELLS,MXWELL,NRCHOP,
     3  LCIRCH,LCRECH,LCU2DS,LCU3DS,LCWDS,IUNIT,NUCLAS,CLASES,TITLE,
     4  Z(LCIBOU),Z(LCUBOU),Z(LCWDS(1)),Z(LCWDS(2)),Z(LCU2DS(4)),
     5  Z(LCU3DS(4)),NPER,Z(LCNTS),KSP,KTS,Z(LCIOFG),LCUBOU,LCDELL,
     6  Z(LCDELR),Z(LCDELC),Z(LCDELL),LCGRAF,LNGRAF,IBCFCB,IWELCB,
     7  IDRNCB,IRCHCB,IEVTCB,IRIVCB,IGHBCB,IHEDUN,IDDNUN,IPKGSP,PLANE,
     8  NOPRT)
C
C8 -- IF NO COMMANDS GIVEN, EXECUTE THE DEFAULT COMMANDS
      IF (EOF .AND. NCMDRD .EQ. 0)
     X CALL DFT1CM (LINE,NROW,NCOL,NLAY,INCTL,OUNIT,EXCMD,EOF,UBIN,
     1  U2DDSN,U3DDSN,U2DANM,U3DANM,ISKSP,ISKTS,LCSTRT,LCIBOU,LCSC1,
     2  LCTOP,LCIOFG,LCBOT,LCDELR,LCDELC,LCWELL,NWELLS,MXWELL,NRCHOP,
     3  LCIRCH,LCRECH,LCU2DS,LCU3DS,LCWDS,IUNIT,NUCLAS,CLASES,TITLE,
     4  Z(LCIBOU),Z(LCUBOU),Z(LCWDS(1)),Z(LCWDS(2)),Z(LCU2DS(4)),
     5  Z(LCU3DS(4)),NPER,Z(LCNTS),KSP,KTS,Z(LCIOFG),LCUBOU,LCDELL,
     6  Z(LCDELR),Z(LCDELC),Z(LCDELL),LCGRAF,LNGRAF,IBCFCB,IWELCB,
     7  IDRNCB,IRCHCB,IEVTCB,IRIVCB,IGHBCB,IHEDUN,IDDNUN,IPKGSP,PLANE,
     8  NOPRT,NCMDEX,NCMDRD)
C
C9 -- ITERATE COMMAND PROCESSING UNTIL END-OF-FILE REACHED ON COMMAND FILE
      IF (EOF) GOTO 2001
      NCMDRD = NCMDRD + 1
      IF (EXCMD) NCMDEX = NCMDEX + 1
      GOTO 1000
C
C10 - END OF PROGRAM
 2001 WRITE (OUNIT,9) NCMDEX,NCMDRD
    9 FORMAT ('1 MODULAR GROUND-WATER MODEL STATISTICAL PROCESSING',
     1   ' TERMINATING NORMALLY',//,I9,' COMMANDS EXECUTED',/,I9,
     2   ' COMMANDS READ')
      ENDFILE (OUNIT)
      STOP
      END
      SUBROUTINE CTL1DF (INCTL,OUNIT,TITLE,UBIN,PLANE,EOF,IUNIT,NUCLAS,
     1  IPKGSP,NCMDRD,NCMDEX,NOPRT)
C
C     READ TITLES & INITIALIZE VARIABLES
C
      INTEGER OUNIT,IUNIT(24),IPKGSP(12)
      CHARACTER TITLE(4)*128, PRTFLG*1
      DIMENSION HEADNG(32), PLANE(4)
      LOGICAL UBIN, EOF
C
C1 -- SET FLAGS FOR ENTRY OF USER-BOUNDARY, AND END OF COMMAND FILE TO FALSE
      UBIN = .FALSE.
      EOF  = .FALSE.
C
C2 -- SET NUMBER OF USER-SPECIFIED FREQUENCY CLASSES, NUMBER OF COMMANDS READ,
C      NUMBER OF COMMANDS EXECUTED TO ZERO
      NUCLAS = 0
      NCMDRD = 0
      NCMDEX = 0
C
C3 -- SET THE TIME VARIABLE FOR ALL PACKAGES TO ZERO
      DO 10 I=1,12
         IPKGSP(I) = 0
10    CONTINUE
C
C4 -- SET THE COEFFICIENTS OF THE SLICING PLANE TO ZERO
      DO 15 I=1,4
         PLANE(I) = 0.0
15    CONTINUE
C
C5 -- READ AND WRITE HEADING FROM THE COMMAND FILE
      READ  (INCTL,1,ERR=98,END=99) HEADNG
      WRITE (TITLE(2),5) HEADNG
      READ  (INCTL,1,ERR=98,END=99) HEADNG,PRTFLG
      WRITE (TITLE(3),5) HEADNG
20    CONTINUE
    1 FORMAT (20A4,/,12A4,:,31X,A1)
    5 FORMAT (32A4)
      TITLE(4) = ''
      NOPRT = 0
      IF (PRTFLG .NE. ' ') NOPRT = -1
      RETURN
C
   98 WRITE (OUNIT,3) INCTL
    3 FORMAT (' CTL1DF:  Error reading from unit:  ',I2)
      STOP
   99 WRITE (OUNIT,4) INCTL
    4 FORMAT (' CTL1DF:  Unexpected E-O-F on unit:  ',I2)
      STOP
      END
      SUBROUTINE BASDDF (TITLE,NCOL,NROW,NLAY,NODES,INBAS,OUNIT,
     1  IUNIT,NPER)
C
C     DEFINE BASIC MODEL PARAMETERS
C
      DIMENSION IUNIT(24),HEADNG(32)
      CHARACTER*128 TITLE(4)
      INTEGER OUNIT
C
C1 -- READ/PRINT TITLE LINES
      WRITE (OUNIT,1)
    1 FORMAT (1H1,20X,'U.S. GEOLOGICAL SURVEY MODULAR',
     1        ' FINITE-DIFFERENCE GROUND-WATER MODEL',/,
     2        30X,'STATISTICAL PRE- AND POST- PROCESSOR',/)
      READ (INBAS,2,ERR=98,END=99) HEADNG
      WRITE (TITLE(1),6) HEADNG
      WRITE (OUNIT,3) TITLE(1)
    3 FORMAT (1X,A128)
    6 FORMAT (32A4)
    2 FORMAT (20A4)
C
C2 -- READ/PRINT SIMULATION SIZING VARIABLES
      READ (INBAS,4,ERR=98,END=99) NLAY,NROW,NCOL,NPER,IPAD1
    4 FORMAT (5I10)
      NODES = NCOL * NROW * NLAY
      WRITE (OUNIT,5) NLAY,NROW,NCOL,NODES,NPER
    5 FORMAT (/,' LAYERS =',I10,' ROWS =',I10,' COLUMNS =',I10,
     1        ' NODES =',I10,' STRESS PERIODS ='I10)
C
C3 -- READ THE ARRAY DESCRIBING WHICH PACKAGES/FILE UNITS ARE USED
      READ (INBAS,101,ERR=98,END=99) IUNIT
  101 FORMAT (24I3)
      WRITE (OUNIT,102) (I,I=1,24), IUNIT
  102 FORMAT (//,' I/O UNITS:',/,1X,'ELEMENT OF IUNIT:',24I3,
     1                         /,1X,'         I/O UNIT:',24I3)
      RETURN
C
   98 WRITE (OUNIT,981) INBAS
  981 FORMAT (' BASDDF:  Error reading from unit:   ',I2)
      STOP
   99 WRITE (OUNIT,993) INBAS
  993 FORMAT (' BASDDF:  Unexpected E-O-F on unit:   ',I2)
      STOP
      END
      SUBROUTINE CTL1AL (ISUMX,ISUMZ,NCOL,NROW,NLAY,NPER,LCNTS,
     1 LCUBND,LCU2DS,LCU3DS,LCWDS,U2DDSN,U2DANM,U3DDSN,U3DANM,
     2 ISKSP,ISKTS,LCWELL,LCIRCH,LCRECH,LCDELL)
C
C     ALLOCATE SPACE NEEDED FOR CORE OF STATISTICAL PACKAGE
C       (VARIABLE ISUMX IS INCLUDED FOR POSSIBLE FUTURE SIZING OF MODEL)
C
$INSERT STKSIZE.INS
$INSERT STKDEF.INS
      INTEGER LCWDS(2)
C
C1 -- DEFINE CONSTANTS AND SET THE "Z" ARRAY POINTER AT THE BEGINNING
      NCR   = NCOL * NROW
      NCRL  = NCR * NLAY
      ISUMX = 1
      ISUMZ = 1
C
C2 -- ALLOCATE SPACE FOR NUMBER OF TIME STEPS IN EACH STRESS PERIOD
      LCNTS = ISUMZ
      ISUMZ = ISUMZ + NPER
C
C3 -- ALLOCATE SPACE FOR WORKING DATA SETS
      LCWDS(1) = ISUMZ
      ISUMZ    = ISUMZ + NCRL * 2
      LCWDS(2) = ISUMZ
      ISUMZ    = ISUMZ + NCRL * 2
C
C4 -- ALLOCATE SPACE FOR USER BOUNDARY ARRAY
      LCUBND = ISUMZ
      ISUMZ  = ISUMZ + NCRL
C
C5 -- ALLOCATE SPACE FOR USER 2-DIMENSIONAL STACK
      DO 1 I=1,ISTKSZ
         LCU2DS(I) = ISUMZ
         ISUMZ     = ISUMZ + NCR * 2
         U2DDSN(I) = ''
         U2DANM(I) = ''
    1 CONTINUE
C
C6 -- ALLOCATE SPACE FOR USER 3-DIMENSIONAL STACK
      DO 2 I=1,ISTKSZ
         LCU3DS(I) = ISUMZ
         ISUMZ     = ISUMZ + NCRL * 2
         U3DDSN(I) = ''
         U3DANM(I) = ''
         ISKSP(I)  = 0
         ISKTS(I)  = 0
    2 CONTINUE
C
C7 -- ALLOCATE SPACE FOR THICKNESS OF LAYERS
      LCDELL = ISUMZ
      ISUMZ  = ISUMZ + NLAY
C
C8 -- PLACE INITIAL ARRAY POINTERS FOR WELL, IRCH, AND RECH
      LCWELL = 1
      LCIRCH = 1
      LCRECH = 1
      RETURN
      END
      SUBROUTINE BASDAL (ISUMX,ISUMZ,NCOL,NROW,NLAY,IN,OUNIT,
     1                    LCIBOU,LCSTRT,LCIOFG)
C
C     ALLOCATE SPACE FOR THE BASIC PACKAGE
C
      INTEGER OUNIT
C
C1 -- READ & IGNORE:  IAPART AND ISTRT
      READ (IN,1,ERR=98,END=99) IAPART, ISTRT
    1 FORMAT (2I10)
C
C2 -- ALLOCATE SPACE FOR MODEL BOUNDARY ARRAY
      NCR = NCOL * NROW
      NCRL = NCR * NLAY
      LCIBOU = ISUMZ
      ISUMZ = ISUMZ + NCRL
C
C3 -- ALLOCATE SPACE FOR STARTING HEADS
      LCSTRT = ISUMZ
      ISUMZ = ISUMZ + NCRL
C
C4 -- ALLOCATE SPACE FOR LAYER I/O FLAGS (ONLY DISK-NOT PRINT)
      LCIOFG = ISUMZ
      ISUMZ  = ISUMZ + (2 * NLAY)
      RETURN
C
   98 WRITE (OUNIT,2) IN
    2 FORMAT (' BASDAL:  Error reading from unit:   ',I2)
      STOP
   99 WRITE (OUNIT,3) IN
    3 FORMAT (' BASDAL:  Unexpected E-O-F on unit:   ',I2)
      STOP
      END

      SUBROUTINE BCFDAL (ISUMX,ISUMZ,IN,ISS,NCOL,NROW,NLAY,OUNIT,IBCFCB,
     1                    LCSC1,LCBOT,LCTOP,LCDELC,LCDELR,NOPRT)
C
C     ALLOCATE SPACE FOR THE BLOCK-CENTERED FLOW PACKAGE
C
      INTEGER OUNIT
$INSERT FLWCOM.INS
C
C1 -- READ STEADY-STATE FLAG, AND BCF CELL-BY-CELL UNIT NUMBER
      READ (IN,2,ERR=98,END=99) ISS,IBCFCB
    2 FORMAT (2I10)
C
C2 -- CHECK IF DIMENSION OF LAYCON HAS BEEN EXCEEDED
      IF (NLAY .GT. 80) THEN
      WRITE (OUNIT,11) NLAY
   11 FORMAT (' Number of layers specified:',I4,/,
     1         ' greater than 80...stopping due to insufficient space')
      STOP
      ENDIF
C
C3 -- READ LAYER-TYPE INDEX
      READ (IN,51) (LAYCON(I),I=1,NLAY)
   51 FORMAT (40I2)
      IF (NOPRT .GE. 0) WRITE (OUNIT,52)
   52 FORMAT (//,6X,'LAYER  AQUIFER TYPE',/,6X,19('-'))
```

```
C
C4 -- CALCULATE NUMBER OF TOP AND BOTTOM LAYERS
      NBOT = 0
      NTOP = 0
      DO 100 I=1,NLAY
         L = LAYCON (I)
         IF (NOPRT .GE. 0) WRITE (OUNIT,7) I,L
    7    FORMAT (1X,I9,I10)
         IF (LAYCON(I) .EQ. 1 .AND. I .NE. 1) THEN
             WRITE (OUNIT,8)
    8        FORMAT (' Aquifer type 1 is only allowed in top layer')
             STOP
         ENDIF
C
C4A --    LAYER TYPE 1 AND 3 NEED SPACE FOR A BOTTOM ARRAY
         IF (L .EQ. 1 .OR. L .EQ. 3) NBOT = NBOT + 1
C
C4B --    LAYER TYPE 2 AND 3 NEED SPACE FOR A TOP ARRAY
         IF (L .EQ. 2 .OR. L .EQ. 3) NTOP = NTOP + 1
  100 CONTINUE
C
C5 -- ALLOCATE SPACE FOR STORAGE COEFFICIENT
      NCR = NROW * NCOL
      IF (ISS .EQ. 0) THEN
         LCSC1 = ISUMZ
         ISUMZ = ISUMZ + (NLAY * NCR)
      ELSE
         LCSC1 = 1
      ENDIF
C
C6 -- ALLOCATE SPACE FOR AQUIFER LAYER BOTTOM
      LCBOT = ISUMZ
      ISUMZ = ISUMZ + (NBOT * NCR)
C
C7 -- ALLOCATE SPACE FOR AQUIFER LAYER TOP
      LCTOP = ISUMZ
      ISUMZ = ISUMZ + (NTOP * NCR)
C
C8 -- ALLOCATE SPACE FOR ROW AND COLUMN WIDTHS
      LCDELR = ISUMZ
      ISUMZ  = ISUMZ + NCOL
      LCDELC = ISUMZ
      ISUMZ  = ISUMZ + NROW
      RETURN
C
   98 WRITE (OUNIT,992) IN
  992 FORMAT (' BCFDAL:  Error reading from unit:   ',I2)
      STOP
   99 WRITE (OUNIT,993) IN
  993 FORMAT (' BCFDAL:  Unexpected E-O-F on unit:   ',I2)
      STOP
      END
      SUBROUTINE WELDAL (ISUMX,ISUMZ,MXWELL,IN,OUNIT,IWELCB,LCWELL)
C
C     ALLOCATE SPACE FOR WELL PACKAGE
C
      INTEGER OUNIT
C
C1 -- READ MAXIMUM NUMBER OF WELLS AND CELL-BY-CELL FLOW UNIT NUMBER
      READ (IN,1,ERR=98,END=99) MXWELL,IWELCB
    1 FORMAT (2I10)
C
C2 -- ALLOCATE SPACE FOR THE WELL ARRAY
      LCWELL = ISUMZ
      ISUMZ = ISUMZ + (4 * MXWELL)
      RETURN
C
   98 WRITE (OUNIT,2) IN
    2 FORMAT (' WELDAL:  Error reading from unit:  ',I2)
      STOP
   99 WRITE (OUNIT,3) IN
    3 FORMAT (' WELDAL:  Unexpected E-O-F on unit:  ',I2)
      STOP
      END

      SUBROUTINE DRNDAL (ISUMX,IN,OUNIT,IDRNCB)
C
C     NO SPACE ALLOCATED FOR THE DRAIN PACKAGE
C
      INTEGER OUNIT
C
C1 -- READ & IGNORE MXDRN, RETURN DRAIN CELL-BY-CELL FLOW UNIT NUMBER
      READ (IN,1,ERR=98,END=99) MXDRN,IDRNCB
    1 FORMAT (2I10)
      RETURN
C
   98 WRITE (OUNIT,2) IN
    2 FORMAT (' DRNDAL:  Error reading from unit:  ',I2)
      STOP
   99 WRITE (OUNIT,3) IN
    3 FORMAT (' DRNDAL:  Unexpected E-O-F on unit:  ',I2)
      STOP
      END
      SUBROUTINE RCHDAL (ISUMX,ISUMZ,NRCHOP,IN,OUNIT,IRCHCB,
     1    NCOL,NROW,LCIRCH,LCRECH)
C
C     ALLOCATE SPACE FOR RECHARGE PACKAGE
C
      INTEGER OUNIT
C
C1 -- READ RECHARGE OPTION FLAG AND CELL-BY-CELL UNIT NUMBER
      READ (IN,1,ERR=98,END=99) NRCHOP,IRCHCB
    1 FORMAT (2I10)
C
C2 -- CHECK RECHARGE OPTION FLAG
      IF (NRCHOP .LT. 1 .OR. NRCHOP .GT. 3) THEN
         WRITE (OUNIT,990) NRCHOP
  990    FORMAT (' Illegal recharge option:',I10,/,' ...stopping')
         STOP
      ENDIF
C
C3 -- ALLOCATE SPACE FOR RECHARGE INDEX ARRAY, IF NEEDED
      LCIRCH = 1
      IF (NRCHOP .EQ. 2) THEN
         LCIRCH = ISUMZ
         ISUMZ  = ISUMZ + NCOL * NROW
      ENDIF
C
C4 -- ALLOCATE SPACE FOR RECHARGE ARRAY
      LCRECH = ISUMZ
      ISUMZ = ISUMZ + NCOL * NROW
      RETURN
C
   98 WRITE (OUNIT,2) IN
    2 FORMAT (' RCHDAL:  Error reading from unit:   ',I2)
      STOP
   99 WRITE (OUNIT,3) IN
    3 FORMAT (' RCHDAL:  Unexpected E-O-F on unit:   ',I2)
      STOP
      END

      SUBROUTINE EVTDAL (ISUMX,IN,OUNIT,IEVTCB)
C
C     NO SPACE ALLOCATED FOR THE EVAPO-TRANSPIRATION PACKAGE
C
      INTEGER OUNIT
C
C1 -- READ & IGNORE NEVTOP, RETURN EVT CELL-BY-CELL FLOW UNIT NUMBER
      READ (IN,1,ERR=98,END=99) NEVTOP,IEVTCB
    1 FORMAT (2I10)
      RETURN
C
   98 WRITE (OUNIT,2) IN
    2 FORMAT (' EVTDAL:  Error reading from unit:   ',I2)
      STOP
   99 WRITE (OUNIT,3) IN
    3 FORMAT (' EVTDAL:  Unexpected E-O-F on unit:   ',I2)
      STOP
      END
      SUBROUTINE RIVDAL (ISUMX,IN,OUNIT,IRIVCB)
C
C     NO SPACE ALLOCATED FOR THE RIVER PACKAGE
C
      INTEGER OUNIT
C
C1 -- READ & IGNORE MXRIVR, RETURN RIVER CELL-BY-CELL FLOW UNIT NUMBER
      READ (IN,1,ERR=98,END=99) MXRIVR,IRIVCB
    1 FORMAT (2I10)
      RETURN
C
   98 WRITE (OUNIT,2) IN
    2 FORMAT (' RIVDAL:  Error reading from unit:  ',I2)
      STOP
   99 WRITE (OUNIT,3) IN
    3 FORMAT (' RIVDAL:  Unexpected E-O-F on unit:  ',I2)
      STOP
      END

      SUBROUTINE GHBDAL (ISUMX,IN,OUNIT,IGHBCB)
C
C     NO SPACE ALLOCATED FOR THE GENERAL HEAD BOUNDARY PACKAGE
C
      INTEGER OUNIT
C
C1 -- READ & IGNORE MXBND, RETURN GHB CELL-BY-CELL FLOW FLAG
      READ (IN,1,ERR=98,END=99) MXBND,IGHBCB
    1 FORMAT (2I10)
      RETURN
C
   98 WRITE (OUNIT,2) IN
    2 FORMAT (' GHBDAL:  Error reading from unit:  ',I2)
      STOP
   99 WRITE (OUNIT,3) IN
    3 FORMAT (' GHBDAL:  Unexpected E-O-F on unit:  ',I2)
      STOP
      END

      SUBROUTINE SIPDAL (ISUMX,IN,OUNIT)
C
C     NO SPACE ALLOCATED FOR THE STRONGLY IMPLICIT SOLVER PACKAGE
C
      INTEGER OUNIT
C
C1 -- READ & IGNORE:  MXITER AND NPARM
      READ (IN,1,ERR=98,END=99) MXITER, NPARM
    1 FORMAT (2I10)
      RETURN
C
   98 WRITE (OUNIT,2) IN
    2 FORMAT (' SIPDAL:  Error reading from unit:  ',I2)
      STOP
   99 WRITE (OUNIT,3) IN
    3 FORMAT (' SIPDAL:  Unexpected E-O-F on unit:  ',I2)
      STOP
      END
      SUBROUTINE SORDAL (ISUMX,IN,OUNIT)
C
C     NO SPACE ALLOCATED FOR THE SLICE-SUCCESSIVE OVER-RELAXATION SOLVER PACKAGE
C
      INTEGER OUNIT
C
C1 -- READ & IGNORE:  MXITER
      READ (IN,1,ERR=98,END=99) MXITER
    1 FORMAT (I10)
      RETURN
C
   98 WRITE (OUNIT,2) IN
    2 FORMAT (' SORDAL:  Error reading from unit:  ',I2)
      STOP
   99 WRITE (OUNIT,3) IN
    3 FORMAT (' SORDAL:  Unexpected E-O-F on unit:  ',I2)
      STOP
      END

      SUBROUTINE CTL1CK (ISUMX,ISUMZ,LENZ,NLAY,NODES,LCGRAF,LNGRAF,
     1                   OUNIT)
C
C     CHECK COMPATIBILITY OF ARRAY SIZE(S) & ALLOCATE SPACE FOR VECTORS
C
      INTEGER OUNIT
C
C1 -- CHECK FOR SUFFICIENT DYNAMIC STORAGE, IF LACKING THEN STOP
      IF (ISUMZ .GT. LENZ) THEN
         WRITE (OUNIT,1)
         WRITE (OUNIT,2) ISUMZ,LENZ
    1    FORMAT (//,' DYNAMIC STORAGE EXCEEDED')
    2    FORMAT (/,I8,' ELEMENTS ALLOCATED',/,I8,' ELEMENTS AVAILABLE')
         STOP
      ELSE
         WRITE (OUNIT,3)
    3    FORMAT (//,' DYNAMIC STORAGE UTILIZATION')
         WRITE (OUNIT,2) ISUMZ,LENZ
         PAD = 100.0 * FLOAT (ISUMZ) / FLOAT (LENZ)
         WRITE (OUNIT,4) PAD
    4    FORMAT (/,F8.1,'% UTILIZED')
C
C2 -- ALLOCATE SPACE FOR GRAPHIC STORAGE, IF LACKING THEN WARN
         LCGRAF = ISUMZ
         LNGRAF = LENZ - ISUMZ
         LENEED = NODES * 4 / NLAY
         IF (LNGRAF .GT. LENEED) THEN
             WRITE (OUNIT,7) LNGRAF
         ELSE
             LACKED = LENEED - LNGRAF
             WRITE (OUNIT,8) LACKED
         ENDIF
      ENDIF
      RETURN
    7 FORMAT (/,I8,' elements available for graphic storage')
    8 FORMAT (/,' Insufficient space available for vector calculation',
     1        /,' Need ',I9,' more elements of dynamic storage')
      END
      SUBROUTINE BASDRP (IBOUND,STRT,INBAS,NCOL,NROW,NLAY,
     1    NODES,INOC,IHEDUN,IDDNUN,IOUT,NOPRT)
C
C     READ AND PREPARE BASIC MODEL PACKAGE
C
      DIMENSION IBOUND(NODES),STRT(NODES)
      DIMENSION ANAME (6,2)
      DATA (ANAME(I,1),I=1,6) /'    ','    ','  BO','UNDA','RY A',
     1                         'RRAY'/
      DATA (ANAME(I,2),I=1,6) /'    ','    ','    ','INIT','IAL ',
     1                         'HEAD'/
C
      NCR = NCOL * NROW
C
C1 -- READ THE MODEL BOUNDARY ARRAY
      DO 100 K=1,NLAY
      LOC = 1 + (K-1) * NCR
      CALL U2DINT (IBOUND(LOC),ANAME(1,1),NROW,NCOL,K,INBAS,IOUT,NOPRT)
  100 CONTINUE
C
C2 -- READ THE STARTING HEAD ARRAY AND SET NO-FLOW NODES TO FIXED HEAD
      READ (INBAS,2) HNOFLO
    2 FORMAT (F10.0)
C
      DO 300 K=1,NLAY
      LOC = 1 + (K-1) * NCR
      CALL U2DREL (STRT(LOC),ANAME(1,2),NROW,NCOL,K,INBAS,IOUT,NOPRT)
  300 CONTINUE
C
      IF (NOPRT .GE. 0) WRITE (IOUT,3) HNOFLO
    3 FORMAT (30X,'AQUIFER HEAD SET EQUAL TO ',1PG11.5,
     1         ' AT ALL NO-FLOW NODES (IBOUND=0).')
      DO 400 I=1,NODES
      IF (IBOUND(I) .EQ. 0) STRT (I) = HNOFLO
  400 CONTINUE
C
C3 -- SKIP COMPUTED HEAD & DRAWDOWN FORMATS,
C     READ COMPUTED HEAD AND DRAWDOWN UNIT NUMBERS
C     SBAS1I INCORPORATED HEREIN:  READING OUTPUT CONTROL PACKAGE
      IHEDUN = 0
      IDDNUN = 0
      IF (INOC .GT. 0) READ (INOC,501) IHEDFM,IDDNFM,IHEDUN,IDDNUN
  501 FORMAT (4I10)
      IF (IHEDUN .NE. 0) THEN
         IF (NOPRT .GE. 0) WRITE (IOUT,502) '   Heads',IHEDUN
  502    FORMAT (40X,A8,'  available for reading on unit ',I3)
      ELSE
         IF (NOPRT .GE. 0) WRITE (IOUT,503) '   Heads'
  503    FORMAT (40X,A8,' unavailable for reading')
      ENDIF
      IF (IDDNUN .NE. 0) THEN
         IF (NOPRT .GE. 0) WRITE (IOUT,502) 'Drawdown',IDDNUN
      ELSE
         IF (NOPRT .GE. 0) WRITE (IOUT,503) 'Drawdown'
      ENDIF
      RETURN
      END
      SUBROUTINE BCFDRP (IBOUND,SC1,BOT,TOP,DELR,DELC,DELL,IN,ISS,
     1                   NCOL,NROW,NLAY,NODES,OUNIT,DUMMY,NOPRT)
C
C     READ AND PREPARE BLOCK-CENTERED FLOW DATA
C
      INTEGER OUNIT
      DIMENSION SC1(NODES),ANAME(6,10),BOT(NODES),TOP(NODES)
      DIMENSION IBOUND(NODES),DUMMY(NODES)
      DIMENSION DELR(NCOL),DELC(NROW),DELL(NLAY)
$INSERT FLWCOM.INS
C
      DATA (ANAME(I,1),I=1,6)
     1 /'    ','PRIM','ARY ','STOR','AGE ','COEF'/
      DATA (ANAME(I,2),I=1,6)
     1 /'    ','TRAN','SMIS','. AL','ONG ','ROWS'/
      DATA (ANAME(I,3),I=1,6)
     1 /'   H','YD. ','COND','. AL','ONG ','ROWS'/
      DATA (ANAME(I,4),I=1,6)
     1 /'VERT',' HYD',' CON','D /T','HICK','NESS'/
      DATA (ANAME(I,5),I=1,6)
     1 /'    ','    ','    ','    ','  BO','TTOM'/
      DATA (ANAME(I,6),I=1,6)
     1 /'    ','    ','    ','    ','    ',' TOP'/
      DATA (ANAME(I,7),I=1,6)
     1 /'  SE','COND','ARY ','STOR','AGE ','COEF'/
      DATA (ANAME(I,8),I=1,6)
     1 /'COLU','MN T','O RO','W AN','ISOT','ROPY'/
      DATA (ANAME(I,9),I=1,6)
     1 /'    ','    ','    ','    ','    ','DELR'/
      DATA (ANAME(I,10),I=1,6)
     1 /'    ','    ','    ','    ','    ','DELC'/
C
C1 -- READ AND IGNORE:  TRPY
      CALL U1DREL (DUMMY,ANAME(1,8),NLAY,IN,OUNIT,NOPRT)
C
C2 -- READ WIDTH OF ROWS AND COLUMNS, DEFINE DEFAULT LAYER THICKNESS
      CALL U1DREL (DELR,ANAME(1,9),NCOL,IN,OUNIT,NOPRT)
      CALL U1DREL (DELC,ANAME(1,10),NROW,IN,OUNIT,NOPRT)
      THICK = 0.0
      DO 50 I=1,NROW
50       THICK = THICK + DELC(I)
      DO 60 I=1,NCOL
60       THICK = THICK + DELR(I)
      THICK = THICK / (NCOL + NROW)
      DO 70 I=1,NLAY
70       DELL(I) = THICK
      WRITE (OUNIT,1) THICK
    1 FORMAT (54X,'DEFAULT LAYER THICKNESS =',G15.7)
C
C3 -- READ STORAGE COEFFICIENT
      NIJ = NCOL * NROW
      KT = 0
      KB = 0
C
      DO 200 K=1,NLAY
        IF (LAYCON(K) .EQ. 1 .OR. LAYCON(K) .EQ. 3) KB = KB + 1
        IF (LAYCON(K) .EQ. 2 .OR. LAYCON(K) .EQ. 3) KT = KT + 1
        LOC  = 1 + (K-1) * NIJ
        LOCB = 1 + (KB-1) * NIJ
        LOCT = 1 + (KT-1) * NIJ
C
        IF (ISS .EQ. 0)
     1    CALL U2DREL (SC1(LOC),ANAME(1,1),NROW,NCOL,K,IN,OUNIT,NOPRT)
```

```
C
          IF (LAYCON(K) .EQ. 0 .OR. LAYCON(K) .EQ. 2) THEN
          CALL U2DREL (DUMMY,ANAME(1,2),NROW,NCOL,K,IN,OUNIT,NOPRT)
        ELSE
          CALL U2DREL (DUMMY,ANAME(1,3),NROW,NCOL,K,IN,OUNIT,NOPRT)
          CALL U2DREL (BOT(LOCB),ANAME(1,5),NROW,NCOL,K,IN,OUNIT,NOPRT)
        ENDIF
C
        IF (K .NE. NLAY)
     1     CALL U2DREL (DUMMY,ANAME(1,4),NROW,NCOL,K,IN,OUNIT,NOPRT)
        IF (LAYCON(K) .EQ. 2 .OR. LAYCON(K) .EQ. 3) THEN
          IF (ISS .EQ. 0)
     1       CALL U2DREL (DUMMY,ANAME(1,7),NROW,NCOL,K,IN,OUNIT,NOPRT)
          CALL U2DREL (TOP(LOCT),ANAME(1,6),NROW,NCOL,K,IN,OUNIT,NOPRT)
        ENDIF
  200 CONTINUE
C
C     NO NEED TO CALL SBCF1N
      RETURN
      END

      SUBROUTINE SIPDRP (IN,OUNIT)
C
C     READ & IGNORE SIP PACKAGE INPUT
C
      INTEGER OUNIT
C
C1 -- READ AND IGNORE:  ACCL,HCLOSE,IPCALC,WSEED,IPRSIP
      READ (IN,1,ERR=98,END=99) ACCL,HCLOSE,IPCALC,WSEED,IPRSIP
    1 FORMAT (2F10.0,I10,F10.0,I10)
      RETURN
C
   98 WRITE (OUNIT,2) IN
    2 FORMAT (' SIPDRP:  Error reading from unit:  ',I2)
      STOP
   99 WRITE (OUNIT,3) IN
    3 FORMAT (' SIPDRP:  Unexpected E-O-F on unit:  ',I2)
      STOP
      END

      SUBROUTINE SORDRP (IN,OUNIT)
C
C     READ AND IGNORE SOR PACKAGE INPUT
C
      INTEGER OUNIT
C
C1 -- READ AND IGNORE:  ACCL,HCLOSE,IPRSOR
      READ (IN,1,ERR=98,END=99) ACCL,HCLOSE,IPRSOR
    1 FORMAT (2F10.0,I10)
      RETURN
C
   98 WRITE (OUNIT,2) IN
    2 FORMAT (' SORDRP:  Error reading from unit:  ',I2)
      STOP
   99 WRITE (OUNIT,3) IN
    3 FORMAT (' SORDRP:  Unexpected E-O-F on unit:  ',I2)
      STOP
      END
      SUBROUTINE BASDST (INBAS,NPER,NTS,KSP,KTS,NLAY,IOFLG,OUNIT)
C
C     ROUTINE TO FAST-FORWARD PAST READ OF: PERLEN & TSMULT
C       AND SAVE NUMBER OF TIME-STEPS/STRESS PERIOD INTO NTS
C       NOTE:  THIS IS CALLED ONLY ONCE UNLIKE IN THE MODULAR MODEL
C
      INTEGER OUNIT, NTS (NPER), IOFLG (NLAY,2)
C
C1 -- READ NUMBER OF TIME-STEPS / STRESS PERIOD INTO ARRAY:  NTS
      KSP = 1
      KTS = 0
      DO 10 I=1,NPER
         READ (INBAS,1,ERR=98,END=99) PERLEN,NTS(I),TSMULT
   10 CONTINUE
    1 FORMAT (F10.0,I10,F10.0)
C
C2 -- INITIALIZE IOFLG ARRAY TO ZERO FOR ALL LAYERS
      DO 20 I=1,NLAY
         IOFLG (I,1) = 0
         IOFLG (I,2) = 0
   20 CONTINUE
      RETURN
C
   98 WRITE (OUNIT,2) IN
    2 FORMAT (' BASDST:  Error reading from unit:  ',I2)
      STOP
   99 WRITE (OUNIT,3) IN
    3 FORMAT (' BASDST:  Unexpected E-O-F on unit:  ',I2)
      STOP
      END

      SUBROUTINE WELDRP (NROW,NCOL,NLAY,WELL,NWELLS,MXWELL,IN,IWELSP,
     1  OUNIT,IWLFLG,NOPRT)
C
C     READ WELL PACKAGE INPUT:  NWELLS,WELL
C
      INTEGER OUNIT
      DIMENSION WELL(4,MXWELL)
      LOGICAL ABORT, IWLFLG
C
C1 -- INCREMENT WELL STRESS PERIOD, READ WELL FLAG/NUMBER OF WELLS
      IWELSP = IWELSP + 1
      READ (IN,8,ERR=98,END=99) ITMP
    8 FORMAT (I10)
C
C2 -- IF ITMP < 0 & NOT 1ST STRESS PERIOD, REUSE WELLS FROM LAST STRESS PERIOD
      IF (ITMP .LT. 0) THEN
         IF (IWLFLG) THEN
             WRITE (OUNIT,1)
    1        FORMAT (' WELLS must be read for the first stress period')
             STOP
         ENDIF
         IF (NOPRT .GE. 0) WRITE (OUNIT,6)
    6    FORMAT (10X,'Reusing WELLS from last stress period')
         RETURN
      ENDIF
C
C3 -- CHECK THAT NUMBER OF WELLS DOES NOT EXCEED MAXIMUM NUMBER OF WELLS
      NWELLS = ITMP
      IF (NWELLS .GT. MXWELL) THEN
         WRITE (OUNIT,11) NWELLS,MXWELL
   11    FORMAT (' NWELLS(',I4,') is greater than MXWELL(',I4,')')
         STOP
      ENDIF
```

```
C
C4 -- PRINT NUMBER OF WELLS; IF NONE THEN RETURN
      IF (NOPRT .GE. 0) WRITE (OUNIT,2) NWELLS
    2 FORMAT (///,50X,I5,' WELLS FOR CURRENT STRESS PERIOD')
      IF (NWELLS .EQ. 0) RETURN
C
C5 -- READ & PRINT WELL LOCATIONS & PUMPAGES, CHECK FOR OUT-OF-BOUNDS
      IF (NOPRT .GE. 0) WRITE (OUNIT,3)
    3 FORMAT (48X,'LAYER    ROW    COL    STRESS RATE   WELL NO.',/,
     1 48X,45('-'))
      ABORT = .FALSE.
      DO 250 II=1,NWELLS
         READ (IN,4,ERR=98,END=99) K,I,J,Q
    4    FORMAT (3I10,F10.0)
         IF (NOPRT .GE. 0) WRITE (OUNIT,5) K,I,J,Q,II
    5    FORMAT (48X,I3,I8,I7,G16.5,I8)
         IF (K .LE. 0 .OR. K .GT. NLAY .OR.
     1       I .LE. 0 .OR. I .GT. NROW .OR.
     2       J .LE. 0 .OR. J .GT. NCOL) THEN
            WRITE (OUNIT,10)
   10       FORMAT (48X,' Preceding WELL is outside model limits')
            ABORT = .TRUE.
         ENDIF
         WELL (1,II) = K
         WELL (2,II) = I
         WELL (3,II) = J
         WELL (4,II) = Q
  250 CONTINUE
      IF (ABORT) STOP
      RETURN
C
   98 WRITE (OUNIT,998) IN
  998 FORMAT (' WELDRP:  Error reading from unit:  ',I2)
      STOP
   99 WRITE (OUNIT,999) IN
  999 FORMAT (' WELDRP:  Unexpected E-O-F on unit:  ',I2)
      STOP
      END

      SUBROUTINE DRNDRP (IN,OUNIT)
C
C     READ & IGNORE DRAIN PACKAGE INPUT:  K,I,J,DRAI
C
      INTEGER OUNIT
C
      READ (IN,8,ERR=98,END=99) ITMP
    8 FORMAT (I10)
C
      IF (ITMP .LE. 0) RETURN
C
      DO 250 I=1,ITMP
      READ (IN,4,ERR=98,END=99) IPAD1,IPAD2,IPAD3,PAD1,PAD2
    4 FORMAT (3I10,2F10.0)
  250 CONTINUE
      RETURN
C
   98 WRITE (OUNIT,2) IN
    2 FORMAT (' DRNDRP:  Error reading from unit:  ',I2)
      STOP
   99 WRITE (OUNIT,3) IN
    3 FORMAT (' DRNDRP:  Unexpected E-O-F on unit:  ',I2)
      STOP
      END
      SUBROUTINE RCHDRP (NRCHOP,IRCH,RECH,NROW,NCOL,NLAY,IN,IRCHSP,
     1  OUNIT,IRCFLG,NOPRT)
C
C     READ RECHARGE PACKAGE INPUT:  INRECH,INIRCH,RECH,IRCH
C
      INTEGER OUNIT
      DIMENSION IRCH(NCOL,NROW),RECH(NCOL,NROW),ANAME(6,2)
      LOGICAL IRCFLG, ABORT
C
      DATA (ANAME(I,1),I=1,6)
     1 /'    ','RECH','ARGE',' LAY','ER I','NDEX'/
      DATA (ANAME(I,2),I=1,6)
     1 /'    ','    ','    ','    ','RECH','ARGE'/
C
C1 -- INCREMENT THE RECHARGE STRESS PERIOD AND READ RECHARGE INPUT FLAGS
      IRCHSP = IRCHSP + 1
      READ (IN,4,ERR=98,END=99) INRECH,INIRCH
    4 FORMAT (2I10)
C
C2 -- READ RECHARGE ARRAY IF READ FLAG IS SET
      IF (INRECH .LT. 0) THEN
         IF (IRCFLG) THEN
            WRITE (OUNIT,1)
    1       FORMAT (' IRCH must be read for 1st stress period')
            STOP
         ENDIF
         IF (NOPRT .GE. 0) WRITE (OUNIT,3)
    3    FORMAT (' Reusing recharge rates from last stress period')
      ELSE
         CALL U2DREL (RECH,ANAME(1,2),NROW,NCOL,0,IN,OUNIT,NOPRT)
      ENDIF
C
C3 -- IF RECHARGE IS TO TOP LAYER, THEN RETURN
      IF (NRCHOP .NE. 2) RETURN
C
C4 -- READ RECHARGE INDICATOR ARRAY & CHECK BOUNDS OF EACH ELEMENT
      IF (INIRCH .GE. 0) THEN
         CALL U2DINT (IRCH,ANAME(1,1),NROW,NCOL,0,IN,OUNIT,NOPRT)
         ABORT = .FALSE.
         DO 10 I=1,NROW
         DO 10 J=1,NCOL
            IF (IRCH (J,I) .GT. NLAY .OR. IRCH (J,I) .LE. 0) THEN
               WRITE (OUNIT,5) I,J,IRCH(J,I)
    5          FORMAT (' At row',I3,' column',I3,I3,' is an illegal',
     1         ' layer index')
               ABORT = .TRUE.
            ENDIF
   10    CONTINUE
         IF (ABORT) STOP
      ELSE
         IF (NOPRT .GE. 0) WRITE (OUNIT,2)
    2    FORMAT (' Reusing rech layer index from last stress period')
      ENDIF
      RETURN
C
   98 WRITE (OUNIT,6) IN
    6 FORMAT (' RCHDRP:  Error reading from unit:  ',I2)
      STOP
   99 WRITE (OUNIT,7) IN
    7 FORMAT (' RCHDRP:  Unexpected E-O-F on unit:  ',I2)
      STOP
      END
      SUBROUTINE EVTDRP (NCOL,NROW,IN,OUNIT,DUMMY,NOPRT)
C
C     READ & IGNORE EVT PACKAGE INPUT:  INSURF,INEVTR,INEXDP,INIEVT
C      SURF,EVTR,EXDP,IEVT
C
      INTEGER OUNIT
      DIMENSION DUMMY(NCOL,NROW), ANAME(6,4)
C
      DATA (ANAME (I,1),I=1,6)
     1 /'    ','    ','  ET',' LAY','ER I','NDEX'/
      DATA (ANAME (I,2),I=1,6)
     1 /'    ','    ','    ','  ET',' SUR','FACE'/
      DATA (ANAME (I,3),I=1,6)
     1 /' EVA','POTR','ANSP','IRAT','ION ','RATE'/
      DATA (ANAME (I,4),I=1,6)
     1 /'    ','    ','EXTI','NCTI','ON D','EPTH'/
C
      READ (IN,6,ERR=98,END=99) INSURF,INEVTR,INEXDP,INIEVT
    6 FORMAT (4I10)
C
      IF (INSURF .GE. 0)
     1 CALL U2DREL (DUMMY,ANAME(1,2),NROW,NCOL,0,IN,OUNIT,NOPRT)
C
      IF (INEVTR .GE. 0)
     1 CALL U2DREL (DUMMY,ANAME(1,3),NROW,NCOL,0,IN,OUNIT,NOPRT)
C
      IF (INEXDP .GE. 0)
     1 CALL U2DREL (DUMMY,ANAME(1,4),NROW,NCOL,0,IN,OUNIT,NOPRT)
C
      IF (NEVTOP .EQ. 2 .AND. INIEVT .GE. 0)
     1 CALL U2DREL (DUMMY,ANAME(1,1),NROW,NCOL,0,IN,OUNIT,NOPRT)
C
      RETURN
C
   98 WRITE (OUNIT,2) IN
    2 FORMAT (' EVTDRP:  Error reading from unit:  ',I2)
      STOP
   99 WRITE (OUNIT,3) IN
    3 FORMAT (' EVTDRP:  Unexpected E-O-F on unit:  ',I2)
      STOP
      END
      SUBROUTINE RIVDRP (IN,OUNIT)
C
C     READ & IGNORE RIVER PACKAGE INPUT:  K,I,J,RIVR
C
      INTEGER OUNIT
C1 -- READ RIVER-READ-FLAG / NUMBER OF REACHES
      READ (IN,8,ERR=98,END=99) ITMP
    8 FORMAT (I10)
C
C2 -- IF NOT REUSING RIVERS, THEN WE MUST READ THEM
      IF (ITMP .LE. 0) RETURN
      DO 250 I=1,ITMP
      READ (IN,4,ERR=98,END=99) IPAD1,IPAD2,IPAD3,PAD1,PAD2,PAD3
    4 FORMAT (3I10,3F10.0)
  250 CONTINUE
      RETURN
C
   98 WRITE (OUNIT,2) IN
    2 FORMAT (' RIVDRP:  Error reading from unit:  ',I2)
      STOP
   99 WRITE (OUNIT,3) IN
    3 FORMAT (' RIVDRP:  Unexpected E-O-F on unit:  ',I2)
      STOP
      END

      SUBROUTINE GHBDRP (IN,OUNIT)
C
C     READ & IGNORE GHB PACKAGE INPUT:  K,I,J,BNDS
C
      INTEGER OUNIT
C
      READ (IN,8,ERR=98,END=99) ITMP
    8 FORMAT (I10)
C
      IF (ITMP .LE. 0) RETURN
C
      DO 250 I=1,ITMP
      READ (IN,4,ERR=98,END=99) IPAD1,IPAD2,IPAD3,PAD1,PAD2
    4 FORMAT (3I10,2F10.0)
  250 CONTINUE
      RETURN
C
   98 WRITE (OUNIT,2) IN
    2 FORMAT (' GHBDRP:  Error reading from unit:  ',I2)
      STOP
   99 WRITE (OUNIT,3) IN
    3 FORMAT (' GHBDRP:  Unexpected E-O-F on unit:  ',I2)
      STOP
      END
      SUBROUTINE CTL1CM (LINE,NROW,NCOL,NLAY,INCTL,OUNIT,EXCMD,EOF,UBIN,
     1  U2DDSN,U3DDSN,U2DANM,U3DANM,ISKSP,ISKTS,LCSTRT,LCIBOU,LCSC1,
     2  LCTOP,LCIOFG,LCBOT,LCDELR,LCDELC,LCWELL,NWELLS,MXWELL,NRCHOP,
     3  LCIRCH,LCRECH,LCU2DS,LCU3DS,LCWDS,IUNIT,NUCLAS,CLASES,TITLES,
     4  IBOUND,UBOUND,DS1,DS2,STKA2D,STKA3D,NPER,NTS,KSP,KTS,IOFLG,
     5  LCUBOU,LCDELL,DELR,DELC,DELL,LCGRAF,LNGRAF,IBCFCB,IWELCB,IDRNCB,
     6  IRCHCB,IEVTCB,IRIVCB,IGHBCB,IHEDUN,IDDNUN,IPKGSP,PLANE,NOPRT)
C
C     PROCESS A SINGLE COMMAND
C
      DOUBLE PRECISION TEXTEX, DS1, DS2, STKA2D, STKA3D
      DIMENSION DS1(NCOL*NROW*NLAY),DS2(NCOL*NROW*NLAY),CLASES(20)
      DIMENSION STKA2D(NCOL,NROW),STKA3D(NCOL,NROW,NLAY)
      DIMENSION PLANE (4), DELL (NLAY)
      INTEGER IBOUND(NCOL,NROW,NLAY),UBOUND(NCOL,NROW,NLAY),IPKGSP(12)
$INSERT STKSIZE.INS
$INSERT STKDEF.INS
$INSERT TINY.INS
      INTEGER IUNIT(24),OUNIT,LCWDS(2),IOFLG(NLAY,2),NTS(NPER)
      INTEGER COORD (3,3)
      LOGICAL EXCMD,EOF,UBIN
      CHARACTER TITLES(4)*128,CULAY*2,CMASK*6,OPRATR*2,DSN3*6,ANAME3*24
      CHARACTER DSN*6,DSN1*6,DSN2*6,LINE*80,ANAME1*24,ANAME2*24,CMD*4
      CHARACTER DSTYPE*3,ORIENT*5,MISSTR*30
      EXTERNAL UTRMUP
      CHARACTER UTRMUP*6
C
C1 -- READ A LINE FROM THE COMMAND FILE
      READ (INCTL,9001,ERR=998,END=10000) LINE
 9001 FORMAT (A80)
C
C2 -- ENTRY POINT FOR DEFAULT PROCESSING
      ENTRY DFLTEN (LINE,NROW,NCOL,NLAY,INCTL,OUNIT,EXCMD,EOF,UBIN,
     1  U2DDSN,U3DDSN,U2DANM,U3DANM,ISKSP,ISKTS,LCSTRT,LCIBOU,LCSC1,
     2  LCTOP,LCIOFG,LCBOT,LCDELR,LCDELC,LCWELL,NWELLS,MXWELL,NRCHOP,
     3  LCIRCH,LCRECH,LCU2DS,LCU3DS,LCWDS,IUNIT,NUCLAS,CLASES,TITLES,
     4  IBOUND,UBOUND,DS1,DS2,STKA2D,STKA3D,NPER,NTS,KSP,KTS,IOFLG,
     5  LCUBOU,LCDELL,DELR,DELC,DELL,LCGRAF,LNGRAF,IBCFCB,IWELCB,IDRNCB,
     6  IRCHCB,IEVTCB,IRIVCB,IGHBCB,IHEDUN,IDDNUN,IPKGSP,PLANE,NOPRT)
C
C3 -- DEFINE VARIABLES APPLICABLE TO ALL COMMANDS
      EXCMD = .TRUE.
      NCR  = NCOL * NROW
      NCRL = NCOL * NROW * NLAY
      CMD =  UTRMUP (LINE(1:6),4)
      IF (CMD .NE. 'TITL' .AND. CMD .NE. '****')
     1    WRITE (OUNIT,9002) (TITLES(I),I=1,4), LINE
 9002 FORMAT (1H1,A128,/,2(1X,A128,/),/,1X,A128,//,' Processing:   ',
     1        A80,//)
C
C4 -- BRANCH TO APPROPRIATE COMMAND PROCESSOR
C
C4A -- 'STAT'ISTICS COMMAND
      IF (CMD .EQ. 'STAT') THEN
C4A1 -- GET COMMAND ARGUMENTS:  DATA SET NAME, LAYER, & MASKS
         DSN = UTRMUP (LINE(6:11),6)
         CULAY = LINE (13:14)
         CMASK = LINE (16:21)
         READ (CULAY,9003,ERR=996) IULAY1
 9003    FORMAT (I2)
         READ (CMASK,9004,ERR=997) IZMASK,IBMASK,IUMASK
 9004    FORMAT (3I2)
         MASKUM = IABS(IZMASK) + IABS(IBMASK) + IABS(IUMASK)
```

```
C
C4A2 -- GET THE REQUESTED DATA ARRAY
        CALL ULC1DS (NROW,NCOL,NLAY,DSN,IUNIT,OUNIT,IULAY1,DS1,LCDS1,
     1   LENDS,ANAME1,EXCMD,U2DDSN,U3DDSN,LCU2DS,LCU3DS,U2DANM,
     2   LCRECH,LCIRCH,LCDELC,LCDELR,NRCHOP,LCWELL,MXWELL,NWELLS,
     3   U3DANM,LCIBOU,LCSTRT,LCSC1,LCBOT,LCTOP,LCIOFG,.TRUE.,
     4   IPKGSP,ISKSP,ISKTS,ISP1,ITS1,NPER,NTS)
        IF (.NOT. EXCMD) GOTO 999
C
C4A3 -- PRINT HEADINGS
        CALL SCTL1H (CMD,DSN,DSN2,DSN2,ANAME1,ANAME2,ANAME2,IULAY1,
     1   IULAY2,ISP1,ISP1,ITS1,ITS1,'   ',OUNIT)
C
C4A4 -- LOAD NODE POINTERS INTO DS2
        CALL STA1SL (DS2,NLAY,NCR,NCRL,DSN,IOFLG)
C
C4A5 -- IF REQUESTED, MASK THE DATA ARRAY
        IF (MASKUM .GT. 0) CALL MAS1EX (LENDS,NCRL,DS1,IULAY1,NLAY,
     1   DS2,EXCMD,UBIN,LCDS1,IBMASK,IZMASK,IUMASK,OUNIT,DSN,IBOUND,
     2   UBOUND)
        IF (.NOT. EXCMD) GOTO 999
C
C4A6 -- CALCULATE AND PRINT DESCRIPTIVE STATISTICS
        LCDS2 = LCDS1 / 2
        IF (MOD (LCDS1,2) .NE. 0) LCDS2 = LCDS2 + 1
        IJUMP = 1
        IF (MASKUM .EQ. 0 .AND. MOD (IULAY1,2) .EQ. 0 .AND.
     1      IULAY1 .GT. 1 .AND. MOD (NCR    ,2) .EQ. 1) IJUMP = 2
        CALL STA1EX (NCOL,NROW,NLAY,LENDS,DS1(LCDS1),DS2(LCDS2),
     1   IJUMP,OUNIT,EXCMD)
        IF (.NOT. EXCMD) GOTO 999
C
C4B -- 'HIST'OGRAM COMMAND
      ELSEIF (CMD .EQ. 'HIST') THEN
C4B1 -- GET COMMAND ARGUMENTS:  DSN, LAYER, MASK, CLASSES
        DSN = UTRMUP (LINE(6:11),6)
        CULAY = LINE (13:14)
        CMASK = LINE (16:21)
        READ (CULAY,9003,ERR=996) IULAY1
        READ (CMASK,9004,ERR=997) IZMASK,IBMASK,IUMASK
        MASKUM = IABS(IZMASK) + IABS(IBMASK) + IABS(IUMASK)
        READ (LINE,9007,ERR=995) IUCLAS
 9007   FORMAT (22X,I3)
C
C4B2 -- GET THE REQUESTED DATA ARRAY
        CALL ULC1DS (NROW,NCOL,NLAY,DSN,IUNIT,OUNIT,IULAY1,DS1,LCDS1,
     1   LENDS,ANAME1,EXCMD,U2DDSN,U3DDSN,LCU2DS,LCU3DS,U2DANM,
     2   LCRECH,LCIRCH,LCDELC,LCDELR,NRCHOP,LCWELL,MXWELL,NWELLS,
     3   U3DANM,LCIBOU,LCSTRT,LCSC1,LCBOT,LCTOP,LCIOFG,.TRUE.,
     4   IPKGSP,ISKSP,ISKTS,ISP1,ITS1,NPER,NTS)
        IF (.NOT. EXCMD) GOTO 999
C
C4B3 -- PRINT HEADINGS
        CALL SCTL1H (CMD,DSN,DSN2,DSN2,ANAME1,ANAME2,ANAME2,IULAY1,
     1   IULAY2,ISP1,ISP1,ITS1,ITS1,'   ',OUNIT)
C
C4B4 -- IF REQUESTED, MASK THE DATA ARRAY
        IF (MASKUM .GT. 0) CALL MAS1EX (LENDS,NCRL,DS1,IULAY1,NLAY,
     1   DS2,EXCMD,UBIN,LCDS1,IBMASK,IZMASK,IUMASK,OUNIT,DSN,IBOUND,
     2   UBOUND)
        IF (.NOT. EXCMD) GOTO 999
```

```
C
C4B5 -- CALCULATE AND PRINT FREQUENCY ANALYSIS
         CALL HIS1EX (DS1(LCDS1),DS2,LENDS,NUCLAS,CLASES,IUCLAS,OUNIT,
     1     EXCMD)
         IF (.NOT. EXCMD) GOTO 999
         CALL SCTL1H (CMD,DSN,DSN2,DSN2,ANAME1,ANAME2,ANAME2,IULAY1,
     1    IULAY2,ISP1,ISP1,ITS1,ITS1,'  ',OUNIT)
C
C4C -- 'COMP'ARISON COMMAND
      ELSEIF (CMD .EQ. 'COMP') THEN
C4C1 -- GET COMMAND ARGUEMENTS:  DSN1, LAYER1, MASK, OPERATOR,
C                                DSN2, LAYER2, LIMIT
         DSN = UTRMUP (LINE(6:11),6)
         DSN1 = DSN
         CULAY = LINE (13:14)
         CMASK = LINE (29:34)
         READ (CULAY,9003,ERR=996) IULAY1
         READ (CMASK,9004,ERR=997) IZMASK,IBMASK,IUMASK
         OPRATR = UTRMUP (LINE(16:21),2)
         DSN = UTRMUP (LINE (19:24),6)
         DSN2 = DSN
         CULAY = LINE (26:27)
         READ (CULAY,9003) IULAY2
         READ (LINE(36:39),9020,ERR=894) LIMIT
 9020    FORMAT (I4)
C
C4C2 -- GET THE DATA ARRAY REQUESTED FOR DSN1
         CALL ULC1DS (NROW,NCOL,NLAY,DSN1,IUNIT,OUNIT,IULAY1,DS1,LCDS1,
     1    LENDS1,ANAME1,EXCMD,U2DDSN,U3DDSN,LCU2DS,LCU3DS,U2DANM,
     2    LCRECH,LCIRCH,LCDELC,LCDELR,NRCHOP,LCWELL,MXWELL,NWELLS,
     3    U3DANM,LCIBOU,LCSTRT,LCSC1,LCBOT,LCTOP,LCIOFG,.TRUE.,
     4    IPKGSP,ISKSP,ISKTS,ISP1,ITS1,NPER,NTS)
         IF (.NOT. EXCMD) GOTO 999
         IF ((DSN1 .EQ. 'HEAD' .OR. DSN1 .EQ. 'DRAWDN' .OR.
     1    DSN1 .EQ. 'TOP'  .OR. DSN1 .EQ. 'BOT') .AND. IULAY1 .EQ. 0)
     2    LENDS1 = NCRL
C
C4C3 -- GET THE DATA ARRAY REQUESTED FOR DSN2
         CALL ULC1DS (NROW,NCOL,NLAY,DSN2,IUNIT,OUNIT,IULAY2,DS2,LCDS2,
     1    LENDS2,ANAME2,EXCMD,U2DDSN,U3DDSN,LCU2DS,LCU3DS,U2DANM,
     2    LCRECH,LCIRCH,LCDELC,LCDELR,NRCHOP,LCWELL,MXWELL,NWELLS,
     3    U3DANM,LCIBOU,LCSTRT,LCSC1,LCBOT,LCTOP,LCIOFG,.FALSE.,
     4    IPKGSP,ISKSP,ISKTS,ISP2,ITS2,NPER,NTS)
         IF (.NOT. EXCMD) GOTO 999
         IF ((DSN2 .EQ. 'HEAD' .OR. DSN2 .EQ. 'DRAWDN' .OR.
     1    DSN2 .EQ. 'TOP'  .OR. DSN2 .EQ. 'BOT') .AND. IULAY2 .EQ. 0)
     2    LENDS2 = NCRL
C
C4C4 -- PRINT HEADINGS
         CALL SCTL1H (CMD,DSN1,DSN2,DSN3,ANAME1,ANAME2,ANAME3,IULAY1,
     1    IULAY2,ISP1,ISP2,ITS1,ITS2,OPRATR,OUNIT)
C
C4C5 -- CHECK IF NODE-BY-NODE COMPARISON IS POSSIBLE
         IF (LENDS1 .NE. LENDS2) THEN
            EXCMD = .FALSE.
            WRITE (OUNIT,5) DSN1,LENDS1,DSN2,LENDS2
    5       FORMAT (/,' Unable to compare, data sets not same size',/,
     1       ' Length of ',A6,' = ',I9,/,' Length of ',A6,' = ',I9)
            GOTO 999
         ENDIF
```

```
C
C4C6 -- PERFORM THE COMPARISON
         CALL COM1EX (EXCMD,NCOL,NROW,NLAY,LENDS1,OUNIT,DSN1,DS1,IULAY1,
     1    DSN2,DS2,IULAY2,OPRATR,IBOUND,UBOUND,IZMASK,IBMASK,IUMASK,
     2    UBIN,IOFLG,LIMIT)
         IF (.NOT. EXCMD) GOTO 999
C
C4D -- 'MATH'EMATICS COMMAND
      ELSEIF (CMD .EQ. 'MATH') THEN
C4D1 -- GET THE COMMAND ARGUMENTS:  DSN1, LAYER1, OPERATOR, DSN2, LAYER2,
C                                   DSN3, ARRAY-NAME
         DSN = UTRMUP (LINE(6:11),6)
         DSN1 = DSN
         CULAY = LINE (13:14)
         READ (CULAY,9003,ERR=996) IULAY1
         OPRATR = UTRMUP (LINE(16:21),2)
         DSN = UTRMUP (LINE (19:24),6)
         DSN2 = DSN
         CULAY = LINE (26:27)
         READ (CULAY,9003) IULAY2
         DSN3 = UTRMUP (LINE (29:34),6)
         ANAME3 = LINE (36:59)
C
C4D2 -- GET THE DATA ARRAY REQUESTED FOR DSN1
         CALL ULC1DS (NROW,NCOL,NLAY,DSN1,IUNIT,OUNIT,IULAY1,DS1,LCDS1,
     1    LENDS1,ANAME1,EXCMD,U2DDSN,U3DDSN,LCU2DS,LCU3DS,U2DANM,
     2    LCRECH,LCIRCH,LCDELC,LCDELR,NRCHOP,LCWELL,MXWELL,NWELLS,
     3    U3DANM,LCIBOU,LCSTRT,LCSC1,LCBOT,LCTOP,LCIOFG,.TRUE.,
     4    IPKGSP,ISKSP,ISKTS,ISP1,ITS1,NPER,NTS)
         IF (.NOT. EXCMD) GOTO 999
         IF ((DSN1 .EQ. 'HEAD' .OR. DSN1 .EQ. 'DRAWDN' .OR.
     1    DSN1 .EQ. 'TOP'  .OR. DSN1 .EQ. 'BOT') .AND. IULAY1 .EQ. 0)
     2    LENDS1 = NCRL
C
C4D3 -- GET THE DATA ARRAY REQUESTED FOR DSN2
         IF (OPRATR .NE. 'AB' .AND. OPRATR .NE. '||')
     1    CALL ULC1DS (NROW,NCOL,NLAY,DSN2,IUNIT,OUNIT,IULAY2,DS2,LCDS2,
     2    LENDS2,ANAME2,EXCMD,U2DDSN,U3DDSN,LCU2DS,LCU3DS,U2DANM,
     3    LCRECH,LCIRCH,LCDELC,LCDELR,NRCHOP,LCWELL,MXWELL,NWELLS,
     4    U3DANM,LCIBOU,LCSTRT,LCSC1,LCBOT,LCTOP,LCIOFG,.FALSE.,
     5    IPKGSP,ISKSP,ISKTS,ISP2,ITS2,NPER,NTS)
         IF (.NOT. EXCMD) GOTO 999
         IF ((DSN2 .EQ. 'HEAD' .OR. DSN2 .EQ. 'DRAWDN' .OR.
     1    DSN2 .EQ. 'TOP'  .OR. DSN2 .EQ. 'BOT') .AND. IULAY2 .EQ. 0)
     2    LENDS2 = NCRL
C
C4D4 -- CHECK IF THE REQUESTED COMPUTATION IS POSSIBLE
         IF (LENDS1 .NE. LENDS2 .AND.
     1       OPRATR .NE. '||'   .AND. OPRATR .NE. 'AB') THEN
            EXCMD = .FALSE.
            WRITE (OUNIT,6) DSN1,LENDS1,DSN2,LENDS2
    6       FORMAT (/,' Unable to compute, data sets not same size',/,
     1       ' Length of ',A6,' = ',I9,/,' Length of ',A6,' = ',I9)
            GOTO 999
         ENDIF
C
C4D5 -- IF POSSIBLE, ASSIGN STRESS PERIOD AND TIME STEP TO THE CALCULATED ARRAY
         IF (ISP1 .EQ. ISP2 .AND. ITS1 .EQ. ITS2) THEN
            ISP3 = ISP1
            ITS3 = ITS1
         ELSE
            ISP3 = 0
            ITS3 = 0
         ENDIF
```

```
C
C4D6 -- PRINT HEADINGS
         CALL SCTL1H (CMD,DSN1,DSN2,DSN3,ANAME1,ANAME2,ANAME3,IULAY1,
     1     IULAY2,ISP1,ISP2,ITS1,ITS2,OPRATR,OUNIT)
C
C4D7 -- CALCULATE A TWO-DIMENSIONAL DATA ARRAY
         IF (LENDS1 .EQ. NCR) THEN
C
C4D7A -- PERFORM THE CALCULATION
            CALL MTH1EX (NCOL,NROW,NLAY,DSN1,DSN2,IULAY1,IULAY2,
     1        LCDS1,LCDS2,DS1,DS2,STKA2D,IOFLG,OPRATR,OUNIT,LENDS1,
     2        EXCMD)
            IF (.NOT. EXCMD) GOTO 999
C
C4D7B -- ALTER THE STACK DESCRIPTORS
            CALL UBUBLE (LCU2DS,U2DDSN,U2DANM,ISKSP,ISKTS,CMD,
     1        .FALSE.,OUNIT,DSN3,ANAME3,ISP3,ITS3)
C
C4D8 -- CALCULATE A THREE-DIMENSIONAL DATA ARRAY
         ELSEIF (LENDS1 .EQ. NCRL) THEN
C
C4D8A -- PERFORM THE CALCULATION
            CALL MTH1EX (NCOL,NROW,NLAY,DSN1,DSN2,IULAY1,IULAY2,
     1        LCDS1,LCDS2,DS1,DS2,STKA3D,IOFLG,OPRATR,OUNIT,LENDS1,
     2        EXCMD)
            IF (.NOT. EXCMD) GOTO 999
C
C4D8B -- ALTER THE STACK DESCRIPTORS
            CALL UBUBLE (LCU3DS,U3DDSN,U3DANM,ISKSP,ISKTS,CMD,
     1        .TRUE.,OUNIT,DSN3,ANAME3,ISP3,ITS3)
         ENDIF
C
C4E -- READ COMMAND
      ELSEIF (CMD .EQ. 'READ') THEN
C4E1 -- GET THE COMMAND DATA SET NAME, INITIALIZE STRESS-PERIOD & TIME-STEP
         DSN    = UTRMUP (LINE(6:11),6)
         ANAME1 = LINE (26:49)
         IUSP = 0
         IUTS = 0
C
C4E2 -- IF DSN IS 'CLASS', IT MUST BE HANDLED DIFFERENTLY
         IF (DSN .EQ. 'CLASS') THEN
            CALL REA1CL (LINE,NUCLAS,CLASES,OUNIT,EXCMD)
            IF (.NOT. EXCMD) GOTO 999
            RETURN
         ENDIF
C
C4E3 -- IS THE DSN ONE OF RESERVED DSNS, IF SO WHAT IS:
C                   THE UNIT NUMBER, FORMAT, AND ARRAY-NAME
         CALL REA1DF (DSN,INUNIT,DSTYPE,ANAME1,OUNIT,
     1     IBCFCB,IWELCB,IDRNCB,IRIVCB,IEVTCB,IGHBCB,IRCHCB,IHEDUN,
     1     IDDNUN,IUNIT(2),IUNIT(8),TEXTEX)
C
C4E4 -- IF NEEDED, GET UNIT NUMBER AND FORMAT FROM COMMAND LINE
         IF (DSN .EQ. 'UBOUND') THEN
           READ (LINE,9011,ERR=993) INUNIT
           DSTYPE = '3FI'
 9011      FORMAT (12X,I2)
         ELSEIF (DSTYPE .EQ. 'XXX') THEN
           READ (LINE,9011,ERR=993) INUNIT
           DSN2          = UTRMUP (LINE (16:18),3)
           DSTYPE        = DSN2(1:1) // 'F' // DSN2(3:3)
           ANAME1        = LINE (26:49)
         ELSE
```

```
C
C4E5 -- GET COMMAND STRESS PERIOD AND TIME STEP AND CHECK ITS BOUNDS
            ORIENT = LINE(20:24)
            READ (ORIENT,9008,ERR=994) IUSP,IUTS
 9008       FORMAT (I2,1X,I2)
            IF (IUSP .LT. 0 .OR. IUSP .GT. NPER) THEN
               EXCMD = .FALSE.
               WRITE (OUNIT,9009) DSN,IUSP,NPER
 9009          FORMAT (/,' READ command for ',A6,' Stress period',I3,
     1          ' is beyond the defined limit:',/,' Number of stress'
     2          ' periods =',I3)
               GOTO 999
            ENDIF
            IF (DSN .NE. 'WELL' .AND. DSN .NE. 'RECH') THEN
               IF (IUTS.LE.0 .OR. IUSP.LE.0 .OR. IUTS.GT.NTS(IUSP))THEN
                  EXCMD = .FALSE.
                  WRITE (OUNIT,9010) DSN,IUSP,IUTS
 9010             FORMAT (/,' READ command for ',A6,' Stress period',I3,
     1             ' Time step',I3,' is beyond the defined limit.')
                  IF (IUSP .GT. 0 .AND. IUSP .LE. NPER)
     1               WRITE (OUNIT,9012) NTS(IUSP)
 9012             FORMAT(' Number of time steps for stress period =',I3)
                  GOTO 999
               ENDIF
            ENDIF
C
C4E6 --  GET THE HEAD/DRAWDOWN OUTPUT FLAGS FOR THE STRESS-PERIOD & TIME-STEP
            IF (DSN .EQ. 'HEAD' .OR. DSN .EQ. 'DRAWDN') THEN
               CALL UIO1FG (NPER,NLAY,NTS,IUSP,IUTS,IOFLG,IPKGSP(12),
     1          IUNIT(12),OUNIT,EXCMD)
               IF (.NOT. EXCMD) GOTO 999
            ENDIF
         ENDIF
C
C4E7 -- PRINT HEADINGS
         CALL SCTL1H (CMD,DSN,DSN2,DSN3,ANAME1,ANAME2,ANAME3,INUNIT,
     1    IULAY2,IUSP,ISP2,IUTS,ITS2,'  ',OUNIT)
C
C4E8 -- READ THE REQUESTED DATA
         CALL REA1EX (DSN,DSTYPE,NCOL,NROW,NLAY,INUNIT,EXCMD,OUNIT,
     1    LCWELL,MXWELL,NWELLS,LCIRCH,LCRECH,NRCHOP,LCU2DS,LCU3DS,
     2    IOFLG,LCUBOU,DS1,DS2,IUSP,IUTS,TEXTEX,IPKGSP,UBIN,ANAME1,
     3    NOPRT)
         IF (.NOT. EXCMD) GOTO 999
C
C4E9 -- ALTER THE PROPER STACK DESCRIPTORS
         IF (DSTYPE .EQ. '2D') CALL UBUBLE (LCU2DS,U2DDSN,U2DANM,
     1    ISKSP,ISKTS,CMD,.FALSE.,OUNIT,DSN,ANAME1,IUSP,IUTS)
         IF (DSTYPE .EQ. '3D') CALL UBUBLE (LCU3DS,U3DDSN,U3DANM,
     1    ISKSP,ISKTS,CMD,.TRUE.,OUNIT,DSN,ANAME1,IUSP,IUTS)
C
C4F -- 'TITL'E COMMAND
      ELSEIF (CMD .EQ. 'TITL') THEN
         TITLES(4) = LINE (6:)
C
C4G -- '****' COMMENT LINE
      ELSEIF (CMD .EQ. '****') THEN
         RETURN
C
C4H -- 'SLIC'E COMMAND
      ELSEIF (CMD .EQ. 'SLIC') THEN
C4H1 -- GET THE COMMAND ARGUMENTS:  COORDINATES OF THE SLICING PLANE
         READ (LINE,9013,ERR=991) ((COORD(I,J),J=1,3),I=1,3)
 9013    FORMAT (4X,9(1X,I3))
```

```
C
C4H2 -- SLICE THE MODEL GRID TO CREATE A USER-BOUNDARY ARRAY
          CALL SLI1EX (NCOL,NROW,NLAY,COORD,PLANE,UBOUND,
     1     DELR,DELC,DELL,OUNIT,EXCMD)
          IF (.NOT. EXCMD) GOTO 999
          UBIN = .TRUE.
C
C4I -- 'THIC'KNESS COMMAND
       ELSEIF (CMD .EQ. 'THIC') THEN
          N = NLAY / 7
          IF (MOD (NLAY,7) .NE. 0) N = N + 1
          DO 115 J=1,N
             L = J * 7
             IF (L .GT. NLAY) L = NLAY
             READ (LINE,9015,ERR=892) (DELL(I),I=1+(J-1)*7,L)
 9015        FORMAT (5X,7F10.0)
             IF (J .NE. N) READ (INCTL,9001) LINE
  115     CONTINUE
          WRITE (OUNIT,9016) (I,DELL(I),I=1,NLAY)
 9016     FORMAT (//,10X,'LAYER     LAYER',/,10X,'NUMBER  THICKNESS',
     *            /,10X,17('-'),999(/,10X,I4,G12.3E2))
C
C4J -- 'VECT'OR COMMAND
       ELSEIF (CMD .EQ. 'VECT') THEN
C
C4J1 -- CHECK THAT USER-BOUNDARY AND PLANE HAVE BEEN DEFINED
          HOLD = 0.0
          DO 120 I=1,3
120          HOLD = HOLD + ABS (PLANE(I))
          IF ((.NOT. UBIN) .OR. (HOLD .LE. TINY)) GOTO 890
C
C4J2 -- GET THE COMMAND ARGUMENTS:  UNIT NUMBER, ORIENTATION, AND FACTOR
          READ (LINE,14,ERR=893) IGUNIT, GRFACT
   14     FORMAT (5X,I2,7X,F10.0)
          ORIENT = UTRMUP (LINE(9:13),5)
C
C4J3 -- MAKE SURE THAT BCF CELL-FACE FLOW-TERMS ARE ON THE STACK
          ANAME2 = 'FRFACE'
          CALL USKDUD (NROW,NCOL,NLAY,ANAME2,U2DDSN,U3DDSN,U2DANM,
     1     U3DANM,ISKSP,ISKTS,LCU2DS,LCU3DS,ISKPOS,IOFLG,IULAY,LCDS,
     2     LENDS,ANAME1,ISP,ITS,.TRUE.,NPER,NTS,IPKGSP(12),IUNIT(12),
     3     .TRUE.)
          IF (ISKPOS .EQ. 0) GOTO 891
          LCFRFA = LCDS
          ANAME2 = 'LOFACE'
          CALL USKDUD (NROW,NCOL,NLAY,ANAME2,U2DDSN,U3DDSN,U2DANM,
     1     U3DANM,ISKSP,ISKTS,LCU2DS,LCU3DS,ISKPOS,IOFLG,IULAY,LCDS,
     2     LENDS,ANAME1,ISP,ITS,.TRUE.,NPER,NTS,IPKGSP(12),IUNIT(12),
     3     .TRUE.)
          IF (ISKPOS .EQ. 0) GOTO 891
          LCLOFA = LCDS
          ANAME2 = 'RIFACE'
          CALL USKDUD (NROW,NCOL,NLAY,ANAME2,U2DDSN,U3DDSN,U2DANM,
     1     U3DANM,ISKSP,ISKTS,LCU2DS,LCU3DS,ISKPOS,IOFLG,IULAY,LCDS,
     2     LENDS,ANAME1,ISP,ITS,.TRUE.,NPER,NTS,IPKGSP(12),IUNIT(12),
     3     .TRUE.)
          IF (ISKPOS .EQ. 0) GOTO 891
          LCRIFA = LCDS
C
C4J4 -- CALCULATE THE FLOW VECTORS
          CALL VEC1EX (NCOL,NROW,NLAY,LCLOFA,LCFRFA,LCRIFA,UBOUND,
     1     LCDELR,LCDELC,LCDELL,LCGRAF,LNGRAF,PLANE,ORIENT,GRFACT,
     2     IGUNIT,OUNIT,EXCMD)
          IF (.NOT. EXCMD) GOTO 999
```

```
C
C4K -- 'WRIT' COMMAND
      ELSEIF (CMD .EQ. 'WRIT') THEN
C
C4K1 -- GET COMMAND ARGUMENTS: DATA SET NAME, LAYER, MASK, UNIT, TYPE,
C           MISSING VALUE
         DSN = UTRMUP (LINE(6:11),6)
         CULAY = LINE (13:14)
         CMASK = LINE (16:21)
         READ (CULAY,9003,ERR=996) IULAY1
         READ (CMASK,9004,ERR=997) IZMASK,IBMASK,IUMASK
         MASKUM = IABS(IZMASK) + IABS(IBMASK) + IABS(IUMASK)
         READ (LINE,9017,ERR=993) INUNIT
 9017    FORMAT (22X,I2)
         DSTYPE = UTRMUP (LINE(26:28),3)
         DSN2   = DSTYPE
         MISSTR = LINE(30:59)
C
C4K2 -- GET THE REQUESTED DATA ARRAY
         CALL ULC1DS (NROW,NCOL,NLAY,DSN,IUNIT,OUNIT,IULAY1,DS1,LCDS1,
     1    LENDS,ANAME1,EXCMD,U2DDSN,U3DDSN,LCU2DS,LCU3DS,U2DANM,
     2    LCRECH,LCIRCH,LCDELC,LCDELR,NRCHOP,LCWELL,MXWELL,NWELLS,
     3    U3DANM,LCIBOU,LCSTRT,LCSC1,LCBOT,LCTOP,LCIOFG,.TRUE.,
     4    IPKGSP,ISKSP,ISKTS,ISP1,ITS1,NPER,NTS)
         IF (.NOT. EXCMD) GOTO 999
C
C4K3 -- PRINT HEADINGS
         CALL SCTL1H (CMD,DSN,DSN2,DSN2,ANAME1,ANAME1,ANAME1,IULAY1,
     1    INUNIT,ISP1,ISP1,ITS1,ITS1,'    ',OUNIT)
C
C4K4 -- REPLACE MASKED DATA WITH THE MISSING VALUES
         IF (MASKUM .GT. 0) CALL MAS1MV (LENDS,NCRL,NCR,DS1,IULAY1,NLAY,
     1    EXCMD,IOFLG,UBIN,LCDS1,IBMASK,IZMASK,IUMASK,OUNIT,DSN,IBOUND,
     2    UBOUND,MISSTR)
         IF (.NOT. EXCMD) GOTO 999
C
C4K5 -- WRITE THE REQUESTED DATA ARRAY
         CALL WRT1EX (NCOL,NROW,NLAY,DS1,INUNIT,DSTYPE,LENDS,
     1    IULAY1,ANAME1,ITS1,ISP1,OUNIT,EXCMD)
         IF (.NOT. EXCMD) GOTO 999
C
C4L -- 'PRIN' COMMAND
      ELSEIF (CMD .EQ. 'PRIN') THEN
C
C4L1 -- GET COMMAND ARGUMENTS: DATA SET NAME, LAYER, MASK, FORMAT CODE,
C           AND MISSING VALUE
         DSN = UTRMUP (LINE(6:11),6)
         CULAY = LINE (13:14)
         CMASK = LINE (16:21)
         READ (CULAY,9003,ERR=996) IULAY1
         READ (CMASK,9004,ERR=997) IZMASK,IBMASK,IUMASK
         MASKUM = IABS(IZMASK) + IABS(IBMASK) + IABS(IUMASK)
         DSTYPE = LINE (23:25)
         DSN2   = DSTYPE
         MISSTR = LINE (27:56)
C
C4L2 -- GET THE REQUESTED DATA ARRAY
         CALL ULC1DS (NROW,NCOL,NLAY,DSN,IUNIT,OUNIT,IULAY1,DS1,LCDS1,
     1    LENDS,ANAME1,EXCMD,U2DDSN,U3DDSN,LCU2DS,LCU3DS,U2DANM,
     2    LCRECH,LCIRCH,LCDELC,LCDELR,NRCHOP,LCWELL,MXWELL,NWELLS,
     3    U3DANM,LCIBOU,LCSTRT,LCSC1,LCBOT,LCTOP,LCIOFG,.TRUE.,
     4    IPKGSP,ISKSP,ISKTS,ISP1,ITS1,NPER,NTS)
         IF (.NOT. EXCMD) GOTO 999
```

```
C
C4L3 -- PRINT HEADINGS
         CALL SCTL1H (CMD,DSN,DSN2,DSN2,ANAME1,ANAME1,ANAME1,
     1    IULAY1,IFMTCD,ISP1,ISP1,ITS1,ITS1,'   ',OUNIT)
C
C4K4 -- REPLACE MASKED DATA WITH THE MISSING VALUES
         IF (MASKUM .GT. 0) CALL MAS1MV (LENDS,NCRL,NCR,DS1,IULAY1,NLAY,
     1    EXCMD,IOFLG,UBIN,LCDS1,IBMASK,IZMASK,IUMASK,OUNIT,DSN,IBOUND,
     2    UBOUND,MISSTR)
         IF (.NOT. EXCMD) GOTO 999
C
C4L5 -- WRITE THE REQUESTED DATA ARRAY
         CALL PRT1EX (NCOL,NROW,NLAY,NCRL,DS1,DS2,DSTYPE,ANAME1,IULAY1,
     1    LCDS1,LENDS,ISP1,ITS1,OUNIT,EXCMD)
         IF (.NOT. EXCMD) GOTO 999
C
C4M -- 'HEAD' COMMAND
      ELSEIF (CMD .EQ. 'HEAD') THEN
C
C4M1 --   GET THE FORTRAN UNIT NUMBER FROM THE COMMAND LINE
         READ (LINE,9018,ERR=993) IOUNIT
 9018    FORMAT (5X,I2)
C
C4M2 --   GET DESCRIPTORS FOR HEAD ARRAY
         CALL REA1DF ('HEAD    ',INUNIT,DSTYPE,ANAME1,OUNIT,
     1    IBCFCB,IWELCB,IDRNCB,IRIVCB,IEVTCB,IGHBCB,IRCHCB,IHEDUN,
     2    IDDNUN,IUNIT(2),IUNIT(8),TEXTEX)
C
C4M3 --   WRITE THE REQUESTED COMPUTED HEADS TO A FORTRAN UNIT
         CALL HEA1EX (NCOL,NROW,NLAY,IOFLG,INUNIT,OUNIT,IUNIT(12),
     1    TEXTEX,IPKGSP(12),NPER,NTS,IOUNIT,LINE(9:80),DS1,EXCMD)
C
C4N -- 'REBO' COMMAND
      ELSEIF (CMD .EQ. 'REBO') THEN
C
C4N1 --   GET STRESS-PERIOD AND TIME-STEP FROM COMMAND LINE
         ORIENT = LINE (6:10)
         READ (ORIENT,9008,ERR=994) IUSP,IUTS
C
C4N2 --   GET DESCRIPTORS FOR HEAD ARRAY
         CALL REA1DF ('HEAD    ',INUNIT,DSTYPE,ANAME1,OUNIT,
     1    IBCFCB,IWELCB,IDRNCB,IRIVCB,IEVTCB,IGHBCB,IRCHCB,IHEDUN,
     1    IDDNUN,IUNIT(2),IUNIT(8),TEXTEX)
C
C4N3 --   RESET THE MODEL BOUNDARY ARRAY
         CALL REB1EX (NCOL,NROW,NLAY,IOFLG,INUNIT,OUNIT,IUNIT(12),
     1    TEXTEX,IPKGSP(12),NPER,NTS,IUSP,IUTS,IBOUND,DS1,EXCMD)
C
C5 -- COMMAND NOT DEFINED IN CTL1CM
      ELSE
         EXCMD = .FALSE.
         WRITE (OUNIT,990) CMD
  990    FORMAT (/,' Unknown command:  ',A4,' refer to table 4')
         GOTO 999
      ENDIF
      RETURN
```

```
C
C6 -- ERROR CONDITIONS...
  890 WRITE (OUNIT,9890)
 9890 FORMAT (/,' Vectors requested but no slice has been defined')
      GOTO 999
  891 WRITE (OUNIT,9891) ANAME2
 9891 FORMAT (/,' Vectors requested but ',A,' has not been read')
      GOTO 999
  892 WRITE (OUNIT,9892)
 9892 FORMAT (/,' Some layer thickness is non-numeric')
      GOTO 999
  893 WRITE (OUNIT,9893)
 9893 FORMAT (/,' Graphic output unit or vector factor is non-numeric')
      GOTO 999
  894 WRITE (OUNIT,9894)
 9894 FORMAT (/,' Comparison limit is non-numeric')
      GOTO 999
  991 WRITE (OUNIT,9991) LINE(6:80)
 9991 FORMAT (/,' Some slicing coordinate is non-numeric:  ',A)
      GOTO 999
  993 WRITE (OUNIT,9993) DSN,LINE(13:14)
 9993 FORMAT (/,' File unit specified for ',A6,' not an integer:  ',A2)
      GOTO 999
  994 WRITE (OUNIT,9994) ORIENT
 9994 FORMAT (/,' Stress period and time step not integers: ',A)
      GOTO 999
  995 WRITE (OUNIT,9995) DSN,LINE(23:25)
 9995 FORMAT (/,' Number of histogram classes for ',A6,' not an ',
     1          'integer:  ',A3)
      GOTO 999
  996 WRITE (OUNIT,9996) DSN,CULAY
 9996 FORMAT (/,' Layer specified for ',A6,' not an integer: ',A2)
      GOTO 999
  997 WRITE (OUNIT,9997) DSN,CMASK
 9997 FORMAT (/,' Mask specifiers for ',A6,' not all integers: ',A6)
      GOTO 999
  998 WRITE (OUNIT,9998)
 9998 FORMAT (/,' Unable to read command line')
      GOTO 10000
  999 WRITE (OUNIT,9999) CMD
 9999 FORMAT (///,1X,A4,' COMMAND ABORTED')
      EXCMD = .FALSE.
      RETURN
C
C7 -- SET END-OF-FILE FLAG
10000 EOF = .TRUE.
      RETURN
      END
      SUBROUTINE DFT1CM (LINE,NROW,NCOL,NLAY,INCTL,OUNIT,EXCMD,EOF,UBIN,
     1  U2DDSN,U3DDSN,U2DANM,U3DANM,ISKSP,ISKTS,LCSTRT,LCIBOU,LCSC1,
     2  LCTOP,LCIOFG,LCBOT,LCDELR,LCDELC,LCWELL,NWELLS,MXWELL,NRCHOP,
     3  LCIRCH,LCRECH,LCU2DS,LCU3DS,LCWDS,IUNIT,NUCLAS,CLASES,TITLES,
     4  IBOUND,UBOUND,DS1,DS2,STKA2D,STKA3D,NPER,NTS,KSP,KTS,IOFLG,
     5  LCUBOU,LCDELL,DELR,DELC,DELL,LCGRAF,LNGRAF,IBCFCB,IWELCB,IDRNCB,
     6  IRCHCB,IEVTCB,IRIVCB,IGHBCB,IHEDUN,IDDNUN,IPKGSP,PLANE,NOPRT,
     7  NCMDEX,NCMDRD)
C
C     DEFAULT COMMAND PROCESSING, TWO CHANGES MUST BE MADE TO
C     MODIFY DEFAULT PROCESSING COMMANDS:  (1) PARAMETER SPECIFICATION
C     OF NDFTCM, AND (2) DATA SPECIFICATIONS FOR DFTCMD.
C
      DOUBLE PRECISION DS1, DS2, STKA2D, STKA3D
      DIMENSION DS1(NCOL*NROW*NLAY),DS2(NCOL*NROW*NLAY),CLASES(20)
      DIMENSION STKA2D(NCOL,NROW),STKA3D(NCOL,NROW,NLAY),DELL(NLAY)
      DIMENSION PLANE(3)
      INTEGER IBOUND(NCOL,NROW,NLAY),UBOUND(NCOL,NROW,NLAY)
$INSERT STKSIZE.INS
$INSERT STKDEF.INS
      INTEGER IUNIT(24),OUNIT,LCWDS(2),IOFLG(NLAY,2),NTS(NPER)
      INTEGER IPKGSP(12)
      LOGICAL EXCMD,EOF,UBIN
      CHARACTER TITLES(4)*128,LINE*80
C
C
      PARAMETER (NDFTCM=14)
      CHARACTER DFTCMD(NDFTCM)*80
      DATA (DFTCMD(I),I=1,NDFTCM) /
     1 'TITL Analysis of active wells',
     2 'STAT WELL   00 01',
     3 'TITL Analysis of recharge volume for all nodes in the model',
     4 'STAT RECHV',
     5 'TITL Frequency analysis of starting heads at non-zero nodes',
     6 'HIST STRT   00 010000  20',
     7 'TITL Checking for pumpage at inactive or constant head nodes',
     8 'COMP WELL   00 GT    0.0 00 00-100',
     9 'TITL Checking for recharge at inactive or constant head nodes',
     1 'COMP RECH   00 GT    0.0 00 00-100',
     2 'TITL Checking for primary storage coefficient > 0.005',
     3 'COMP SC1    00 GT  0.005 00 000000  100',
     4 'TITL Checking for starting heads below layer bottom',
     5 'COMP STRT   00 LT    BOT 00' /
C
C1 -- PRINT A PAGE SHOWING THE DEFAULT COMMANDS ARE IN EFFECT
      WRITE (OUNIT,1) (TITLES(I),I=1,3)
    1 FORMAT (1H1,A128,2(/,1X,A128),//,1X,
     1 'PROCESSING DEFAULT COMMANDS:',/)
      WRITE (OUNIT,2) (DFTCMD(I),I=1,NDFTCM)
    2 FORMAT (/,10X,A80,/,10X,A80)
C
C2 -- PROCESS THE DEFAULT COMMAND INDIVIDUALLY
      DO 10 I=1,NDFTCM
         CALL DFLTEN (DFTCMD(I),NROW,NCOL,NLAY,INCTL,OUNIT,EXCMD,EOF,
     1     UBIN,U2DDSN,U3DDSN,U2DANM,U3DANM,ISKSP,ISKTS,LCSTRT,LCIBOU,
     2     LCSC1,LCTOP,LCIOFG,LCBOT,LCDELR,LCDELC,LCWELL,NWELLS,MXWELL,
     3     NRCHOP,LCIRCH,LCRECH,LCU2DS,LCU3DS,LCWDS,IUNIT,NUCLAS,CLASES,
     4     TITLES,IBOUND,UBOUND,DS1,DS2,STKA2D,STKA3D,NPER,NTS,KSP,KTS,
     5     IOFLG,LCUBOU,LCDELL,DELR,DELC,DELL,LCGRAF,LNGRAF,IBCFCB,
     6     IWELCB,IDRNCB,IRCHCB,IEVTCB,IRIVCB,IGHBCB,IHEDUN,IDDNUN,
     7     IPKGSP,PLANE,NOPRT)
         NCMDRD = NCMDRD + 1
         IF (EXCMD) NCMDEX = NCMDEX + 1
   10 CONTINUE
```

```
C
C3 -- RETURN TO THE MAIN PROGRAM AND EXIT BY SETTING THE END-OF-FILE FLAG
      EOF = .TRUE.
      RETURN
      END

      SUBROUTINE SCTL1H (CMD,DSN1,DSN2,DSN3,ANAME1,ANAME2,ANAME3,
     1  IULAY1,IULAY2,ISP1,ISP2,ITS1,ITS2,OPRATR,OUNIT)
C
C     PRINT HEADING FOR COMMANDS VIA CONSISTENT FORMATS
C
      INTEGER OUNIT
      CHARACTER CMD*4,DSN1*6,DSN2*6,DSN3*6,ANAME1*24,ANAME2*24
      CHARACTER ANAME3*24,OPRATR*2,CHAROP*17
C
      IF (CMD .EQ. 'STAT') THEN
C
C1 -- SUBTITLES FOR STATISTICS COMMAND
         WRITE (OUNIT,1) 'STATISTICS FOR : ',DSN1,ANAME1
         IF (IULAY1 .GT. 0) WRITE (OUNIT,2) IULAY1
         IF (IULAY1 .EQ. 0) WRITE (OUNIT,3)
         IF (IULAY1 .LT. 0) WRITE (OUNIT,4)
         IF (ISP1 .GT. 0) WRITE (OUNIT,5) ISP1,ITS1
         WRITE (OUNIT,6)
      ELSEIF (CMD .EQ. 'HIST') THEN
C
C2 -- SUBTITLES FOR HISTOGRAM COMMAND
         WRITE (OUNIT,1) ' HISTOGRAM FOR : ',DSN1,ANAME1
         IF (IULAY1 .GT. 0) WRITE (OUNIT,2) IULAY1
         IF (IULAY1 .EQ. 0) WRITE (OUNIT,3)
         IF (IULAY1 .LT. 0) WRITE (OUNIT,4)
         IF (ISP1 .GT. 0) WRITE (OUNIT,5) ISP1,ITS1
         WRITE (OUNIT,6)
      ELSEIF (CMD .EQ. 'COMP') THEN
C
C3 -- SUBTITLES FOR COMPARISON COMMAND
         WRITE (OUNIT,1) ' COMPARISON OF : ',DSN1,ANAME1
         IF (IULAY1 .GT. 0) WRITE (OUNIT,2) IULAY1
         IF (IULAY1 .EQ. 0) WRITE (OUNIT,3)
         IF (IULAY1 .LT. 0) WRITE (OUNIT,4)
         IF (ISP1 .GT. 0) WRITE (OUNIT,5) ISP1,ITS1
         WRITE (OUNIT,1) '       AGAINST : ',DSN2,ANAME2
         IF (IULAY2 .GT. 0) WRITE (OUNIT,2) IULAY2
         IF (IULAY2 .EQ. 0) WRITE (OUNIT,3)
         IF (IULAY2 .LT. 0) WRITE (OUNIT,4)
         IF (ISP2 .GT. 0) WRITE (OUNIT,5) ISP2,ITS2
         WRITE (OUNIT,7) DSN1,OPRATR,DSN2
      ELSEIF (CMD .EQ. 'MATH') THEN
```

```
C
C4 -- SUBTITLES FOR MATHEMATICAL COMPUTATION COMMAND
         WRITE (OUNIT,1) 'COMPUTATION OF : ',DSN3,ANAME3
         WRITE (OUNIT,1) '             FROM : ',DSN1,ANAME1
         IF (IULAY1 .GT. 0) WRITE (OUNIT,2) IULAY1
         IF (IULAY1 .EQ. 0) WRITE (OUNIT,3)
         IF (IULAY1 .LT. 0) WRITE (OUNIT,4)
         IF (ISP1 .GT. 0) WRITE (OUNIT,5) ISP1,ITS1
         IF (OPRATR .EQ. '+' .OR. OPRATR .EQ. 'AD') THEN
                CHAROP = '             PLUS : '
         ELSEIF (OPRATR .EQ. '-' .OR. OPRATR .EQ. 'SU') THEN
                CHAROP = '            MINUS : '
         ELSEIF (OPRATR .EQ. '/' .OR. OPRATR .EQ. 'DI') THEN
                CHAROP = '       DIVIDED BY : '
         ELSEIF (OPRATR .EQ. '*' .OR. OPRATR .EQ. 'MU') THEN
                CHAROP = ' MULTIPLIED BY : '
         ELSEIF (OPRATR .EQ. '**' .OR. OPRATR .EQ. 'EX') THEN
                CHAROP = '        RAISED TO : '
         ELSEIF (OPRATR .EQ. '||' .OR. OPRATR .EQ. 'AB') THEN
                CHAROP = ' ABSOLUTE VALUES '
         ELSE
                CHAROP = '          ?         : '
         ENDIF
         IF (OPRATR .NE. '||' .AND. OPRATR .NE. 'AB') THEN
             WRITE (OUNIT,1) CHAROP,DSN2,ANAME2
             IF (IULAY2 .GT. 0) WRITE (OUNIT,2) IULAY2
             IF (IULAY2 .EQ. 0) WRITE (OUNIT,3)
             IF (IULAY2 .LT. 0) WRITE (OUNIT,4)
             IF (ISP2 .GT. 0) WRITE (OUNIT,5) ISP2,ITS2
         ELSE
             WRITE (OUNIT,1) CHAROP
         ENDIF
         WRITE (OUNIT,6)
      ELSEIF (CMD .EQ. 'READ') THEN
C
C5 -- SUBTITLES FOR ARRAY READING COMMAND
         WRITE (OUNIT,1) '          READING : ',DSN1,ANAME1
         WRITE (OUNIT,8) IULAY1
         IF (ISP1 .NE. 0) WRITE (OUNIT,5) ISP1,ITS1
      ELSEIF (CMD .EQ. 'SLIC') THEN
C
C6 -- SUBTITLES FOR ARRAY SLICING COMMAND
         WRITE (OUNIT,1) '    CREATION OF: ','UBOUND',
     1                   'USER BOUNDARY MASK'
         WRITE (OUNIT,6)
      ELSEIF (CMD .EQ. 'WRIT') THEN
C
C7 -- SUBTITLES FOR ARRAY WRITING COMMAND
         WRITE (OUNIT,1) '    WRITING OF : ',DSN1,ANAME1
         IF (IULAY1 .GT. 0) WRITE (OUNIT,2) IULAY1
         IF (IULAY1 .EQ. 0) WRITE (OUNIT,3)
         IF (ISP1   .GT. 0) WRITE (OUNIT,5) ISP1,ITS1
         WRITE (OUNIT,8) IULAY2
         WRITE (OUNIT,9) DSN2
         WRITE (OUNIT,6)
      ELSEIF (CMD .EQ. 'PRIN') THEN
```

```
C
C8 -- SUBTITLES FOR ARRAY PRINTING COMMAND
         WRITE (OUNIT,1) '    PRINTING OF : ',DSN1,ANAME1
         IF (IULAY1 .GT. 0) WRITE (OUNIT,2) IULAY1
         IF (IULAY1 .EQ. 0) WRITE (OUNIT,3)
         IF (ISP1   .GT. 0) WRITE (OUNIT,5) ISP1,ITS1
         WRITE (OUNIT,9) DSN2
         WRITE (OUNIT,6)
      ENDIF
      RETURN
    1 FORMAT (//,10X,A17,A6,' - ',A24)
    2 FORMAT (36X,'LAYER ',I3)
    3 FORMAT (36X,'ALL LAYER(S)')
    4 FORMAT (36X,'COMPRESSED TO ONE LAYER')
    5 FORMAT (36X,'STRESS PERIOD ',I3,/,36X,'TIME STEP      ',I3)
    6 FORMAT (/)
    7 FORMAT (/,19X,'WHERE : ',A6,1X,A2,1X,A6,/)
    8 FORMAT (36X,'ON UNIT: ',I3)
    9 FORMAT (36X,'USING FORMAT CODE: ',A)
      END

      SUBROUTINE STA1SL (IPOINT,NLAY,NCR,NCRL,DSN,IOFLG)
C
C     SAVE LOCATIONS OF POINTS INTO IPOINT FOR LOCATING EXTREMA DURING STAT
C
      CHARACTER DSN*6
      INTEGER IOFLG(NLAY,2), IPOINT(NCRL)
C
$INSERT FLWCOM.INS
C
C1 -- INITIALIZE POINTERS TO LOCATION WITHIN THE MATRIX & WITHIN THE ARRAY
      LCGRID = 1
      LCARAY = 1
C
C2 -- CALCULATE THE MATRIX POSITION FOR EACH ARRAY POSITION
      IF (DSN .EQ. 'TOP') THEN
C
C2A --    TOP OF UNIT MAY NOT CONTAIN ALL LAYERS
         DO 10 K=1,NLAY
            LCGRID = LCGRID + NCR * (K-1)
            IF (LAYCON(K) .EQ. 2 .OR. LAYCON(K) .EQ. 3) THEN
               DO 5 I=1,NCR
                  IPOINT(LCARAY) = LCGRID
                  LCGRID         = LCGRID + 1
                  LCARAY         = LCARAY + 1
    5          CONTINUE
            ENDIF
   10    CONTINUE
      ELSEIF (DSN .EQ. 'BOT') THEN
C
C2B --    BOTTOM OF UNIT MAY NOT CONTAIN ALL LAYERS
         DO 20 K=1,NLAY
            LCGRID = LCGRID + NCR * (K-1)
            IF (LAYCON(K) .EQ. 1 .OR. LAYCON(K) .EQ. 3) THEN
               DO 15 I=1,NCR
                  IPOINT(LCARAY) = LCGRID
                  LCGRID         = LCGRID + 1
                  LCARAY         = LCARAY + 1
   15          CONTINUE
            ENDIF
   20    CONTINUE
      ELSEIF (DSN .EQ. 'HEAD') THEN
```

```
C
C2C --   COMPUTED HEADS MAY NOT CONTAIN ALL LAYERS
         DO 30 K=1,NLAY
            LCGRID = LCGRID + NCR * (K-1)
            IF (IOFLG(K,1) .NE. 0) THEN
               DO 25 I=1,NCR
                  IPOINT(LCARAY) = LCGRID
                  LCGRID         = LCGRID + 1
                  LCARAY         = LCARAY + 1
   25          CONTINUE
            ENDIF
   30    CONTINUE
      ELSEIF (DSN .EQ. 'DRAWDN') THEN
C
C2D --   COMPUTED DRAWDOWN MAY NOT CONTAIN ALL LAYERS
         DO 40 K=1,NLAY
            LCGRID = LCGRID + NCR * (K-1)
            IF (IOFLG(K,2) .NE. 0) THEN
               DO 35 I=1,NCR
                  IPOINT(LCARAY) = LCGRID
                  LCGRID         = LCGRID + 1
                  LCARAY         = LCARAY + 1
   35          CONTINUE
            ENDIF
   40    CONTINUE
      ELSE
C
C2E --   ALL OTHER DATA SETS CONTAIN EVERY LAYER
         DO 50 I=1,NCRL
            IPOINT(I) = I
   50    CONTINUE
      ENDIF
C
      RETURN
      END

      SUBROUTINE STA1EX (NCOL,NROW,NLAY,NODES,ARRAY,IPOINT,IJUMP,
     1                   OUNIT,EXCMD)
C
C     SORT INPUT ARRAY, THEN COMPUTE STATISTICS VIA SSTA1E
C
      DOUBLE PRECISION ARRAY
      DIMENSION ARRAY(NODES)
      DIMENSION STAT(15), IPOINT(NODES+1)
      INTEGER OUNIT
      LOGICAL EXCMD
C
C1 -- IF LESS THAN TWO DATA VALUES, THEN STATISTICS ARE USELESS
      IF (NODES .LT. 2) THEN
         WRITE (OUNIT,91) NODES
         EXCMD = .FALSE.
         RETURN
      ENDIF
C
C2 -- SORTS VALUES WITH PERMUTATIONS (USE ONE OF FOLLOWING)
C
C2A -- IMSL EDITION 10
C        CALL DSVRBP (NODES,ARRAY,ARRAY,IPOINT(IJUMP))
C
C2B -- IMSL EDITION 9
      CALL VSRTRD (ARRAY,NODES,IPOINT(IJUMP))
```

```
C
C3 -- COMPUTE AND PRINT THE STATISTICS
      CALL SSTA1E (ARRAY,NODES,STAT,OUNIT)
      WRITE (OUNIT,92)
      WRITE (OUNIT,93) STAT(1), STAT(8), STAT(2), STAT(3),
     1                 STAT(14), STAT(5)
      WRITE (OUNIT,94)
      WRITE (OUNIT,95) ARRAY(1), ARRAY(NODES), STAT(7), STAT(6),
     1                 STAT(9), NODES
      WRITE (OUNIT,96)
      WRITE (OUNIT,93) STAT(10), STAT(12), STAT(4), STAT(13), STAT(11)
      NCR = NCOL * NROW
      WRITE (OUNIT,97)
      IJUMP = IJUMP - 1
C
C4 -- FIND AND PRINT LOCATION OF MINIMUM:  ROW, COLUMN, LAYER
      CALL ULC1ND (IPOINT(IJUMP+1),NCOL,NCR,I,J,K)
      WRITE (OUNIT,98) 'MINIMUM',I,J,K
C
C5 -- FIND AND PRINT LOCATION OF MAXIMUM
      CALL ULC1ND (IPOINT(IJUMP+NODES),NCOL,NCR,I,J,K)
      WRITE (OUNIT,98) 'MAXIMUM',I,J,K
C
C6 -- FIND AND PRINT LOCATION OF MEDIAN(S)
      IREM = MOD (NODES,2)
      IMED = NODES / 2
      CALL ULC1ND (IPOINT(IJUMP+IMED),NCOL,NCR,I,J,K)
      WRITE (OUNIT,98) ' MEDIAN',I,J,K
      IF (IREM .EQ. 0) THEN
         IMED = IMED + 1
         CALL ULC1ND (IPOINT(IJUMP+IMED),NCOL,NCR,I,J,K)
         WRITE (OUNIT,98) ' MEDIAN',I,J,K
      ENDIF
      RETURN
   91 FORMAT (//,10X,'Insufficient data to compute statistics, number',
     1   ' of observations = ',I2)
   92 FORMAT (//,10X,'ARITHMETIC',6X,'ABSOLUTE VALUE',11X,'GEOMETRIC',
     1 12X,'HARMONIC',11X,'ROOT MEAN',/,10X,4(6X,'MEAN',10X),4X,
     2 'SQUARE',12X,'VARIANCE')
   93 FORMAT (4X,6G20.6E2)
   94 FORMAT (/,54X,'SUM OF',12X,'STANDARD',16X,'MEAN',11X,'NUMBER OF',
     1  /,13X,'MINIMUM',13X,'MAXIMUM',14X,'VALUES',11X,'DEVIATION',
     2     11X,'DEVIATION',14X,'VALUES')
   95 FORMAT (4X,5G20.6E2,I16)
   96 FORMAT (/,10X,'COEFFICIENT',15X,'LOWER',35X,'UPPER',6X,
     1  'NON-PARAMETRIC',/,10X,'OF SKEWNESS',12X,'QUARTILE',14X,
     2  'MEDIAN',12X,'QUARTILE',12X,'SKEWNESS')
   97 FORMAT (///,56X,'LOCATION',/,40X,'STATISTIC    ROW COLUMN  LAYER')
   98 FORMAT (/,42X,A7,3I7)
      END
      SUBROUTINE SSTA1E (ARRAY,NODES,STAT,OUNIT)
C
C COMPUTE STATISTICS ON AN ARRAY CONTAINING NODES POINTS
C  INPUT VECTOR ARRAY MUST BE SORTED BEFORE BEING PASSED TO SSTA1E
C  THE STATISTICS ARE RETURNED INTO THE FOLLOWING STAT ELEMENTS:
C
C    1 = ARITHMETIC MEAN
C    2 = GEOMETRIC MEAN (WHEN ALL ARRAY > 0)
C    3 = HARMONIC  MEAN (WHEN ALL ARRAY > 0)
C    4 = MEDIAN
C    5 = VARIANCE
C    6 = STANDARD DEVIATION
C    7 = SUM OF VALUES
C    8 = MEAN ABSOLUTE VALUE
C    9 = MEAN DEVIATION
C   10 = COEFFICIENT OF SKEWNESS :  MOMENT         (WHEN  N GE 3)
C   11 = COEFFICIENT OF SKEWNESS :  NON-PARAMETRIC (WHEN Q1 NE Q2)
C   12 = LOWER QUARTILE
C   13 = UPPER QUARTILE
C   14 = SQUARE ROOT OF MEAN OF THE SQUARES
C
      DOUBLE PRECISION PROX, HOLD, ARRAY, SSD, SUMX, SUMA, RECX
      DOUBLE PRECISION SUM2, AVG, COUNT, SUMDEV, SCD
      DIMENSION ARRAY (NODES)
      DIMENSION STAT(15)
      INTEGER OUNIT
      LOGICAL GTZERO, TRIP1, TRIP2
$INSERT TINY.INS
C
C1 -- INITIALIZE VARIABLES
      WRITE (OUNIT,93)
      SUMX = 0.
      SUMA = 0.
      PROX = 1.
      RECX = 0.
      SUM2 = 0.
      GTZERO = .TRUE.
      IF (ARRAY(1) .LE. 0.) THEN
         GTZERO = .FALSE.
         WRITE (OUNIT,94)
      ENDIF
      TRIP1 = .FALSE.
      TRIP2 = .FALSE.
C
C2 -- CALCULATE PRODUCT OF VALUES AND SUMS OF: VALUES, SQUARES, ABSOLUTE VALUES,
C       AND RECIPROCALS
      DO 10 I=1,NODES
         HOLD = ARRAY(I)
         SUMX = SUMX + HOLD
         SUM2 = SUM2 + HOLD * HOLD
         SUMA = SUMA + ABS (HOLD)
         IF (GTZERO .AND. .NOT. TRIP1) THEN
            PROX = PROX * HOLD
            RECX = RECX + 1.0 / HOLD
            IF (PROX .GE. HUGE .OR. PROX .LE. SMALL .OR.
     1          RECX .GE. HUGE .OR. RECX .LE. SMALL) TRIP1 = .TRUE.
         ENDIF
 10   CONTINUE
      IF (TRIP1) WRITE (OUNIT,95)
```

```
C
C3 -- CALCULATE: MEAN, SUM OF VALUES, MEAN ABSOLUTE VALUE, AND ROOT-MEAN-SQUARE
      COUNT    = DBLE (NODES)
      AVG      = SUMX / COUNT
      STAT (1) = AVG
      STAT (7) = SUMX
      STAT (8) = SUMA / COUNT
      STAT(14) = SQRT ( SUM2 / COUNT )
C
C4 -- CALCULATE: GEOMETRIC & HARMONIC MEANS
      IF (GTZERO .AND. .NOT. TRIP1) THEN
         HOLD     = 1.0 / COUNT
         HOLD     = PROX ** HOLD
         STAT (2) = HOLD
         STAT (3) = 1.0 / (RECX / COUNT)
      ELSE
         STAT (2) = 0.
         STAT (3) = 0.
      ENDIF
C
C5 -- CALCULATE:  MEDIAN, LOWER & UPPER QUARTILES
      M = NODES / 2
      IF (MOD(NODES,2) .EQ. 0) THEN
         STAT (4) = ( ARRAY (M) + ARRAY (M+1) ) / 2.0
      ELSE
         STAT (4) = ARRAY (M+1)
      ENDIF
      M = NODES / 4
      IF (MOD((NODES-1),4) .NE. 0) THEN
         IF (M .EQ. 0) M = 1
         STAT (12) = ( ARRAY (M) + ARRAY (M+1) ) / 2.0
         M = NODES - M
         STAT (13) = ( ARRAY (M) + ARRAY (M+1) ) / 2.0
      ELSE
         STAT (12) = ARRAY (M+1)
         STAT (13) = ARRAY (NODES-M)
      ENDIF
C
C6 -- CALCULATE:  NON-PARAMETRIC SKEWNESS
      HOLD = STAT (13) - STAT (12)
      IF (HOLD .LE. TINY) THEN
         WRITE (OUNIT,99)
         STAT (11) = 0.
      ELSE
         STAT (11) = ( STAT (12) + STAT (13) - 2. * STAT (4) ) / HOLD
      ENDIF
```

```
C
C7 -- CALCULATE: VARIANCE, STANDARD DEVIATION, MEAN DEVIATION,
C               AND COEFFICIENT OF SKEWNESS : MOMENT ESTIMATOR
      SSD = 0.0
      SCD = 0.0
      SUMDEV = 0.
      TRIP2 = .FALSE.
      DO 20 I=1,NODES
         HOLD   = ARRAY(I) - AVG
         SUMDEV = SUMDEV + DABS (HOLD)
         SSD    = SSD + HOLD * HOLD
         IF (.NOT. TRIP2 .AND. SCD .LT. HUGE .AND. SCD .GT. SMALL) THEN
            SCD = SCD + HOLD * HOLD * HOLD
         ELSE
            TRIP2 = .TRUE.
         ENDIF
20    CONTINUE
      STAT (5)  = SSD / (COUNT - 1)
      STAT (6)  = SQRT  ( STAT (5) )
      STAT (9)  = SUMDEV / COUNT
      STAT (10) = 0.0
      IF (NODES .LT. 3) THEN
         WRITE (OUNIT,96)
      ELSEIF (ABS (STAT(6)) .LE. TINY) THEN
         WRITE (OUNIT,97)
      ELSEIF (TRIP2) THEN
         WRITE (OUNIT,98)
      ELSE
         STAT (10) = SCD * COUNT / ( (COUNT - 1) * (COUNT - 2) )
         STAT (10) = STAT(10) / ( STAT(5) * STAT(6) )
      ENDIF
      RETURN
   93 FORMAT (///)
   94 FORMAT (10X,'Zero or negative values present in matrix, ',
     1  'therefore geometric and harmonic means were not computed',/)
   95 FORMAT (10X,'Overflow or underflow occurred, '
     1  'therefore geometric and harmonic means were not computed',/)
   96 FORMAT (10X,'Number of observations < 3, ',
     1  'therefore skewness was not computed',/)
   97 FORMAT (10X,'Standard deviation = 0, ',
     1  'therefore skewness was not computed',/)
   98 FORMAT (10X,'Overflow or underflow occurred, ',
     1  'therefore skewness was not computed',/)
   99 FORMAT (10X,'Upper and lower quartiles equal, ',
     1  'therefore non-parametric skewness was not computed',/)
      END
      SUBROUTINE HIS1EX (DARRAY,ARRAY,N,NUCLAS,CLASES,IUCLAS,OUNIT,
     1 EXCMD)
C
C     COMPUTE AND OUTPUT HISTOGRAM
C
C minimum number of classes = 3
C
      DOUBLE PRECISION DARRAY
      DIMENSION DARRAY (N), CLASES(20), FREQS(22)
      DIMENSION  ARRAY (N), UCLASS(20)
      INTEGER OUNIT
      LOGICAL EXCMD
$INSERT TINY.INS
C
C1 -- INTERPRET USER'S IUCLAS SPECIFICATION
C
C             IUCLAS <    0 : COMPUTED LOGARITHMIC CLASSES
C        0 <= IUCLAS <    3 : USER-SPECIFIED CLASSES UNLESS NUCLAS=0,
C                              THEN USES 20 COMPUTED ARITHMETIC CLASSES
C        3 <= IUCLAS <   20 : # OF COMPUTED ARITHMETIC CLASSES
C       20 <= IUCLAS        : 20 COMPUTED ARITHMETIC CLASSES
C
      IF (IUCLAS .GE. 3 .AND. IUCLAS .LT. 20) THEN
         NCLASS = IUCLAS
         WRITE (OUNIT,1) NCLASS,'arithmetic'
      ELSEIF (IUCLAS .GE. 20) THEN
         NCLASS = 20
         WRITE (OUNIT,1) NCLASS,'arithmetic'
      ELSEIF (IUCLAS .GE. 0 .AND. IUCLAS .LT. 3) THEN
         IF (NUCLAS .EQ. 0) THEN
            WRITE (OUNIT,2)
            NCLASS = 20
            IUCLAS = 20
         ELSE
            NCLASS = NUCLAS + 1
            NCUT   = NCLASS
            WRITE (OUNIT,3) NCLASS
            DO 40 I=1,NCLASS
               UCLASS (I) = CLASES (I)
   40       CONTINUE
         ENDIF
      ENDIF
    1 FORMAT (/,I10,1X,A11,' classes were computed')
    2 FORMAT (10X,'User-specified classes requested, but have not ',
     1   ' been read; using 20 arithmetic computed classes')
    3 FORMAT (10X,'Using',I5,' classes from CLASS data set')
C
C2 -- SORT THE ARRAY TO GET THE MIN AND MAX VALUES (USE ONE OF FOLLOWING)
C
C2A --   IMSL EDITION 10
C        CALL DSVRGN (N,DARRAY,DARRAY)
C
C2B --   IMSL EDITION 9
       CALL VSRTAD (DARRAY,N)
```

```
C
C3 -- CHECK THE MIN AND MAX VALUES
      DIFF = DABS (DARRAY(1) - DARRAY(N) )
      IF (DIFF .LE. TINY) THEN
         WRITE (OUNIT,6) N, DARRAY(1)
    6    FORMAT (/,10X,'All',I9,' values in the array are the ',
     1     'same:',F20.6)
         EXCMD = .FALSE.
         RETURN
      ENDIF
      IF (DARRAY(N) .GT. BIG) THEN
         WRITE (OUNIT,7) DARRAY(N)
    7    FORMAT (/,10X,'Value too large:',G20.5E2,', deactivated nodes',
     1                   ' may not have been masked')
         EXCMD = .FALSE.
         RETURN
      ENDIF
C
C4 -- COPY THE DOUBLE PRECISION ARRAY INTO THE SINGLE PRECISION ARRAY
      DO 50 I = 1, N
         ARRAY(I) = DARRAY(I)
   50 CONTINUE
C
C5 -- AUTO-SCALING OF CLASSES
      IF (IUCLAS .LT. 0) THEN
C
C5A --    AUTO-SCALING OF LOGARITHMIC CLASSES
         IF (ARRAY(1) .GE. -TINY) THEN
C
C5A1 --      MINIMUM >= 0, MAXIMUM > 0
            IF (ABS (ARRAY(1)) .LE. TINY) THEN
               XMIN = 0.0
            ELSE
               XMIN = LOG10 (ARRAY(1))
            ENDIF
            XMAX = LOG10 (ARRAY(N))
            SGN1 =  1.0
            SGN2 = -1.0
         ELSEIF (ARRAY(1) .LT. 0.0   .AND.
     1           ARRAY(N) .GE. TINY) THEN
C
C5A2 --      MINIMUM < 0, MAXIMUM >= 0
            XMIN = LOG10 (-ARRAY(1))
            IF (ABS (ARRAY(N)) .LE. TINY) THEN
               XMAX = 0.0
            ELSE
               XMAX = LOG10 (ARRAY(N))
            ENDIF
            SGN1 =  1.0
            SGN2 =  1.0
         ELSE
```

```
C
C5A3 --      MINIMUM < 0, MAXIMUM < 0
             XMIN = LOG10 (-ARRAY(1))
             XMAX = LOG10 (-ARRAY(N))
             SGN1 = -1.0
             SGN2 =  1.0
          ENDIF
          IF (XMIN .LT. 0.) XMIN = XMIN - 0.5
          IF (XMAX .GT. 0.) XMAX = XMAX + 0.5
          IXMIN = INT (XMIN)
          IXMAX = INT (XMAX)
          IRANGE = SGN1 * IXMAX + SGN2 * IXMIN
          IF (IRANGE .EQ. 0) THEN
              WRITE (OUNIT,8)
    8         FORMAT (11X,'Logarithmic scaling inappropriate.  ',
     1                'All data within the same power-of-ten')
              EXCMD = .FALSE.
              RETURN
          ENDIF
          INC    = 1
          NCLASS = IRANGE + 2
          IF (ARRAY(1) .LT. 0. .AND. ARRAY(N) .GT. 0.) NCLASS = NCLASS+2
90        CONTINUE
          IF (NCLASS .GT. 20) THEN
              INC = INC + 1
              NCLASS = IRANGE / INC
              GOTO 90
          ENDIF
          WRITE (OUNIT,1) NCLASS,'logarithmic'
          NCUT = NCLASS - 1
          II = 1
          DO 100 I=1,NCUT
             UCLASS(I) = (-SGN2) * 10.0 **
     1                  (((-SGN2) * (II-1) * INC) + IXMIN)
             II = II + 1
             IF (ABS (SGN2) .LE. TINY) THEN
                 SGN2 = -1.
                 IXMIN = 0
                 II    = 1
             ENDIF
             IF ((ARRAY(1) .LT. 0 .AND. ARRAY(N) .GT. 0) .AND.
     1           (ABS (UCLASS(I) + 1.) .LE. TINY)) SGN2 = 0.
  100     CONTINUE
      ELSEIF (IUCLAS .GE. 3) THEN
C
C5B --    AUTO-SCALING OF ARITHMETIC CLASSES
          RINC = (ARRAY(N) - ARRAY(1)) / FLOAT(NCLASS - 2)
          NCUT = NCLASS - 1
          DO 200 I=1,NCUT
             UCLASS (I) = ARRAY(1) + FLOAT(I-1) * RINC
  200     CONTINUE
          UCLASS (NCUT) = ARRAY(N)
      ENDIF
C
C6 -- FREQUENCIES ARRAY MUST BE EMPTY TO AVOID CUMULATIVE FREQUENCY
 1000 DO 1100 I=1,NCLASS+1
 1100     FREQS (I) = 0.
C
C7 -- COUNT NUMBER OF OCCURRENCES IN EACH CLASS (USE ONE OF THE FOLLOWING)
C
C7A -- IMSL EDITION 10
C        CALL OWFRQ (N,ARRAY,NCUT,2,ARRAY(1),ARRAY(N),0.,UCLASS,FREQS)
C
C7B -- IMSL EDITION 9
      CALL BDCOU1 (ARRAY,N,NCUT,UCLASS,0,0,FREQS,IER)
```

```
C
C8 -- PRINT THE FREQUENCY TABLE
      WRITE (OUNIT,900) UCLASS(1),FREQS(1)
      WRITE (OUNIT,901)
     X   (J,UCLASS(J-1),UCLASS(J),FREQS(J), J=2,NCLASS-1)
      WRITE (OUNIT,902) NCLASS,UCLASS(NCLASS-1),FREQS(NCLASS)
C
C9 -- PREPARE A VERTICAL LINE-PRINTER HISTOGRAM (USE ONE OF THE FOLLOWING)
C
C9A --  IMSL EDITION 10 (NEXT 2 STATEMENTS)
C        CALL UMACH (-2,OUNIT)
C        CALL VHSTP (NCLASS,FREQS,4,'HISTOGRAM')
C
C9B --  IMSL EDITION 9  (NEXT 2 STATEMENTS)
       CALL UGETIO (3,I,OUNIT)
       CALL USHST (FREQS,NCLASS,4,IER)
C
      RETURN
  900 FORMAT (//,10X,'CLASS',6X,'GREATER THAN',2X,'LESS THAN OR',
     1  ' EQUAL TO',9X,'POPULATION',//,12X,'1',22X,G20.6E3,F20.0)
  901 FORMAT (11X,I2,2X,2G20.6E3,F20.0)
  902 FORMAT (11X,I2,G22.6E3,20X,F20.0)
      END
      SUBROUTINE COM1EX (EXCMD,NCOL,NROW,NLAY,LENDS,OUNIT,
     1  DSN1,ARRAY1,IULAY1,DSN2,ARRAY2,IULAY2,OPRATR,IBOUND,UBOUND,
     2  IZMASK,IBMASK,IUMASK,UBIN,IOFLG,LIMIT)
C
C     COMPARE TWO ARRAYS AND PRINT NODES WHERE TEST PASSES
C      --IGNORE TEST WHERE NODE IS MASKED
C
      DOUBLE PRECISION ARRAY1, ARRAY2
      DIMENSION ARRAY1(NCOL*NROW*NLAY), ARRAY2(NCOL*NROW*NLAY)
      INTEGER OUNIT,UBOUND(NCOL*NROW*NLAY),IBOUND(NCOL*NROW*NLAY)
      INTEGER IOFLG(NLAY,2)
      CHARACTER DSN1*6,DSN2*6,OPRATR*2
      EXTERNAL UMASKD, IMASKD
      LOGICAL  UMASKD, IMASKD, EXCMD, FASTFF, UBIN
$INSERT TINY.INS
C
C1 -- DESCRIBE COMPARISON TEST BYPASS CONDITIONS
      MSKTYP = 1
      WRITE (OUNIT,1)
      IF (LIMIT .NE. 0) WRITE (OUNIT,6)
      IF (IZMASK .NE. 0) THEN
            WRITE (OUNIT,2)
            IZMASK = 0
      ENDIF
      IF (IBMASK .NE. 0) THEN
         MSKTYP = MSKTYP + 1
         IF      (IBMASK .EQ. -3) THEN
            WRITE (OUNIT,3) 'active or constant head'
         ELSEIF (IBMASK .EQ. -2) THEN
            WRITE (OUNIT,3) 'active or inactive'
         ELSEIF (IBMASK .EQ. -1) THEN
            WRITE (OUNIT,3) 'active'
         ELSEIF (IBMASK .EQ.  1) THEN
            WRITE (OUNIT,3) 'inactive or constant head'
         ELSEIF (IBMASK .EQ.  2) THEN
            WRITE (OUNIT,3) 'inactive'
         ELSEIF (IBMASK .EQ.  3) THEN
            WRITE (OUNIT,3) 'constant head'
         ENDIF
      ENDIF
      IF (IUMASK .NE. 0) THEN
         IF (.NOT. UBIN) THEN
            IUMASK = 0
            WRITE (OUNIT,4)
            GOTO 13
         ENDIF
         MSKTYP = MSKTYP + 2
         IF (IUMASK .LT. 0) WRITE (OUNIT,5) 'greater than zero'
         IF (IUMASK .GT. 0) WRITE (OUNIT,5) 'less than or equal to zero'
      ENDIF
   13 IF (MSKTYP .EQ. 1) WRITE (OUNIT,7)
```

```
C
C2 -- DETERMINE COMPARISON OPERATOR
      IOPR = 0
      IF (OPRATR .EQ. '='  .OR. OPRATR .EQ. 'EQ')  IOPR = 1
      IF (OPRATR .EQ. '' .OR. OPRATR .EQ. 'NE')  IOPR = 2
      IF (OPRATR .EQ. '<'  .OR. OPRATR .EQ. 'LT')  IOPR = 3
      IF (OPRATR .EQ. '<=' .OR. OPRATR .EQ. 'LE' .OR. OPRATR .EQ. '=<')
     1                                             IOPR = 4
      IF (OPRATR .EQ. '>'  .OR. OPRATR .EQ. 'GT')  IOPR = 5
      IF (OPRATR .EQ. '>=' .OR. OPRATR .EQ. 'GE' .OR. OPRATR .EQ. '=>')
     1                                             IOPR = 6
      IF (IOPR .EQ. 0) THEN
         EXCMD = .FALSE.
         WRITE (OUNIT,8) OPRATR
         RETURN
      ENDIF
C
C3 -- INITIALIZE LAYER RANGE AND BOUNDARY ARRAY POINTERS
      NCR = NCOL * NROW
      NCRL = NCR * NLAY
      IF (IULAY1 .GT. 0) THEN
         LAYSTR = IULAY1
         LAYEND = IULAY1
         LCB1   = 1 + NCR * (IULAY1 - 1)
         LAY1   = IULAY1
        LENDS = NCRL
      ELSEIF (IULAY1 .EQ. 0) THEN
         LAYSTR = 1
         LAYEND = NLAY
         LCB1   = 1
         LAY1   = 1
      ELSE
         LAYSTR = 1
         LAYEND = 1
         LCB1   = 1
         LAY1   = 1
      ENDIF
C
      IF (IULAY2 .GT. 0) THEN
         LAY2 = IULAY2
         LCB2 = 1 + NCR * (IULAY2 - 1)
      ELSE
         LAY2 = 1
         LCB2 = 1
      ENDIF
      LAYDIF = LAY2 - LAY1
      NMASK  = 0
C
C4 -- PROCEED WITH THE MASK AND COMPARE...
      WRITE (OUNIT,9) DSN1,DSN2
      NFOUND = 0
```

```
C
C4A -- PERFORM THE COMPARISON BY LAYER, CHECK FOR MISSING LAYER
      DO 3000 K=LAYSTR,LAYEND
         K2 = K + LAYDIF
         LCLAY1 = 1
         IPAD = LENDS
         CALL ULC1LY (NROW,NCOL,NLAY,DSN1,LCLAY1,OUNIT,K,IOFLG,
     1    IPAD,EXCMD)
         FASTFF = .FALSE.
         IF (.NOT. EXCMD) THEN
            EXCMD  = .TRUE.
            FASTFF = .TRUE.
         ENDIF
         LCLAY2 = 1
         IPAD = LENDS
         CALL ULC1LY (NROW,NCOL,NLAY,DSN2,LCLAY2,OUNIT,K2,IOFLG,
     1    IPAD,EXCMD)
         IF (FASTFF) THEN
            IF (.NOT. EXCMD) THEN
               EXCMD = .TRUE.
               GOTO 3000
            ENDIF
            LCLAY2 = LCLAY2 + NCR
            GOTO 3000
         ENDIF
         IF (.NOT. EXCMD) THEN
            EXCMD = .TRUE.
            LCLAY1 = LCLAY1 + NCR
            GOTO 3000
         ENDIF
C
C4B --    MOVE THROUGH LAYER BY ROW AND COLUMN
         DO 2000 I=1,NROW
         DO 1000 J=1,NCOL
C
C4B1 --   MASK HANDLING:
C                 NO MASK,    IMASK,   UMASK,  IMASK+UMASK
         GOTO (      50,        20,       30,       10) MSKTYP
   10    IF ( IMASKD ( IBOUND (LCB1),IBMASK ) ) GOTO 850
         IF ( UMASKD ( UBOUND (LCB1),UBMASK ) ) GOTO 850
         IF (LAYDIF .NE. 0) THEN
            IF ( IMASKD ( IBOUND (LCB2),IBMASK ) ) GOTO 850
            IF ( UMASKD ( UBOUND (LCB2),UBMASK ) ) GOTO 850
         ENDIF
         GOTO 50
   20    IF ( IMASKD ( IBOUND (LCB1),IBMASK ) ) GOTO 850
         IF (LAYDIF .NE. 0) THEN
            IF ( UMASKD ( IBOUND (LCB2),IBMASK ) ) GOTO 850
         ENDIF
         GOTO 50
   30    IF ( UMASKD ( UBOUND (LCB1),UBMASK ) ) GOTO 850
         IF (LAYDIF .NE. 0) THEN
            IF ( UMASKD ( UBOUND (LCB2),UBMASK ) ) GOTO 850
         ENDIF
   50    CONTINUE
         DELTA = ABS ( ARRAY1 (LCLAY1) - ARRAY2 (LCLAY2) )
```

```
C
C4B2 --   COMPARE:    EQ, NE, LT, LE, GT, GE
          GOTO (     100,110,120,120,140,140) IOPR
  100     IF (DELTA .LT. TINY) GOTO 800
          GOTO 900
  110     IF (DELTA .GT. TINY) GOTO 800
          GOTO 900
  120     IF (ARRAY1 (LCLAY1) .LT. ARRAY2 (LCLAY2) ) GOTO 800
          IF (IOPR .EQ. 4) GOTO 100
          GOTO 900
  140     IF (ARRAY1 (LCLAY1) .GT. ARRAY2 (LCLAY2) ) GOTO 800
          IF (IOPR .EQ. 6) GOTO 100
          GOTO 900
C
C4B3 --   COMPARISON WAS TRUE...PRINT LOCATION AND VALUES
  800     NFOUND = NFOUND + 1
          WRITE (OUNIT,801) I,J,K,ARRAY1(LCLAY1),K2,ARRAY2(LCLAY2)
          GOTO 900
  850     NMASK = NMASK + 1
C
C4B4 --   FINISHED WITH THIS NODE, ADVANCE POINTERS
  900     LCLAY1 = LCLAY1 + 1
          LCLAY2 = LCLAY2 + 1
          LCB1   = LCB1   + 1
          LCB2   = LCB2   + 1
C
C4B5 --   IF MAXIMUM NUMBER OF NODES HAS BEEN PRINTED THEN STOP COMPARING
          IF (LIMIT .GT. 0 .AND. NFOUND .GT. LIMIT) THEN
             WRITE (OUNIT,9000)
             GOTO 3001
          ENDIF
 1000 CONTINUE
 2000 CONTINUE
 3000 CONTINUE
C
 3001 CONTINUE
      WRITE (OUNIT,9001) NFOUND,NMASK,LENDS
      RETURN
    1 FORMAT (10X,'COMPARISON TEST BYPASS CONDITIONS:',/)
    2 FORMAT (/,10X,' Zero-mask not allowed during comparison command')
    3 FORMAT (10X,'when model boundary nodes are ',A)
    4 FORMAT (/,10X,'User-boundary mask requested, but user boundary '
     1          'has not been defined ....Mask ignored',/)
    5 FORMAT (10X,'when user boundary nodes are ',A)
    6 FORMAT (10X,'when number of nodes passing text exceeds ',I4)
    7 FORMAT ('+',45X,'WERE NOT ENABLED VIA MASK OPTIONS')
    8 FORMAT (/,10X,'Unknown operator:  ',A2)
    9 FORMAT (//,21X,2('   --------- ',A6,' --------- '),/,
     1  7X,'ROW',4X,'COLUMN',2(5X,'LAYER',15X,'VALUE'),/)
  801 FORMAT (2I10,2(I10,G20.6E3))
 9000 FORMAT (/,10X,'MAXIMUM NUMBER OF TRUE COMPARISONS REACHED',/)
 9001 FORMAT (/,10X,I9,' nodes identified',
     1         /,10X,I9,' nodes ignored via mask options',
     2         /,10X,I9,' nodes in data set')
      END
      SUBROUTINE MTH1EX (NCOL,NROW,NLAY,DSN1,DSN2,IULAY1,IULAY2,LCDS1,
     1  LCDS2,ARRAY1,ARRAY2,ARRAY3,IOFLG,OPRATR,OUNIT,LENDS,EXCMD)
C
C     PERFORM MATHEMATICAL OPERATION ON TWO ARRAYS AND CREATE A THIRD ARRAY
C
      DOUBLE PRECISION ARRAY1, ARRAY2, ARRAY3
      DIMENSION ARRAY1(NCOL*NROW*NLAY), ARRAY2(NCOL*NROW*NLAY)
      DIMENSION ARRAY3(LENDS)
      CHARACTER OPRATR*2,DSN1*6,DSN2*6
      INTEGER OUNIT,IOFLG(NLAY,2)
      LOGICAL EXCMD, FASTFF,SPARS1,SPARS2,SKPLAY
C
C1 -- DETERMINE IF EITHER DATA SET IS POTENTIALLY SPARSELY LAYERED
      SPARS1 = .FALSE.
      SPARS2 = .FALSE.
      IF ((DSN1 .EQ. 'HEAD' .OR. DSN1 .EQ. 'DRAWDN' .OR.
     1  DSN1 .EQ. 'TOP' .OR. DSN1 .EQ. 'BOT') .AND. IULAY1 .EQ. 0)
     2  SPARS1 = .TRUE.
      IF ((DSN2 .EQ. 'HEAD' .OR. DSN2 .EQ. 'DRAWDN' .OR.
     1  DSN2 .EQ. 'TOP' .OR. DSN2 .EQ. 'BOT') .AND. IULAY2 .EQ. 0)
     2  SPARS2 = .TRUE.
C
C2 -- DETERMINE OPERATOR TYPE
      IF (OPRATR .EQ. 'AD' .OR. OPRATR .EQ. '+') THEN
         IOPR = 1
      ELSEIF (OPRATR .EQ. 'SU' .OR. OPRATR .EQ. '-') THEN
         IOPR = 2
      ELSEIF (OPRATR .EQ. 'MU' .OR. OPRATR .EQ. '*') THEN
         IOPR = 3
      ELSEIF (OPRATR .EQ. 'DI' .OR. OPRATR .EQ. '/') THEN
         IOPR = 4
      ELSEIF (OPRATR .EQ. 'EX' .OR. OPRATR .EQ. '**') THEN
         IOPR = 5
      ELSEIF (OPRATR .EQ. 'AB' .OR. OPRATR .EQ. '||') THEN
         IOPR = 6
      ELSE
         EXCMD = .FALSE.
         WRITE (OUNIT,1) OPRATR
         RETURN
      ENDIF
C
C3 -- SET ARRAY POINTERS
      IPNT = 1
      IPT1 = LCDS1
      IPT2 = LCDS2
      IPT3 = 1
      LAY1 = IULAY1 - 1
      LAY2 = IULAY2 - 1
      IF (IULAY1 .EQ. 0) LAY1 = 0
      IF (IULAY2 .EQ. 0) LAY2 = 0
      LAY3 = 0
      NCR = NCOL * NROW
      NCRL = NCR * NLAY
C
C4 -- COMPUTE THE REQUESTED DATA SET
  100 CONTINUE
      IF (IPNT .GT. NCR .OR. IPNT .EQ. 1) THEN
C
C4A --    SET POINTERS AT THE BEGINNING OF EACH LAYER
         IPNT = 1
         SKPLAY = .FALSE.
         FASTFF = .FALSE.
         LAY1 = LAY1 + 1
         LAY2 = LAY2 + 1
         LAY3 = LAY3 + 1
```

```
C
C4A1 --  ALLOW FOR SPARSELY LAYERED DATA SET 1, SET FLAG
         IF (SPARS1) THEN
            IPT1 = 1
            IPAD = NCRL
            CALL ULC1LY (NROW,NCOL,NLAY,DSN1,IPT1,OUNIT,LAY1,IOFLG,
     1       IPAD,EXCMD)
            IF (.NOT. EXCMD) THEN
               EXCMD  = .TRUE.
               FASTFF = .TRUE.
            ENDIF
         ENDIF
C
C4A2 --  ALLOW FOR SPARSELY LAYERED DATA SET 2, SET FLAG
         IF (SPARS2) THEN
            IPT2 = 1
            IPAD = NCRL
            CALL ULC1LY (NROW,NCOL,NLAY,DSN2,IPT2,OUNIT,LAY2,IOFLG,
     1       IPAD,EXCMD)
         ENDIF
         IF (FASTFF) THEN
            SKPLAY = .TRUE.
            IF (EXCMD) THEN
               IPT2 = IPT2 + NCR
               WRITE (OUNIT,2) LAY3, DSN1
            ELSE
               EXCMD = .TRUE.
            ENDIF
         ENDIF
         IF (.NOT. EXCMD) THEN
            SKPLAY = .TRUE.
            EXCMD  = .TRUE.
            IPT1   = IPT1 + NCR
            WRITE (OUNIT,2) LAY3, DSN2
         ENDIF
      ENDIF
C
C4B -- WHEN FLAG SET, SET OUT DATA LAYER TO ZEROES
      IF (SKPLAY) THEN
         ARRAY3 (IPT3) = 0.0
      ELSE
C5 --     PERFORM OPERATION:  + , - , * , / , **, or ||
         GOTO (110,120,130,140,150,160) IOPR
  110    ARRAY3 (IPT3) = ARRAY1 (IPT1) + ARRAY2 (IPT2)
         GOTO 200
  120    ARRAY3 (IPT3) = ARRAY1 (IPT1) - ARRAY2 (IPT2)
         GOTO 200
  130    ARRAY3 (IPT3) = ARRAY1 (IPT1) * ARRAY2 (IPT2)
         GOTO 200
  140    ARRAY3 (IPT3) = ARRAY1 (IPT1) / ARRAY2 (IPT2)
         GOTO 200
  150    ARRAY3 (IPT3) = ARRAY1 (IPT1) ** ARRAY2 (IPT2)
         GOTO 200
  160    ARRAY3 (IPT3) = ABS ( ARRAY1 (IPT1) )
         GOTO 200
  200    IPT1 = IPT1 + 1
         IPT2 = IPT2 + 1
      ENDIF
      IPNT = IPNT + 1
      IPT3 = IPT3 + 1
      IF (IPT3 .LE. LENDS) GOTO 100
```

```
C
      RETURN
    1 FORMAT (//,10X,'Unknown operator:  ',A2)
    2 FORMAT (10X,'Populating layer',I3,' with zeroes because ',A6,
     1  ' has no corresponding layer')
      END

      SUBROUTINE REA1CL (LINE,NUCLAS,CLASES,OUNIT,EXCMD)
C
C     READ AND STORE USER-SPECIFIED CLASS DATA
C
      DIMENSION CLASES(20)
      INTEGER OUNIT
      CHARACTER LINE*80,ANAME*24
      LOGICAL EXCMD
      DATA ANAME /'HISTOGRAM CUT POINTS'/
C
C1 -- READ NUMBER OF CLASSES AND CHECK RANGE
      READ (LINE,1,ERR=90) NUCLAS
    1 FORMAT (15X,I2)
      IF (NUCLAS .LT. 3) THEN
         EXCMD = .FALSE.
         WRITE (OUNIT,2) NUCLAS
    2    FORMAT (/,' Attempted to READ',I4,' cut points;  minimum = 3')
         NUCLAS = 0
         RETURN
      ELSEIF (NUCLAS .GT. 20) THEN
         EXCMD = .FALSE.
         WRITE (OUNIT,3) NUCLAS
    3    FORMAT (/,' Attempted to READ',I4,' cut points;  maximum = 20')
         NUCLAS = 0
         RETURN
      ENDIF
C
C2 -- READ THE UNIT NUMBER CONTAINING THE CLASSES
      READ (LINE,4,ERR=91) INUNIT
    4 FORMAT (12X,I2)
C
C3 -- READ THE CLASSES
      CALL U1DREL (CLASES,ANAME,NUCLAS,INUNIT,OUNIT,0)
      RETURN
C
   90 EXCMD = .FALSE.
      WRITE (OUNIT,990) 'Number of cut points', LINE (16:18)
      RETURN
   91 EXCMD = .FALSE.
      WRITE (OUNIT,990) 'Unit number for reading CLASS', LINE(13:14)
      RETURN
  990 FORMAT (/,1X,A,' is not an integer:  ',A)
      END
      SUBROUTINE REA1DF (DSN,INUNIT,DSTYPE,ANAME,OUNIT,
     1  I1,I2,I3,I4,I5,I6,I7,I8,I9,I10,I11,TEXTEX)
C
C     DEFINE MODEL OUTPUT DATA SETS FOR THE READ COMMAND
C      RETURNS: ARRAY NAME, INPUT UNIT, TYPE, AND EXPECTED TEXT STRING
C
      DOUBLE PRECISION TEXTEX,EXTEXT(15)
      INTEGER OUNIT,IPKGUN(11)
      CHARACTER DSN*6,DSTYPE*3,ANAME*24,NAMUNI(15)*24,DSNUNI(15)*6
      LOGICAL EXCMD
      DATA (DSNUNI(I),NAMUNI(I),EXTEXT(I),I=1,15) /
     1       'STORAG', 'STORAGE CBC FLOW',           ' STORAGE',         1
     &       'CNHEAD', 'CONSTANT HEAD CBC FLOW',     'ANT HEAD',         2
     &       'RIFACE', 'RIGHT-FACE CBC FLOW',        'HT FACE ',         3
     &       'FRFACE', 'FRONT-FACE CBC FLOW',        'NT FACE ',         4
     &       'LOFACE', 'LOWER-FACE CBC FLOW',        'ER FACE ',         5
     2       'CBCWEL', 'WELL CBC FLOW',              '   WELLS',         6
     3       'CBCDRN', 'DRAIN CBC FLOW',             '  DRAINS',         7
     4       'CBCRIV', 'RIVER CBC FLOW',             ' LEAKAGE',         8
     5       'CBCEVT', 'EVAP-TRANS CBC FLOW',        '      ET',         9
     6       'CBCGHB', 'GEN-HEAD-BOUND CBC FLOW',    'P BOUNDS',        10
     7       'CBCRCH', 'RECHARGE CBC FLOW',          'RECHARGE',        11
     1       'HEAD',   'COMPUTED HEADS',             '    HEAD',        12
     2       'DRAWDN', 'COMPUTED DRAWDOWNS',         'DRAWDOWN',        13
     3       'WELL',   'WELLS',                      'NOT USED',        14
     4       'RECH',   'RECHARGE RATE',              'NOT USED'/        15
C
C1 -- INITIALIZE VARIABLES
      DSTYPE = 'XXX'
      IPOINT = 0
      CALL SREA1U (I1,I2,I3,I4,I5,I6,I7,I8,I9,I10,I11,IPKGUN)
C
C2 -- MATCH THE DATA SET NAME WITH UNIT NUMBER, ARRAY NAME, DATA TYPE,
C      AND EXPECTED TEXT STRING
      DO 10 I=1,15
         IF (DSN .EQ. DSNUNI(I)) THEN
            IPNT   = I - 4
            IF (I .LE. 5) IPNT = 1
            INUNIT = IPKGUN (IPNT)
            IF (ANAME .EQ. '') ANAME  = NAMUNI (I)
            DSTYPE = '3UR'
            TEXTEX = EXTEXT (I)
            GOTO 50
         ENDIF
   10 CONTINUE
C
C3 -- IF THE DATA SET IS HEAD, DRAWDOWN, PUMPAGE, OR RECHARGE; RESET DATA TYPE
   50 IF (DSN .EQ. 'HEAD' .OR. DSN .EQ. 'DRAWDN') DSTYPE = 'SUR'
      IF (DSN .EQ. 'WELL') DSTYPE = 'WRP'
      IF (DSN .EQ. 'RECH') DSTYPE = 'RRP'
C
C4 -- CHECK IF THE DATA SET IS A USER BOUNDARY ARRAY
      IF (DSN .EQ. 'UBOUND') THEN
         IF (ANAME .EQ. '') ANAME  = 'USER BOUNDARY'
         ISP    = 0
         ITS    = 0
         DSTYPE = '3FI'
         RETURN
      ENDIF
      RETURN
      END
      SUBROUTINE REA1EX (DSN,DSTYPE,NCOL,NROW,NLAY,INUNIT,EXCMD,OUNIT,
     1  LCWELL,MXWELL,NWELLS,LCIRCH,LCRECH,NRCHOP,LCU2DS,LCU3DS,
     2  IOFLG,LCUBOU,IWDS,ARRAY,IUSP,IUTS,TEXTEX,IPKGSP,UBIN,ANAME,
     3  NOPRT)
C
C     EXECUTE READ COMMAND BY CALLING APPROPRIATE DATA SET READING ROUTINE
C       DSTYPE (input) = TYPE OF ARRAY READER TO CALL
C       DSTYPE (output)= TYPE OF STACK TO BE BUBBLED WHEN EXCMD = .TRUE.
C
$INSERT ZARRAY.COMMON.INS
$INSERT STKSIZE.INS
$INSERT STKDEF.INS
      DOUBLE PRECISION TEXTEX
      DIMENSION ARRAY (NCOL*NROW*NLAY)
      INTEGER LCWDS(2),OUNIT,IOFLG(NLAY,2),IWDS(NCOL*NROW*NLAY)
      INTEGER IPKGSP(12)
      CHARACTER DSN*6,DSTYPE*3,ANAME*24
      LOGICAL EXCMD, UBIN
C
C1 -- INITIALIZE NUMBER OF CELLS / LAYER, AND NUMBER OF CELLS / SIMULATION
      NCR = NCOL * NROW
      NCRL = NCR * NLAY
C
C2 -- READ THE PROPER ARRAY, DETERMINED BY ITS TYPE
      IF (DSTYPE .EQ. '2FI') THEN
C
C2A --   2-DIMENSIONAL, FIXED-FORMAT, INTEGER ARRAY
         DSTYPE = '2D'
         CALL U2DINT(IWDS,ANAME,NROW,NCOL,0,INUNIT,OUNIT,0)
         CALL UXFERI (Z(LCU2DS(ISTKSZ)),IWDS,NCR)
      ELSEIF (DSTYPE .EQ. '2FR') THEN
C
C2B --   2-DIMENSIONAL, FIXED-FORMAT, REAL ARRAY
         DSTYPE = '2D'
         CALL U2DREL (ARRAY,ANAME,NROW,NCOL,0,INUNIT,OUNIT,0)
         CALL UXFERR (Z(LCU2DS(ISTKSZ)),ARRAY,NCR)
      ELSEIF (DSTYPE .EQ. '3FI' .AND. DSN .EQ. 'UBOUND') THEN
C
C2C --   3-DIMENSIONAL, FIXED-FORMAT, USER-BOUNDARY ARRAY
         DO 20 K=1,NLAY
            IPOS = LCUBOU + (K-1) * NCR
            CALL U2DINT (Z(IPOS),ANAME,NROW,NCOL,K,INUNIT,OUNIT,0)
   20    CONTINUE
         UBIN = .TRUE.
      ELSEIF (DSTYPE .EQ. '3FI' .AND. DSN .NE. 'UBOUND') THEN
C
C2D --   3-DIMENSIONAL, FIXED-FORMAT, INTEGER ARRAY
         DSTYPE = '3D'
         DO 30 K=1,NLAY
            IPOS = 1 + (K-1) * NCR
            CALL U2DINT(IWDS(IPOS),ANAME,NROW,NCOL,K,INUNIT,OUNIT,0)
   30    CONTINUE
         CALL UXFERI (Z(LCU3DS(ISTKSZ)),IWDS,NCRL)
      ELSEIF (DSTYPE .EQ. '3FR') THEN
C
C2E --   3-DIMENSIONAL, FIXED-FORMAT, REAL ARRAY
         DSTYPE = '3D'
         DO 50 K=1,NLAY
            CALL U2DREL (ARRAY,ANAME,NROW,NCOL,K,INUNIT,OUNIT,0)
            IPOS = LCU3DS(ISTKSZ) + (K-1) * NCR * 2
            CALL UXFERR (Z(IPOS),ARRAY,NCR)
   50    CONTINUE

      ELSEIF (DSTYPE .EQ. 'SUR') THEN
```

```
C
C2F --    3-DIMENSIONAL, SPARSELY-LAYERED, UNFORMATTED REAL ARRAY
          DSTYPE = '3D'
          LAYEND = 0
          IF (DSN .EQ. 'HEAD') THEN
             J = 1
          ELSEIF (DSN .EQ. 'DRAWDN') THEN
             J = 2
          ELSE
             WRITE (OUNIT,2)
             EXCMD = .FALSE.
             RETURN
          ENDIF
          DO 60 I=1,NLAY
             IF (IOFLG(I,J) .NE. 0) LAYEND = LAYEND + 1
   60     CONTINUE
          IF (LAYEND .EQ. 0) THEN
             EXCMD = .FALSE.
             WRITE (OUNIT,4) DSN,IUSP,IUTS
             RETURN
          ENDIF
          DO 70 K=1,LAYEND
             CALL ULYREL (DSN,IUSP,IUTS,INUNIT,ARRAY,NCOL,NROW,
     1        NLAY,TEXTEX,PERTIM,TOTIM,OUNIT,EXCMD)
             IPOS = LCU3DS(ISTKSZ) + (K-1) * NCR * 2
             CALL UXFERR (Z(IPOS),ARRAY,NCR)
   70     CONTINUE
       ELSEIF (DSTYPE .EQ. '3UR') THEN
C
C2G --    3-DIMENSIONAL, UNFORMATTED, REAL ARRAY
          DSTYPE = '3D'
          CALL U3DREL (DSN,IUSP,IUTS,INUNIT,ARRAY,NCOL,NROW,
     1     TEXTEX,NLAY,OUNIT,EXCMD)
          CALL UXFERR (Z(LCU3DS(ISTKSZ)),ARRAY,NCRL)
       ELSEIF (DSTYPE .EQ. 'WRP') THEN
C
C2H --    WELLS
          NREAD = IUSP - IPKGSP(2)
          IF (NREAD .LE. 0) THEN
              IPKGSP(2) = 0
              NREAD     = IUSP
              REWIND (INUNIT)
              READ   (INUNIT,3) MXWELL, IWELCB
          ENDIF
          DO 80 N=1,NREAD
             CALL WELDRP (NROW,NCOL,NLAY,Z(LCWELL),NWELLS,MXWELL,INUNIT,
     1                    IPKGSP(2),OUNIT,.FALSE.,NOPRT)
   80     CONTINUE
       ELSEIF (DSTYPE .EQ. 'RRP') THEN
C
C2I --    RECHARGE
          NREAD = IUSP - IPKGSP(8)
          IF (NREAD .LE. 0) THEN
              IPKGSP(8) = 0
              NREAD     = IUSP
              REWIND (INUNIT)
              READ   (INUNIT,3) NRCHOP, IRCHCB
          ENDIF
          DO 90 I=1,NREAD
             CALL RCHDRP (NRCHOP,Z(LCIRCH),Z(LCRECH),NROW,NCOL,NLAY,
     1         INUNIT,IPKGSP(8),OUNIT,.FALSE.,NOPRT)
   90     CONTINUE
       ELSE
```

```
C
C2J --    UNKNOWN DATA SET TYPE
          EXCMD = .FALSE.
          WRITE (OUNIT,1) DSN,DSTYPE
       ENDIF
       RETURN
     1 FORMAT (/,' Data set type for ',A6,' is unknown:   ',A3)
     2 FORMAT (/,' Only HEAD or DRAWDN may have data set type SUR')
     3 FORMAT (2I10)
     4 FORMAT (/,1X,A,' is not available for reading for stress period',
      1          I4,', time-step',I4,' because save flags are not set')
       END

       SUBROUTINE SREA1U (I1,I2,I3,I4,I5,I6,I7,I8,I9,I10,I11,IPKGUN)
C
C      MEMORY MANAGEMENT TO PUT UNIT NUMBERS INTO AN ARRAY
C      FOR LOOKUP WITHIN THE DO-LOOP OF REA1DF
       INTEGER IPKGUN(11)
       IPKGUN(1) = I1
       IPKGUN(2) = I2
       IPKGUN(3) = I3
       IPKGUN(4) = I4
       IPKGUN(5) = I5
       IPKGUN(6) = I6
       IPKGUN(7) = I7
       IPKGUN(8) = I8
       IPKGUN(9) = I9
       IPKGUN(10) = I10
       IPKGUN(11) = I11
       RETURN
       END

       SUBROUTINE SLI1EX (NCOL,NROW,NLAY,COORD,PLANE,UBOUND,
      1 DELR,DELC,DELL,OUNIT,EXCMD)
C
C EXECUTE THE 'SLIC'E COMMAND TO GENERATE A USER BOUNDARY BY EITHER TAKING
C  A ROW, COLUMN, OR LAYER DIRECTLY FROM THE MODEL GRID (SSLI1Y)
C  OR COMPUTING THE PLANAR EQUATION (SSLI1Q) & EXTRACTING THE USER BOUNDARY.
C
       INTEGER UBOUND (NCOL,NROW,NLAY), OUNIT, COORD(3,3)
       DIMENSION PLANE(4), EQ (3,3)
       DIMENSION DELR(NCOL),DELC(NROW),DELL(NLAY)
       LOGICAL EASY, EXCMD, BONDED
C
C1 -- PRINT THE SLICING COORDINATES TO USER'S OUTPUT UNIT
       WRITE (OUNIT,900) ((COORD(I,J),J=1,3), I=1,3)
C
C2 -- INITIALIZE THE USER BOUNDARY & TAKE A SLICE IF THE PROBLEM IS 'EASY'
       CALL SSLI1Y (NCOL,NROW,NLAY,COORD,DELR,DELC,DELL,
      1             PLANE,UBOUND,EASY,OUNIT,EXCMD)
       IF (EASY) THEN
          WRITE (OUNIT,901) (PLANE(I),I=1,4)
          RETURN
       ENDIF
C
C3 -- TRANSFORM THE GRID CELL COORDINATES TO ENGINEERING UNITS
       DO 50 N=1,3
          CALL UPOS1G (COORD(N,2),COORD(N,1),COORD(N,3),NROW,NCOL,NLAY,
      1     .TRUE.,DELC,DELR,DELL,EQ(N,2),EQ(N,1),EQ(N,3))
50     CONTINUE
C
C4 -- COMPUTE THE EQUATION OF THE SLICING PLANE
       CALL SSLI1Q (EQ,PLANE,OUNIT,EXCMD)
       IF (.NOT. EXCMD) RETURN
```

```
C
C5 -- EXTRACT THE USER BOUNDARY ARRAY
      BONDED = .FALSE.
      DO 120 K=1,NLAY
      DO 110 J=1,NROW
C
C5A --    FIX THE ROW AND LAYER LOCATIONS
         CALL UPOS1G (1,J,K,NROW,NCOL,NLAY,
     1                 .TRUE.,DELC,DELR,DELL,XNODE,YNODE,ZNODE)
C
C5B --    FIND THE CLOSEST COLUMN LOCATION
         XNODE = (PLANE(4) - PLANE(3) * ZNODE - PLANE(1) * YNODE)
     1           / PLANE(2)
         CALL UPOS1G (INODE,JNODE,KNODE,NROW,NCOL,NLAY,
     1                 .FALSE.,DELC,DELR,DELL,XNODE,YNODE,ZNODE)
C
C5C --    SET THE USER BOUNDARY IF COMPUTED COLUMN IS WITHIN MODEL GRID
         IF (INODE .GT. 0 .AND. INODE .LE. NCOL) THEN
             BONDED = .TRUE.
             UBOUND (INODE,JNODE,KNODE) = 1
         ENDIF
110   CONTINUE
120   CONTINUE
C
C5 -- IF NO COORDINATES WERE SLICED BY THE PLANE, THEN AN ERROR HAS OCCURRED
      IF (.NOT. BONDED) THEN
          EXCMD = .FALSE.
          WRITE (OUNIT,909)
      ENDIF
      WRITE (OUNIT,901) (PLANE(I),I=1,4)
      RETURN
900   FORMAT (10X,'COMPUTING USER BOUNDARY FROM THE GRID COORDINATES:',//,
     1         10X,'  ROW     COLUMN    LAYER',/,
     2         10X,'------+-----------+-------',/,
     3        3(9X,I5,5X,I5,5X,I5,/))
901   FORMAT (//,10X,'The equation of the slicing plane is:',//,10X,
     1 G10.3E2,' * Y  +  ',G10.3E2,' * X  +  ',G10.3E2,' * Z  =  ',
     2 G10.3E2)
909   FORMAT (//,10X,'No model coordinates were within the slicing ',
     1 'plane')
      END
      SUBROUTINE SSLI1Y (NCOL,NROW,NLAY,COORD,DELR,DELC,DELL,
     1                    PLANE,UBOUND,EASY,OUNIT,EXCMD)
C
C USER BOUNDARY (UBOUND) GENERATION ROUTINE:  INITIALIZE THE USER UBOUND;
C  DETERMINE IF ARRAY CAN BE SLICED ALONG AN EXISTING COLUMN, ROW OR LAYER;
C  IF SO, THEN SLICE THE ARRAY AND RETURN THE RESULTANT UBOUNDING ARRAY.
C  IF NOT, SET 'EASY' TO FALSE & RETURN.
C
      DIMENSION PLANE(4)
      DIMENSION DELR(NCOL), DELC(NROW), DELL(NLAY)
      INTEGER OUNIT, COORD(3,3), UBOUND(NCOL,NROW,NLAY)
      LOGICAL EXCMD, EASY
$INSERT TINY.INS
      EASY = .TRUE.
C
C1 -- INITIALIZE THE EQUATION OF THE PLANE & THE UBOUND ARRAY TO ZEROES
      DO 10 I=1,4
10       PLANE (I) = 0.0
100   DO 110 K=1,NLAY
      DO 109 I=1,NCOL
      DO 108 J=1,NROW
108      UBOUND(I,J,K) = 0
109    CONTINUE
110   CONTINUE
C
C2 -- CHECK IF UBOUND IS ALIGNED WITH A COLUMN
         IF (ABS(COORD(1,1)) .LE. TINY .AND. ABS(COORD(1,3)) .LE. TINY)
     1      THEN
            I = COORD(1,2)
            IF (I .LE. 0 .OR. I .GT. NCOL) THEN
                WRITE (OUNIT,900) 'COLUMN'
                GOTO 999
            ENDIF
            WRITE (OUNIT,901) 'column',I
            DO 106 K=1,NLAY
            DO 105 J=1,NROW
105           UBOUND (I,J,K) = 1
106         CONTINUE
            CALL UPOS1G (I,1,1,NROW,NCOL,NLAY,.TRUE.,DELC,DELR,DELL,
     1                   XNODE,YNODE,ZNODE)
            PLANE (4) = XNODE
            PLANE (2) = 1.0
            RETURN
         ENDIF
C
C3 -- CHECK IF UBOUND IS ALIGNED WITH A ROW
         IF ((ABS(COORD(1,2)) .LE. TINY .AND. ABS(COORD(1,3)) .LE. TINY)
     1   .OR.(COORD(1,1) .EQ. COORD(2,1) .AND. COORD(2,1) .EQ. COORD(3,1)
     2   .AND. ABS(COORD(1,1)) .GT. TINY )) THEN
            J = COORD(1,1)
            IF (J .LE. 0 .OR. J .GT. NROW) THEN
                WRITE (OUNIT,900) 'ROW'
                GOTO 999
            ENDIF
            WRITE (OUNIT,901) 'row',J
            DO 116 K=1,NLAY
            DO 115 I=1,NCOL
115            UBOUND (I,J,K) = 1
116         CONTINUE
            CALL UPOS1G (1,J,1,NROW,NCOL,NLAY,.TRUE.,DELC,DELR,DELL,
     1                   XNODE,YNODE,ZNODE)
            PLANE (4) = YNODE
            PLANE (1) = 1.0
            RETURN
         ENDIF
```

```
C
C4 -- CHECK IF UBOUND IS ALIGNED WITH A LAYER
          IF ((ABS(COORD(1,1)) .LE. TINY .AND. ABS(COORD(1,2)) .LE. TINY)
     1    .OR. (COORD(1,3) .EQ. COORD(2,3) .AND. COORD(2,3) .EQ. COORD(3,3)
     2      )) THEN
             K = COORD(1,3)
             IF (K .LE. 0 .OR. K .GT. NLAY) THEN
                WRITE (OUNIT,900) 'LAYER'
                GOTO 999
             ENDIF
             WRITE (OUNIT,901) 'layer',K
             DO 126 I=1,NCOL
             DO 125 J=1,NROW
125             UBOUND (I,J,K) = 1
126          CONTINUE
             CALL UPOS1G (1,1,K,NROW,NCOL,NLAY,.TRUE.,DELC,DELR,DELL,
     1                          XNODE,YNODE,ZNODE)
             PLANE (4) = ZNODE
             PLANE (3) = 1.0
             RETURN
          ENDIF
C
C5 -- NO COMMON COORDINATES...SET FLAG TO COMPUTE EQUATION USING SSLI1Q
       EASY = .FALSE.
       RETURN
999    EXCMD = .FALSE.
       RETURN
900    FORMAT (//,10X,'Specified ',A,' is outside the range of model ',
     1          'coordinates')
901    FORMAT (//,10X,'User boundary defined by ',A,I3)
       END

       SUBROUTINE SSLI1Q (A,U,OUNIT,EXCMD)
C
C DETERMINE THE EQUATION OF A SLICING PLANE VIA 3 GRID COORDINATES IN 'A'
C THE COEFFICIENTS OF THE PLANAR EQUATION ARE RETURNED IN THE VECTOR  'B'
C      U(1) * X  +  U(2) * Y  +  U(3) * Z  =  U(4)
C
       DIMENSION A(3,3), V(2,3), U(4)
       INTEGER OUNIT
       LOGICAL EXCMD
$INSERT USERS>JSSCOTT>MOD>TINY.INS
C
C1 -- CALCULATE TWO VECTORS LYING IN THE PLANE
       DO 20 I=2,3
          DO 10 J=1,3
             V (I-1,J) = A (I,J) - A (I-1,J)
10        CONTINUE
20     CONTINUE
C
C2 -- CALCULATE VECTOR PERPENDICULAR TO THE PLANE (CROSS PRODUCT)
       U (1) = V (1,2) * V (2,3) - V (2,2) * V (1,3)
       U (2) = V (1,3) * V (2,1) - V (1,1) * V (2,3)
       U (3) = V (1,1) * V (2,2) - V (1,2) * V (2,1)
C
C3 -- FIND THE NORM OF THE PERPENDICULAR VECTOR
       UNORM = SQRT ( U(1)*U(1) + U(2)*U(2) + U(3)*U(3) )
```

```
C     UMIN = 0.0
C      IF (U (2) .LT. UMIN .AND. ABS (U (2)) .GT. TINY) UMIN = U (2)
C      IF (U (3) .LT. UMIN .AND. ABS (U (3)) .GT. TINY) UMIN = U (3)
C
C4 -- IF ALL COEFFICIENTS ARE ZERO, THEN THE POINTS ARE COLINEAR
C      IF (ABS (UMIN) .LE. TINY) THEN
      IF (ABS (UNORM) .LE. TINY) THEN
         WRITE (OUNIT,900)
         EXCMD = .FALSE.
         RETURN
      ENDIF
C
C5 -- RESCALE COEFFICIENTS
      DO 30 I=1,3
C         U (I) = U (I) / UMIN
         U (I) = U (I) / UNORM
30    CONTINUE
C
C6 -- SOLVE FOR RIGHT-HAND SIDE OF THE PLANE EQUATION
      U (4) = 0.0
      DO 40 I=1,3
         U (4) = U (4) + (U (I) * A (1,I))
40    CONTINUE
      RETURN
900   FORMAT (//,' Unable to create user-boundary mask, coordinates',
     1          ' are colinear')
      END

      SUBROUTINE VEC1EX (NCOL,NROW,NLAY,LCLOFA,LCFRFA,LCRIFA,UBOUND,
     1 LCDELR,LCDELC,LCDELL,LCGRAF,LNGRAF,PLANE,ORIENT,GRFACT,IGUNIT,
     2 OUNIT,EXCMD)
C
C COMPUTE FLOW VECTORS, PROJECT ONTO VIEWING PLANE, AND WRITE ORIGIN &
C DESTINATION NODES TO AN OUTPUT DISK FILE IN A FORMAT COMPATIBLE
C WITH THE ARC/INFO 'GENERATE' COMMAND.
C
      DIMENSION PLANE (4)
      CHARACTER ORIENT*5
      INTEGER OUNIT, UBOUND(NCOL,NROW,NLAY)
      LOGICAL EXCMD, LOGON
      EXTERNAL UXFERD
$INSERT ZARRAY.COMMON.INS
$INSERT TINY.INS
C
C1 -- CALCULATE NUMBER OF NODES / LAYER, NODES / MATRIX
      NCR  = NCOL * NROW
      NCR2 = NCR  * 2
      NCRL = NCR  * NLAY
C
C2 -- IF GRAPHIC MULTIPLIER NOT SPECIFIED, SET IT TO UNITY;
C      IF GRAPHIC MULTIPLIER IS NEGATIVE, THEN SET FLAG FOR LOGARITHMIC SCALING
      IF (ABS(GRFACT) .LE. TINY) GRFACT = 1.0
      LOGON = .FALSE.
      IF (GRFACT .LT. 0.0) THEN
          GRFACT = ABS (GRFACT)
          LOGON  = .TRUE.
      ENDIF
```

```
C
C3 -- WRITE TO THE PRINT FILE A DESCRIPTION OF THE INTERPRETTED COMMAND
      WRITE (OUNIT,900) (PLANE(I),I=1,4),IGUNIT,GRFACT
900   FORMAT (//,10X,
     1       'FLOW VECTORS COMPUTED FROM CELL BY CELL FLOW TERMS',
     2 /,36X,'ON THE PLANE:  ',G10.3E2,' * Y + ',G10.3E2,' * X + ',
     3       G10.3E2,' * Z = ',G10.3E2,
     4 /,36X,'WRITTEN TO UNIT: ',I3,
     5 /,36X,'VECTOR SCALE FACTOR = ',G10.3E2)
      IF (LOGON) THEN
          WRITE (OUNIT,901) 'BASE-10 LOGARITHMIC'
      ELSE
          WRITE (OUNIT,901) 'ARITHMETIC'
      ENDIF
901   FORMAT (36X,'VECTORS SHOWN IN ',A,' UNITS')
902   FORMAT (36X,'HORIZONTAL AXIS REPRESENTS ',A,
     1       /,36X,' VERTICAL  AXIS REPRESENTS ',A)
C
C4 -- SET THE ORIENTATION WHEN SLICE IS ALIGNED WITH LAYER, COLUMN, OR ROW
      IF      (PLANE(1) .LE. TINY .AND. PLANE(2) .LE. TINY) THEN
          ORIENT = 'TOP'
      ELSEIF (PLANE(1) .LE. TINY .AND. PLANE(3) .LE. TINY) THEN
          ORIENT = 'SIDE'
      ELSEIF (PLANE(2) .LE. TINY .AND. PLANE(3) .LE. TINY) THEN
          ORIENT = 'FRONT'
      ENDIF
C
C5 -- DETERMINE THE VIEWING PLANE & SEE IF IT IS CONSISTENT WITH THE SLICE
      IF      (ORIENT .EQ. 'TOP') THEN
C         TOP VIEW (PLAN)
          IORENT = 1
          IF (ABS(PLANE(3)) .LE. TINY) GOTO 990
          WRITE (OUNIT,902) 'COLUMNS','ROWS'
      ELSEIF (ORIENT .EQ. 'SIDE') THEN
C         SIDE VIEW (RIGHT)
          IORENT = 2
          IF (ABS(PLANE(2)) .LE. TINY) GOTO 990
          WRITE (OUNIT,902) 'ROWS','LAYERS'
      ELSEIF (ORIENT .EQ. 'FRONT') THEN
C         SIDE VIEW (FRONT)
          IORENT = 3
          IF (ABS(PLANE(1)) .LE. TINY) GOTO 990
          WRITE (OUNIT,902) 'COLUMNS','LAYERS'
      ELSE
          WRITE (OUNIT,9997) ORIENT
 9997     FORMAT (/,' Unknown orientation specified:  ',A)
          EXCMD = .FALSE.
          RETURN
      ENDIF
```

```
C
C6 -- COMPUTE THE EXTENT OF THE Y & Z DIMENSIONS OF THE MODEL GRID
      SUMX = 0.0
      DO 2 ICOL = 0,NCOL-1
         SUMX = SUMX + Z (LCDELR + ICOL)
2     CONTINUE
      WRITE (OUNIT,903) 'RIGHT ',SUMX
903   FORMAT (36X,'THE ',A,' EDGE OF THE GRID IS AT ',G15.8E2)
      SUMY = 0.0
      DO 4 IROW = 0,NROW-1
         SUMY = SUMY + Z (LCDELC + IROW)
4     CONTINUE
      WRITE (OUNIT,903) 'FRONT ',SUMY
      SUMZ = 0.0
      DO 6 ILAY = 0,NLAY-1
         SUMZ = SUMZ + Z (LCDELL + ILAY)
6     CONTINUE
      WRITE (OUNIT,903) 'BOTTOM',SUMZ
      VLMIN = 99E18
      GZMIN = 99E18
      VLMAX = 0.
      NVECT = 0
C
C7 -- CALCULATE THE UNSCALED LENGTH OF THE ORTHOGONAL VECTORS
      DO 100 IPNT = 1,NCRL
C
C7A --   FIND CURRENT GRID LOCATION & SKIP VALUES NOT ON THE SLICING PLANE
         CALL ULC1ND (IPNT,NCOL,NCR,I,J,K)
         IF (ABS(UBOUND(J,I,K)) .LT. TINY) GOTO 100
         NVECT = NVECT + 1
         IF (NVECT*4 .GT. LNGRAF) GOTO 991
C
C7B --   CALCULATE THE AVERAGE OF THE FLOW FACES
         IZPNT  = (IPNT-1) * 2
         RLFACE = UXFERD (Z (LCRIFA + IZPNT))
         IF (J .NE. 1) THEN
              SPHOLD = UXFERD (Z (LCRIFA + IZPNT - 2))
              RLFACE = ( RLFACE + SPHOLD ) / 2.0
         ENDIF
         FBFACE = UXFERD (Z (LCFRFA + IZPNT))
         IF (I .NE. 1) THEN
              SPHOLD = UXFERD (Z (LCFRFA + IZPNT - NCOL * 2))
              FBFACE = ( FBFACE + SPHOLD ) / 2.0
         ENDIF
         TLFACE = UXFERD (Z (LCLOFA + IZPNT))
         IF (K .NE. 1) THEN
              SPHOLD = UXFERD (Z (LCLOFA + IZPNT - NCR2))
              TLFACE = ( TLFACE + SPHOLD ) / 2.0
         ENDIF
```

```
C
C7C --     CALCULATE THE LENGTH OF THE ORTHOGONAL VECTORS
           XL = RLFACE
           YL = FBFACE
           ZL = TLFACE
           KPNT = LCGRAF + (NVECT - 1) * 4
           Z(KPNT  ) = IPNT
           Z(KPNT+1) = XL
           Z(KPNT+2) = YL
           Z(KPNT+3) = ZL
           VL = SQRT (XL*XL + YL*YL + ZL*ZL)
           AL = ABS (VL)
           IF (AL .LT. VLMIN) VLMIN = AL
           IF (AL .GT. VLMAX) VLMAX = AL
           IF (AL .LT. GZMIN .AND. AL .GT. TINY) GZMIN = AL
100     CONTINUE
        WRITE (OUNIT,950) 'MINIMUM VECTOR', VLMIN, 'MAXIMUM VECTOR', VLMAX
C
C8 -- DETERMINE THE OFFSET NECESSARY FOR LOG TRANSFORMATION OF VALUES > 1.
        OFFLOG = ALOG10 (GZMIN)
        IF (OFFLOG .LT. 0.0 .AND. LOGON) THEN
            OFFLOG = 10. ** AINT (ABS (OFFLOG) + 1.5)
            WRITE (OUNIT,951) OFFLOG
  951       FORMAT (38X,'  BASE-10 MULTIPLIER   =',G11.3E2)
        ELSE
            OFFLOG = 1.0
        ENDIF
        VSMIN = 99E18
        VSMAX = 0.
C
C9 -- SCALE THE VECTORS AND PROJECT ONTO THE PLANE
        DO 200 JPNT = 1, NVECT
C
C9A --     FIND LOCATION, CALCULATE ARITHMETICALLY SCALED LENGTH,
C           COMPUTE LOGARTHMIC SCALING FACTOR
           KPNT = LCGRAF + (JPNT - 1) * 4
           IPNT = Z(KPNT)
           CALL ULC1ND (IPNT,NCOL,NCR,I,J,K)
           XL   = Z(KPNT+1)
           YL   = Z(KPNT+2)
           ZL   = Z(KPNT+3)
           VL   = SQRT (XL*XL + YL*YL + ZL*ZL)
           IF (LOGON .AND. VL .GT. TINY) THEN
              FACTLG = GRFACT * ALOG10 (VL * OFFLOG) / VL
           ELSE
              FACTLG = GRFACT
           ENDIF
           VL   = VL * FACTLG
           IF (VSMIN .GT. VL) VSMIN = VL
           IF (VSMAX .LT. VL) VSMAX = VL
C
C9B --     CALCULATE THE END OF THE VECTOR
           X1 = XL * FACTLG
           Y1 = YL * FACTLG
           Z1 = ZL * FACTLG
C
C9C --     PROJECT THE VECTOR ONTO THE VIEWING PLANE
           CALL UPOS1G (J,I,K,NROW,NCOL,NLAY,.TRUE.,
     1       Z(LCDELC),Z(LCDELR),Z(LCDELL),DELTAX,DELTAY,DELTAZ)
           GOTO (10,20,30) IORENT
```

```
C
C9C1 --   IORENT=1 (' TOP ') => X-DIMENSION IS COLUMNS, Y-DIMENSION IS ROWS
10            CONTINUE
              X0 = DELTAX
              X1 = X1 + X0
              Y0 = SUMY - DELTAY
              Y1 = Y0 - Y1
              GOTO 90
C
C9C2 --   IORENT=2 ('RIGHT') => X-DIMENSION IS ROWS,    Y-DIMENSION IS LAYERS
20            CONTINUE
              X0 = SUMY - DELTAY
              X1 = X0 - Y1
              Y0 = SUMZ - DELTAZ
              Y1 = Y0 - Z1
              GOTO 90
C
C9C3 --   IORENT=3 ('FRONT') => X-DIMENSION IS COLUMNS, Y-DIMENSION IS LAYERS
30            CONTINUE
              X0 = DELTAX
              X1 = X1 + X0
              Y0 = SUMZ - DELTAZ
              Y1 = Y0 - Z1
90          CONTINUE
            WRITE (IGUNIT,9000) IPNT,X0,Y0,X1,Y1
200     CONTINUE
C
C10 -- END OF GRID...ALL VECTORS HAVE BEEN WRITTEN.
        WRITE (IGUNIT,'(3HEND)')
        WRITE (OUNIT,952) NVECT
 952    FORMAT(35X,'NUMBER OF VECTORS WRITTEN =',I7)
        WRITE (OUNIT,950) 'MINIMUM SCALED', VSMIN, 'MAXIMUM SCALED', VSMAX
 950    FORMAT (2(38X,A,' LENGTH  =',G11.3E2,/))
        RETURN
C
 990    WRITE (OUNIT,9991) ORIENT
9991    FORMAT (/,' Orientation of ',A,' is incompatible with the slice')
        EXCMD = .FALSE.
        RETURN
 991    WRITE (OUNIT,9992)
9992    FORMAT (/,' Insufficient graphic storage available')
        EXCMD = .FALSE.
        RETURN
9000    FORMAT (I10,2(/,2F20.4),/,'END')
        END
```

```
      SUBROUTINE WRT1EX (NCOL,NROW,NLAY,ARRAY,INUNIT,DSTYPE,
     1 LENDS,IULAY,TEXT,KSTP,KPER,OUNIT,EXCMD)
C
C WRITE AN ARRAY TO A DISK FILE
C
      DOUBLE PRECISION ARRAY
      DIMENSION ARRAY(NCOL,NROW,NLAY)
      CHARACTER DSTYPE*3, FMT(20)*20, TEXT*24
      LOGICAL EXCMD
      INTEGER OUNIT
      PARAMETER (CNSTNT=0.0, IPRN=-1, IZERO=0)
      DATA FMT(1) /'(11G10.3)'/
      DATA FMT(2) /'(9G13.6)'/
      DATA FMT(3) /'(15F7.1)'/
      DATA FMT(4) /'(15F7.2)'/
      DATA FMT(5) /'(15F7.3)'/
      DATA FMT(6) /'(15F7.4)'/
      DATA FMT(7) /'(20F5.0)'/
      DATA FMT(8) /'(20F5.1)'/
      DATA FMT(9) /'(20F5.2)'/
      DATA FMT(10) /'(20F5.3)'/
      DATA FMT(11) /'(20F5.4)'/
      DATA FMT(12) /'(10G11.4)'/
      DATA FMT(13) /'(8G9.0)'/
      DATA FMT(14) /'(8G9.1)'/
      DATA FMT(15) /'(8G9.2)'/
      DATA FMT(16) /'(8G9.3)'/
      DATA FMT(17) /'(8F9.0)'/
      DATA FMT(18) /'(8F9.1)'/
      DATA FMT(19) /'(8F9.2)'/
      DATA FMT(20) /'(8F9.3)'/
C
C1 -- DETERMINE THE FORMAT OF DATA TO BE WRITTEN
      READ (DSTYPE,1,ERR=900) IP
1     FORMAT (I3)
      IF (IP .GT. 20) IP = 0
      IF (IP .EQ. 0)  IP = 12
C
C2 -- DETERMINE WHICH LAYER(S) ARE TO BE WRITTEN
      NCR = NCOL * NROW
      IF (IULAY .EQ. 0) THEN
         ISTART = 1
         IF (LENDS .NE. NCR) THEN
            IEND   = NLAY
         ELSE
            IEND   = 1
         ENDIF
      ELSE
         ISTART = IULAY
         IEND   = IULAY
      ENDIF
C
C3 -- WRITE THE DATA
      IF (IP .GT. 0) THEN
C
C3A --    WRITE FORMATTED DATA
         DO 100 K=ISTART,IEND
            WRITE (INUNIT,2) INUNIT,CNSTNT,FMT(IP),IPRN
    2       FORMAT (I10,F10.0,A20,I10)
            DO 50 I=1,NROW
               WRITE (INUNIT,FMT(IP)) (ARRAY(J,I,K),J=1,NCOL)
50          CONTINUE
  100    CONTINUE
       ELSE
```

```
C
C3B --    WRITE UNFORMATTED DATA
          IHOLD = -INUNIT
          DO 200 K=ISTART,IEND
             WRITE (OUNIT,3) IHOLD,CNSTNT,IPRN
    3        FORMAT(10X,'SAMPLE ARRAY-CONTROL RECORD:',I10,F10.0,20X,I10)
             WRITE (INUNIT) KSTP,KPER,IZERO,IZERO,TEXT,NCOL,NROW,K
             WRITE (INUNIT) ((ARRAY(J,I,K),J=1,NCOL),I=1,NROW)
200       CONTINUE
      ENDIF
      RETURN
C
900   WRITE (OUNIT,901) IP
      EXCMD = .FALSE.
901   FORMAT (10X,'Invalid format code:  ',A3)
      RETURN
      END

      SUBROUTINE PRT1EX (NCOL,NROW,NLAY,NCRL,DARRAY,ARRAY,DSTYPE,ANAME,
     1 IULAY,LCDS,LENDS,ISP,ITS,OUNIT,EXCMD)
C
C WRITE AN ARRAY IN THE PRINT FILE
C
      DOUBLE PRECISION DARRAY
      DIMENSION DARRAY(NCRL), ARRAY(NCRL)
      CHARACTER DSTYPE*3
      INTEGER OUNIT
      LOGICAL EXCMD
C
C1 -- DETERMINE THE FORMAT OF DATA TO BE WRITTEN, CALCULATE NODES / LAYER
      READ (DSTYPE,1,ERR=900) IP
1     FORMAT (I3)
      NCR = NCOL * NROW
C
C2 -- PRINT THE DATA BY LAYERS IN THE REQUESTED FORMAT
      IF (IULAY .EQ. 0) THEN
         ISTART = 1
         IF (LENDS .NE. NCR) THEN
            IEND   = NLAY
         ELSE
            IEND   = 1
         ENDIF
      ELSE
         ISTART = IULAY
         IEND   = IULAY
      ENDIF
      IPOS = LCDS
      DO 200 K=ISTART,IEND
        DO 150 I=1,NCR
           ARRAY(I) = DARRAY(IPOS)
           IPOS     = IPOS + 1
150     CONTINUE
        IF (IP .LT. 0) THEN
C
C2A --     WRITE OUT THE DATA IN STRIP FORMAT
           CALL ULAPRS (ARRAY,ANAME,ITS,ISP,NCOL,NROW,K,-IP,OUNIT)
        ELSE
C
C2B --     WRITE OUT THE DATA IN WRAP FORMAT
           CALL ULAPRW (ARRAY,ANAME,ITS,ISP,NCOL,NROW,K,IP,OUNIT)
        ENDIF
200   CONTINUE
      RETURN
```

```
C
900    WRITE (OUNIT,901) IP
       EXCMD = .FALSE.
901    FORMAT (10X,'Invalid format code:  ',A3)
       RETURN
       END

       SUBROUTINE HEA1EX (NCOL,NROW,NLAY,IOFLG,INUNIT,OUNIT,INOC,TEXTEX,
      1 ITSOC,NPER,NTS,IOUNIT,NODLST,ARRAY,EXCMD)
C
C EXTRACT BINARY COMPUTED HEADS FOR SELECTED NODES, AND WRITE TO A FILE
C
       DOUBLE PRECISION TEXTEX
       INTEGER IOFLG(NLAY,2), NTS(NPER), IROW(6), ICOL(6), ILAY(6)
       INTEGER OUNIT
       DIMENSION ARRAY(NCOL,NROW,NLAY)
       CHARACTER NODLST*72, HDFMT*25
       LOGICAL EXCMD
$INSERT MISVAL.INS
       DATA HDFMT /'(5I5,3G15.6E2)'/
C
C1 -- CALCULATE TOTAL TIME STEPS, DEFINE THE MISSING VALUE INDICATOR,
C        AND PRINT COMMAND DESCRIPTION
       NUMTS = 0
       DO 9 I=1,NPER
     9    NUMTS = NUMTS + NTS(I)
       READ (MISVAL(1),1) UNDEF
       WRITE (OUNIT,7) IOUNIT, HDFMT, NUMTS
C
C2 -- READ, CHECK, AND PRINT LOCATIONS OF NODES WHOSE HEADS ARE TO BE SAVED
       NUMNOD = 0
       ISTART = 1
   10 CONTINUE
          IEND   = ISTART + 11
          IF (NODLST(ISTART:IEND) .EQ. '            ') GOTO 20
          NUMNOD = NUMNOD + 1
          READ (NODLST(ISTART:IEND),2,ERR=900)
     1          IROW(NUMNOD),ICOL(NUMNOD),ILAY(NUMNOD)
          ISTART = IEND + 1
          IF (IROW(NUMNOD) .LE. 0 .OR. IROW(NUMNOD) .GT. NROW) THEN
             WRITE (OUNIT,3) ' Row', NUMNOD, NROW
             EXCMD = .FALSE.
          ENDIF
          IF (ICOL(NUMNOD) .LE. 0 .OR. ICOL(NUMNOD) .GT. NCOL) THEN
             WRITE (OUNIT,3) ' Column', NUMNOD, NCOL
             EXCMD = .FALSE.
          ENDIF
          IF (ILAY(NUMNOD) .LE. 0 .OR. ILAY(NUMNOD) .GT. NLAY) THEN
             WRITE (OUNIT,3) ' Layer', NUMNOD, NLAY
             EXCMD = .FALSE.
          ENDIF
       GOTO 10
20     CONTINUE
       IF (.NOT. EXCMD) RETURN
       WRITE (OUNIT,8) NUMNOD,(IROW(I),ICOL(I),ILAY(I),I=1,NUMNOD)
C
C3 -- READ SEQUENTIALLY THROUGH THE COMPUTED HEADS FILE, EXTRACTING DATA
       REWIND (INUNIT)
       DO 90 ISP = 1,NPER
          DO 80 ITS = 1, NTS(ISP)
             NUMNOT = 0
C
C3A --       GET THE HEAD SAVE FLAGS FOR THIS STRESS-PERIOD & TIME-STEP
             CALL UIO1FG (NPER,NLAY,NTS,ISP,ITS,IOFLG,ITSOC,INOC,
     1        OUNIT,EXCMD)
```

```
C
C3B --        READ AND STORE THE COMPUTED HEADS IN AN ARRAY
              DO 60 L = 1, NLAY
                 IF (IOFLG(L,1) .NE. 0) CALL ULYREL ('HEAD    ',ISP,ITS,
     1            INUNIT,ARRAY(1,1,L),NCOL,NROW,NLAY,TEXTEX,PERTIM,TOTIM,
     2            OUNIT,EXCMD)
   60         CONTINUE
C
C3C --        FOR EACH NODE REQUESTED, WRITE THE COMPUTED HEAD OR MISSING VALUE
              DO 70 N = 1, NUMNOD
                 IF (IOFLG(ILAY(N),1) .EQ. 0) THEN
                    NUMNOT = NUMNOT + 1
                    VALUE  = UNDEF
                    PEROUT = UNDEF
                    TOTOUT = UNDEF
                 ELSE
                    VALUE  = ARRAY (ICOL(N),IROW(N),ILAY(N))
                    PEROUT = PERTIM
                    TOTOUT = TOTIM
                 ENDIF
                 WRITE(IOUNIT,HDFMT) IROW(N),ICOL(N),ILAY(N),ISP,ITS,
     1                               PEROUT,TOTOUT,VALUE
   70         CONTINUE
              IF (NUMNOT .GT. 0) WRITE (OUNIT,5) NUMNOT,ISP,ITS
   80      CONTINUE
   90 CONTINUE
      RETURN
  900 EXCMD = .FALSE.
      WRITE (OUNIT,6) NUMNOD
      RETURN
    1 FORMAT (F10.0)
    2 FORMAT (3(I3,1X))
    3 FORMAT (A,' number ',I3,' is less than 0 or greater than',I4)
    5 FORMAT (I2,' nodes were set to a missing value because their ',
     1   'layer(s) was not saved for stress period',I4,', time step',I4)
    6 FORMAT (/,' Non-numeric row, column, or layer for node number',I4)
    7 FORMAT (//,14X,'WRITING OF : COMPUTED HEADS',
     1          /,17X,'ON UNIT : ',I2,
     2          /,12X,'USING FORMAT : ',A,
     3          /,23X,I3,' TIME STEP(S)')
    8 FORMAT (24X,I2,' NODES',//,
     1         14X,'ROW    COLUMN    LAYER',
     2         6(/,14X,I3,6X,I3,6X,I3),/)
      END
      SUBROUTINE REB1EX (NCOL,NROW,NLAY,IOFLG,IHEDUN,OUNIT,INOC,
     1 TEXTEX,ITSOC,NPER,NTS,IUSP,IUTS,IBOUND,HEAD,EXCMD)
C
C RESET THE MODEL BOUNDARY ARRAY FOR CELLS WHICH HAVE GONE DRY DURING SIMULATION
C
      DOUBLE PRECISION TEXTEX, TEST, DHEAD
      DIMENSION IOFLG (NLAY,2), HEAD (NCOL,NROW,NLAY)
      INTEGER IBOUND (NCOL,NROW,NLAY), NTS (NPER), OUNIT
      LOGICAL EXCMD
      PARAMETER (DHEAD=1.D30)
$INSERT TINY.INS
C
C1 -- WRITE DESCRIPTION OF THE INTERPRETTED COMMAND LINE
      WRITE (OUNIT,1) IUSP, IUTS
C
C2 -- RETRIEVE THE HEAD-SAVE FLAGS FOR THE REQUESTED STRESS PERIOD & TIME-STEP
      CALL UIO1FG (NPER,NLAY,NTS,IUSP,IUTS,IOFLG,ITSOC,INOC,
     1 OUNIT,EXCMD)
      IF (.NOT.EXCMD) THEN
         WRITE (OUNIT,2)
         RETURN
      ENDIF
C
C3 -- RETRIEVE COMPUTED HEADS BY LAYER, FOR THE REQUESTED TIME-STEP
      NFOUND = 0
      DO 10 L=1,NLAY
         IF (IOFLG(L,1) .NE. 0) THEN
            CALL ULYREL ('HEAD    ',IUSP,IUTS,IHEDUN,HEAD(1,1,L),
     1       NCOL,NROW,NLAY,TEXTEX,PERTIM,TOTIM,OUNIT,EXCMD)
            IF (EXCMD) NFOUND = NFOUND + 1
         ELSE
            WRITE (OUNIT,3) L, IUSP, IUTS
         ENDIF
   10 CONTINUE
      IF (NFOUND .EQ. 0) THEN
         EXCMD = .FALSE.
         WRITE (OUNIT,2)
         RETURN
      ENDIF
```

```
C
C4 -- COMPARE EACH VALUE OF COMPUTED HEAD TO THE VALUE SET BY THE
C      MODULAR MODEL SUBROUTINE SBCF1H WHEN CELLS GO DRY.
      NFOUND = 0
      WRITE (OUNIT,4)
      DO 40 K=1,NLAY
         IF (IOFLG(K,1) .EQ. 0) GOTO 40
         DO 30 I=1,NROW
            DO 20 J=1,NCOL
               THEAD = HEAD(J,I,K)
               IF (THEAD .LE. TINY) GOTO 20
               TEST = DHEAD / DBLE (THEAD)
               TEST = DABS (1.0D0 - TEST)
               IF (TEST .LE. TINY) THEN
                  IBOUND (J,I,K) = 0
                  WRITE (OUNIT,5) J,I,K
                  NFOUND = NFOUND + 1
               ENDIF
   20       CONTINUE
   30    CONTINUE
   40 CONTINUE
      WRITE (OUNIT,6) NFOUND
      RETURN
    1 FORMAT (//,   14X, 'RESETTING OF : MODEL-BOUNDARY ARRAY',
     1          /,9X,'FOR STRESS PERIOD :',I3,
     2          /,9X,'          TIME STEP :',I3,/)
    2 FORMAT (/,1X,'Unable to reset model-boundary array because ',
     1             'computed heads were not saved')
    3 FORMAT (1X,'Computed heads were not saved for layer:',I3,
     1             ', stress period:',I3,', time step:',I3)
    4 FORMAT (7X,'THE FOLLOWING NODES HAVE GONE DRY',//,
     1        14X,'ROW     COLUMN     LAYER')
    5 FORMAT (14X,I3,6X,I3,6X,I3)
    6 FORMAT (/,I4,' NODE(S) WENT DRY')
      END

      SUBROUTINE ULC1DS (NROW,NCOL,NLAY,DSN,IUNIT,OUNIT,IULAY,ARAY3D,
     1 LCDS,LENDS,ANAME,EXCMD,U2DDSN,U3DDSN,LCU2DS,LCU3DS,U2DANM,
     2 LCRECH,LCIRCH,LCDELC,LCDELR,NRCHOP,LCWELL,MXWELL,NWELLS,
     3 U3DANM,LCIBOU,LCSTRT,LCSC1,LCBOT,LCTOP,LCIOFG,FWDSCR,
     4 IPKGSP,ISKSP,ISKTS,ISP,ITS,NPER,NTS)
C
C     FIND THE LOCATION, LENGTH, AND NAME OF A DATA SET & RETURN INTO ARAY3D
C      ARAY3D OCCUPIES Z-ARRAY, BUT THIS IS NOT KNOWN DURING ULC1DS!
C
$INSERT ZARRAY.COMMON.INS
$INSERT STKSIZE.INS
$INSERT STKDEF.INS
C
      DOUBLE PRECISION ARAY3D
      DIMENSION ARAY3D(NCOL*NROW*NLAY)
      CHARACTER DSN*6, ANAME*24
      INTEGER IUNIT(24),OUNIT,IPKGSP(12),NTS(NPER)
      LOGICAL EXCMD, FWDSCR
C
C1 -- SEARCH FOR A DATA SET NAME BY LOOKING IN EACH OF SEVERAL LOCATIONS
      IF (DSN .EQ. 'WELL' .OR. DSN .EQ. 'AREA' .OR.
     1    DSN .EQ. 'RECH' .OR. DSN .EQ. 'RECHF') THEN
C
C1A --    PUMPAGE, CELL AREA, AND RECHARGE DATA SETS REQUIRE COMPUTATION
          CALL ULC1SP (NROW,NCOL,NLAY,IUNIT,DSN,ANAME,IULAY,
     1     ARAY3D,LENDS,OUNIT,EXCMD,Z(LCWELL),MXWELL,NWELLS,
     2     Z(LCRECH),Z(LCIRCH),Z(LCDELC),Z(LCDELR),Z(LCIBOU),NRCHOP,
     3     IPKGSP,ISP,ITS)
          IF (.NOT. EXCMD) RETURN
```

```
C
C1B -- IF NOT COMPUTED, THERE ARE THREE OTHER POSSIBILITIES
      ELSE
C
C1B1 --   IS THE DSN A MODULAR MODEL DATA ARRAY STORED IN A RESERVED LOCATION
          CALL USYDUD (NROW,NCOL,NLAY,IUNIT,DSN,LCDS,LENDS,ANAME,
     1      LCIBOU,LCSTRT,LCSC1,LCBOT,LCTOP,LCWELL,LCRECH)
          IF (LCDS .GT. 0) THEN
             ISP = 0
             ITS = 0
             GOTO 40
          ENDIF
C
C1B2 --   IF STILL NOT FOUND, CHECK IF DATA SET ON ONE OF THE STACKS
          IBASE2 = 1E8
          IBASE3 = 1E8
          DO 20 I=1,ISTKSZ
             IF (IBASE2 .GT. LCU2DS(I)) IBASE2 = LCU2DS(I)
             IF (IBASE3 .GT. LCU3DS(I)) IBASE3 = LCU3DS(I)
   20     CONTINUE
          IF (LCDS .LT. 0) CALL USKDUD (NROW,NCOL,NLAY,DSN,U2DDSN,
     2      U3DDSN,U2DANM,U3DANM,ISKSP,ISKTS,LCU2DS,LCU3DS,ISKPOS,
     3      Z(LCIOFG),IULAY,LCDS,LENDS,ANAME,ISP,ITS,FWDSCR,NPER,
     4      NTS,IPKGSP(12),IUNIT(12),.FALSE.,Z(IBASE2),IBASE2,
     5      Z(IBASE3),IBASE3,ARAY3D)
          IF (LCDS .GT. 0) GOTO 50
C
C1B3 --   IF STILL NOT FOUND, CHECK IF DATA SET NAME IS A REAL NUMBER?
          IF (LCDS .EQ. 0) THEN
             READ (DSN,2,ERR=90) VALUE
    2        FORMAT (F6.0)
             LENDS = NCOL * NROW * NLAY
             IF (IULAY .NE. 0) LENDS = NCOL * NROW
             DO 5 I=1,LENDS
                ARAY3D(I) = VALUE
    5        CONTINUE
             ANAME = 'CONSTANT VALUE = ' // DSN
             ISP   = 0
             ITS   = 0
             LCDS  = 1
             RETURN
          ENDIF
C
C1B4 --   STORE THE REQUESTED DATA ARRAY IN ARAY3D
   40     CALL UXFERR (ARAY3D,Z(LCDS),LENDS)
      ENDIF
C
C2 -- RESET LOCATION AND LENGTH WHEN USER NEEDS A SPECIFIC LAYER
   50 CONTINUE
      LCDS = 1
      IF (IULAY .GT. 0) CALL ULC1LY (NROW,NCOL,NLAY,DSN,LCDS,OUNIT,
     1  IULAY,Z(LCIOFG),LENDS,EXCMD)
      RETURN
C
C3 -- DATA SET NOT FOUND IN ANY OF THE POSSIBLE CACHES
   90 EXCMD = .FALSE.
      WRITE (OUNIT,1) DSN
      RETURN
    1 FORMAT (10X,'Data set ',A6,' was requested, but was not found')
      END
      SUBROUTINE ULC1SP (NROW,NCOL,NLAY,IUNIT,DSN,ANAME,IULAY,ARAY3D,
     1  LENDS,OUNIT,EXCMD,WELL,MXWELL,NWELLS,RECH,IRCH,DELC,DELR,
     2  IBOUND,NRCHOP,IPKGSP,ISP,ITS)
C
C     COMPUTE AND RETURN:  WELL, AREA, RECH, OR RECHF DATA SETS
C
      INTEGER IBOUND(NCOL,NROW,NLAY),IUNIT(24),OUNIT,IRCH(NCOL,NROW)
      INTEGER IPKGSP(12)
      DOUBLE PRECISION ARAY3D
      DIMENSION ARAY3D(NCOL,NROW,NLAY),RECH(NCOL,NROW),DELC(NROW)
      DIMENSION DELR(NCOL),WELL(4,MXWELL)
      CHARACTER DSN*6, ANAME*24
      LOGICAL EXCMD
C
C1 -- IF DATA SET NAME IS 'WELL', THEN PREPARE PUMPAGE DATA
      IF (DSN .EQ. 'WELL') THEN
         CALL USYDUD (NROW,NCOL,NLAY,IUNIT,DSN,LCDS,LENDS,ANAME,
     1     1,1,1,1,1,1,1)
         IF (LCDS .LE. 0) THEN
            EXCMD = .FALSE.
            WRITE (OUNIT,1) DSN,IUNIT(2)
            RETURN
         ENDIF
         CALL SWEL1T (NROW,NCOL,NLAY,MXWELL,NWELLS,WELL,IULAY,
     1     ARAY3D,LENDS,OUNIT)
         ISP = IPKGSP (2)
         ITS = 1
C
C2 -- IF DATA SET NAME IS 'AREA', COMPUTE AREA OF CELLS
      ELSEIF (DSN .EQ. 'AREA') THEN
         LENDS = NCOL * NROW
         ISP   = 0
         ITS   = 0
         ANAME = 'AREA OF CELLS'
         DO 120 K=1,NLAY
            DO 110 I=1,NCOL
               DO 100 J=1,NROW
                  ARAY3D (I,J,K) = DELC(J) * DELR(I)
100            CONTINUE
110         CONTINUE
120      CONTINUE
C
C3 -- IF DATA SET NAME IS 'RECH' OR 'RECHF', THEN PREPARE RECHARGE DATA
      ELSE
         CALL USYDUD (NROW,NCOL,NLAY,IUNIT,'RECH  ',LCDS,LENDS,ANAME,
     1     1,1,1,1,1,1,1)
         IF (LCDS .LE. 0) THEN
            EXCMD = .FALSE.
            WRITE (OUNIT,1) DSN,IUNIT(8)
            RETURN
         ENDIF
         CALL SRCH1T (NROW,NCOL,NLAY,DSN,LENDS,NRCHOP,IRCH,RECH,
     1     ANAME,OUNIT,IULAY,DELC,DELR,ARAY3D,IBOUND)
         ISP = IPKGSP (8)
         ITS = 1
      ENDIF
C
      RETURN
    1 FORMAT (//,10X,'Data set: ',A6,' not available.  ',
     1           ' IUNIT from basic package for data set = ',I3)
      END
      SUBROUTINE SWEL1T (NROW,NCOL,NLAY,MXWELL,NWELLS,WELL,IULAY,
     1   WELL3D,LENDS,OUNIT)
C
C     TRANSFORM WELL ARRAY TO COMPRESSED 2-D, OR 3-D FOR ANALYSIS
C
      DOUBLE PRECISION WELL3D
      DIMENSION WELL(4,MXWELL), WELL3D(NCOL,NROW,NLAY)
      INTEGER OUNIT
C
C1 -- INITIALIZE: NUMBER LAYERS TO COMPUTE, WELL ARRAY
      IF (IULAY .LT. 0) THEN
         LENDS = NCOL * NROW
           N   = 1
      ELSE
         LENDS = NCOL * NROW * NLAY
           N   = NLAY
      ENDIF
      DO 20 K=1,N
      DO 20 J=1,NCOL
      DO 20 I=1,NROW
         WELL3D (J,I,K) = 0.0
   20 CONTINUE
      IF (NWELLS .EQ. 0) RETURN
C
C2 -- CONSTRUCT ARRAY AS PER SPECIFICATION
      IF (IULAY .LT. 0) THEN
C
C2A --   COMPRESS SUM OF WELLS TO A SINGLE 2-D LAYER
C        IABS(IULAY) mask layer will be used if masking requested
         DO 30 M=1,NWELLS
            I = WELL (2,M)
            J = WELL (3,M)
            WELL3D (J,I,1) = WELL3D (J,I,1) + WELL (4,M)
   30    CONTINUE
      ELSE
C
C2B --   COMPLETE 3-D WELL ARRAY RETURNED
         DO 40 M=1,NWELLS
            I = WELL (2,M)
            J = WELL (3,M)
            K = WELL (1,M)
            WELL3D (J,I,K) = WELL (4,M)
   40    CONTINUE
      ENDIF
      RETURN
      END
      SUBROUTINE SRCH1T (NROW,NCOL,NLAY,DSN,LENDS,NRCHOP,IRCH,RECH,
     1   ANAME,OUNIT,IULAY,DELC,DELR,RECH3D,IBOUND)
C
C     TRANSFORM RECHARGE ARRAY TO COMPRESSED 2-D OR 3-D ARRAY
C     RETURNS RECHARGE RATE WHEN DSN = 'RECH'
C     RETURNS RECHARGE FLUX WHEN DSN = 'RECHF'
C
      DOUBLE PRECISION RECH3D
      DIMENSION RECH3D (NCOL,NROW,NLAY)
      DIMENSION RECH(NCOL,NROW),IRCH(NCOL,NROW),DELC(NROW),DELR(NCOL)
      DIMENSION IBOUND(NCOL,NROW,NLAY)
      INTEGER OUNIT
      CHARACTER DSN*6,ANAME*24
C
C1 -- INITIALIZE NUMBER OF LAYERS COMPUTED, RECHARGE ARRAY, & NAME
      IF (IULAY .GE. 0) THEN
         LENDS = NCOL * NROW * NLAY
           N   = NLAY
      ELSE
         LENDS = NCOL * NROW
           N   = 1
      ENDIF
      DO 10 K=1,N
      DO 10 I=1,NROW
      DO 10 J=1,NCOL
         RECH3D(J,I,K) = 0.0
   10 CONTINUE
      ANAME = 'RECHARGE RATE
C
C2 -- POPULATE RECHARGE ARRAY WITH RECHARGE RATES
C        IBOUND IS NOT USED TO IGNORE RECHARGE, UNLIKE:   RCH1FM
C        IBOUND IS ONLY USED TO APPLY RECH TO NEXT LOWER LAYER IN GRID
      IF (NRCHOP .EQ. 1 .OR. IULAY .LT. 0) THEN
C
C2A --   APPLY RECHARGE TO TOP LAYER
         DO 20 I=1,NROW
         DO 20 J=1,NCOL
            RECH3D (J,I,1) = RECH (J,I)
   20    CONTINUE
      ELSEIF (NRCHOP .EQ. 2 .AND. IULAY .GE. 0) THEN
C
C2B --   APPLY RECHARGE TO LAYER SPECIFIED IN IRCH
         DO 30 I=1,NROW
         DO 30 J=1,NCOL
            RECH3D (J,I,IRCH(J,I)) = RECH (J,I)
   30    CONTINUE
      ELSEIF (NRCHOP .EQ. 3 .AND. IULAY .GE. 0) THEN
C
C2C --   APPLY RECHARGE TO HIGHEST ACTIVE LAYER
         DO 40 I=1,NROW
         DO 40 J=1,NCOL
         DO 40 K=1,NLAY
            IF (IBOUND(J,I,K) .EQ. 0) GOTO 40
            RECH3D (J,I,K) = RECH (J,I)
   40    CONTINUE
      ENDIF
```

```
C
C3 -- CONVERT RECHARGE TO VOLUME/UNIT DEPTH, IF DATA SET NAME IS 'RECHF'
      IF (DSN .EQ. 'RECHV') THEN
         ANAME = 'RECHARGE FLUX
         DO 50 K=1,N
         DO 50 I=1,NROW
         DO 50 J=1,NCOL
            RECH3D (J,I,K) = RECH3D (J,I,K) * DELR(J) * DELC(I)
   50    CONTINUE
      ENDIF
      RETURN
      END
      SUBROUTINE USYDUD (NROW,NCOL,NLAY,IUNIT,DSN,LCDS,LENDS,
     1   ANAME,LCIBOU,LCSTRT,LCSC1,LCBOT,LCTOP,LCWELL,LCRECH)
C
C     DETERMINE IF MODEL INPUT DATA SET HAS BEEN READ & DESCRIBE IT
C      RETURN:  LOCATION AND LENGTH IN Z-ARRAY
C               WHEN LCDS = 0 :  DSN IS SYSTEM, BUT IUNIT = 0
C                    LCDS < 0 :  DSN IS NOT A SYSTEM NAME.
C
      CHARACTER DSN*6, ANAME*24
      DIMENSION IUNIT(24)
$INSERT FLWCOM.INS
C
C1 -- INITIALIZE LENGTH OF DATA ARRAY, CALCULATE NUMBER OF NODES
      LENDS = 0
      NCR = NROW * NCOL
      NCRL = NCR * NLAY
C
C2 -- DETERMINE DESCRIPTIVE NAME, LOCATION, AND LENGTH OF DATA SET
      IF (DSN .EQ. 'IBOUND') THEN
         ANAME = 'MODEL-BOUNDARY ARRAY    '
         LCDS  = LCIBOU
         LENDS = NCRL
      ELSEIF (DSN .EQ. 'STRT') THEN
         ANAME = 'INITIAL HEADS           '
         LCDS  = LCSTRT
         LENDS = NCRL
      ELSEIF (DSN .EQ. 'SC1') THEN
         ANAME = 'PRIMARY STORAGE COEFF.  '
         LCDS  = LCSC1
         IF (IUNIT(1) .LE. 0 .OR. LCSC1 .EQ. 1) LCDS = 0
         LENDS = NCRL
      ELSEIF (DSN .EQ. 'BOT') THEN
         ANAME = 'LAYER BOTTOM            '
         LCDS  = LCBOT
         IF (IUNIT(1) .LE. 0) LCDS = 0
         DO 10 I=1,NLAY
            IF (LAYCON(I).EQ.1.OR.LAYCON(I).EQ.3) LENDS=LENDS+NCR
   10    CONTINUE
      ELSEIF (DSN .EQ. 'TOP') THEN
         ANAME = 'LAYER TOP               '
         LCDS  = LCTOP
         IF (IUNIT(1) .LE. 0) LCDS = 0
         DO 20 I=1,NLAY
            IF (LAYCON(I).EQ.2.OR.LAYCON(I).EQ.3) LENDS=LENDS+NCR
   20    CONTINUE
      ELSEIF (DSN .EQ. 'WELL') THEN
         ANAME = 'WELL PUMPAGE            '
         LCDS  = LCWELL
         IF (IUNIT(2) .LE. 0) LCDS = 0
         LENDS = NCRL
      ELSEIF (DSN .EQ. 'RECH') THEN
         ANAME = 'RECHARGE                '
         LCDS  = LCRECH
         IF (IUNIT(8) .LE. 0) LCDS = 0
         LENDS = NCRL
      ELSE
         LCDS  = -999
      ENDIF
      RETURN
      END
      SUBROUTINE USKDUD (NROW,NCOL,NLAY,DSN,U2DDSN,U3DDSN,
     1  U2DANM,U3DANM,ISKSP,ISKTS,LCU2DS,LCU3DS,ISKPOS,IOFLG,IULAY,
     2  LCDS,LENDS,ANAME,ISP,ITS,FWDSCR,NPER,NTS,ITSOC,INOC,NOCOPY,
     3  STK2D,IBASE2,STK3D,IBASE3,ARRAY)
C
C     FIND LOCATION OF USER DATA SET ON STACK
C     RETURNS: LCDS   = LOCATION OF BEGINNING OF ARRAY ON Z-ARRAY
C              ISKPOS = 0:  DATA SET NOT FOUND ON ANY STACK
C                     > 0:  LOCATION OF USER ARRAY ON STACK
C              LENDS  = LENGTH OF ARRAY FOUND
C              ANAME  = NAME OF ARRAY
C     WHEN FWDSCR IS TRUE:  SEARCH IS UPWARD THROUGH STACKS
C     WHEN FWDSCR IS FALSE: SEARCH BEGINS IN STACK POSITION TWO
C
$INSERT STKSIZE.INS
$INSERT STKDEF.INS
      DOUBLE PRECISION STK2D, STK3D, ARRAY
      DIMENSION STK2D(NCOL*NROW*ISTKSZ), STK3D(NCOL*NROW*NLAY*ISTKSZ)
      DIMENSION ARRAY(NCOL*NROW*NLAY)
      CHARACTER DSN*6, ANAME*24
      INTEGER IOFLG(NLAY,2), NTS(NPER)
      LOGICAL FWDSCR, CKD, EXCMD, NOCOPY
C
C1 -- CALCULATE NUMBER OF NODES IN A LAYER, INITIALIZE LOCATION OF DATA, AND
C       NUMBER OF STACK ENTRIES
      NCR = NCOL * NROW
      NCRL = NCR * NLAY
      LCDS = 0
C
C2 -- SET THE STARTING STACK POSITION FOR THE SEARCH
      IF (FWDSCR) THEN
         ISTART = 1
      ELSE
         ISTART = 2
      ENDIF
C
C3 -- SCAN THE NAMES OF THE 2-D STACK FOR THE DATA SET NAME REQUESTED;
C       IF FOUND, RETURN THE LENGTH, STACK POSTION, LOCATION, DESCRIPTIVE
C       NAME, STRESS PERIOD, AND TIME STEP
      CKD  = .FALSE.
       I   = ISTART
      IEND = ISTKSZ
    5 IF (DSN .EQ. U2DDSN(I)) THEN
         LENDS = NCR
         ISKPOS = I
         LCDS   = LCU2DS(I)
         ANAME  = U2DANM(I)
         ISP = 0
         ITS = 0
         GOTO 50
      ENDIF
      I = I + 1
      IF (I .LE. IEND) GOTO 5
C
C4 -- IF STARTED WITH STACK POSITION TWO, GO BACK & CHECK POSITION ONE
      IF (.NOT. FWDSCR .AND. .NOT. CKD) THEN
          CKD = .TRUE.
            I  = 1
         IEND = 1
         GOTO 5
      ENDIF
```

```
C
C5 -- SCAN THE NAMES OF THE 3-D STACK FOR THE DATA SET NAME REQUESTED;
C      IF FOUND, RETURN THE LENGTH, STACK POSTION, LOCATION, DESCRIPTIVE
C      NAME, STRESS PERIOD, AND TIME STEP
       CKD = .FALSE.
        I  = ISTART
      IEND = ISTKSZ
   15 IF (DSN .EQ. U3DDSN(I)) THEN
         LENDS  = NCRL
         ISKPOS = I
         LCDS   = LCU3DS(I)
         ANAME  = U3DANM(I)
         ISP = ISKSP(I)
         ITS = ISKTS(I)
         GOTO 50
      ENDIF
      I = I + 1
      IF (I .LE. IEND) GOTO 15
C
C6 -- IF STARTED WITH STACK POSITION TWO, GO BACK & CHECK POSITION ONE
      IF (.NOT. FWDSCR .AND. .NOT. CKD) THEN
          CKD = .TRUE.
           I  = 1
         IEND = 1
         GOTO 15
      ENDIF
C
C7 -- WHEN NOT FOUND, SET STACK POSITION TO ZERO;
      ISKPOS = 0
      RETURN
   50 CONTINUE
      IF (NOCOPY) RETURN
C
C8 -- COPY THE ARRAY FROM THE APPROPRIATE STACK INTO THE WORK SPACE
      IF (LENDS .EQ. NCR) THEN
         ISTART   = (LCU2DS(ISKPOS) - IBASE2) / 2
         DO 100 I = 1, LENDS
            IMAT  = I + ISTART
            ARRAY (I) = STK2D (IMAT)
100      CONTINUE
      ELSE
         ISTART   = (LCU3DS(ISKPOS) - IBASE3) / 2
         DO 200 I = 1, LENDS
            IMAT  = I + ISTART
            ARRAY(I) = STK3D (IMAT)
200      CONTINUE
      ENDIF
C
C9 -- IF NO SPECIFIC LAYER REQUESTED, THEN RETURN THE COMPLETE DATA ARRAY
      IF (IULAY .NE. 0) RETURN
```

```
C
C10-- WHEN DATA SET IS HEAD OR DRAWDOWN, RE-CALCULATE THE LENGTH OF THE ARRAY
      IF (DSN .NE. 'HEAD' .AND. DSN .NE. 'DRAWDN') RETURN
      CALL UI01FG (NPER,NLAY,NTS,ISP,ITS,IOFLG,ITSOC,INOC,OUNIT,EXCMD)
      IF (.NOT.EXCMD) THEN
         ISKPOS = 0
         RETURN
      ENDIF
      IF (DSN .EQ. 'HEAD') THEN
         LENDS = 0
         DO 300 I=1,NLAY
            IF (IOFLG(I,1) .NE. 0) LENDS = LENDS + NCR
  300    CONTINUE
      ELSEIF (DSN .EQ. 'DRAWDN') THEN
         LENDS = 0
         DO 400 I=1,NLAY
            IF (IOFLG(I,2) .NE. 0) LENDS = LENDS + NCR
  400    CONTINUE
      ENDIF
      RETURN
      END


      SUBROUTINE ULC1LY (NROW,NCOL,NLAY,DSN,LCDS,OUNIT,IULAY,IOFLG,
     1                   LENDS,EXCMD)
C
C     FIND THE LOCATION OF THE SPECIFIED LAYER OFFSET FROM THE BEGINNING OF
C      THE DATA SET IN THE Z-ARRAY
C
C     LCDS (input)    = location of beginning of 3-D array
C     LCDS (output)   = location of beginning of IULAY layer of 3-D array
C                       left unchanged when IULAY is illegal
C     LENDS (input)   = length of data set DSN
C     LENDS (output)  = length of IULAY layer of data set DSN
C
      CHARACTER DSN*6
      INTEGER OUNIT, IOFLG(NLAY,2)
      LOGICAL EXCMD
$INSERT FLWCOM.INS
C
C1 -- IF LAYER ZERO IS REQUESTED, THEN THE COMPLETE ARRAY WILL BE USED
      IF (IULAY .LT. 1) RETURN
C
C2 -- CHECK THAT THE LAYER REQUESTED IS WITHIN THE LIMITS OF THE SIMULATION
      NCR  = NCOL * NROW
      NCRL = NCR  * NLAY
      IF (LENDS .LT. NCRL .AND. IULAY .GT. 1        .AND.
     1    DSN .NE. 'BOT'   .AND. DSN    .NE. 'TOP'    .AND.
     2    DSN .NE. 'HEAD'  .AND. DSN    .NE. 'DRAWDN') THEN
         WRITE (OUNIT,2) DSN, IULAY
         RETURN
      ENDIF
      IF (IULAY .GT. NLAY) THEN
         EXCMD = .FALSE.
         WRITE (OUNIT,1) DSN, IULAY, NLAY
         RETURN
      ENDIF
C
C3 -- COMPUTE THE LOCATION BY OFFSETTING FROM START OF ARRAY
      IF (DSN .EQ. 'BOT') THEN
```

```
C
C3A --    BOTTOM OF UNIT MAY NOT CONTAIN ALL LAYERS
          IF (LAYCON(IULAY) .NE. 1 .AND. LAYCON(IULAY) .NE. 3) THEN
             EXCMD = .FALSE.
             WRITE (OUNIT,3) DSN, IULAY, IULAY, LAYCON(IULAY)
             RETURN
          ENDIF
          DO 10 I=1,IULAY-1
          IF (LAYCON(I) .EQ. 1 .OR. LAYCON(I) .EQ. 3)LCDS=LCDS+NCR
   10     CONTINUE
       ELSEIF (DSN .EQ. 'TOP') THEN
C
C3B --    TOP OF UNIT MAY NOT CONTAIN ALL LAYERS
          IF (LAYCON(IULAY) .NE. 2 .AND. LAYCON(IULAY) .NE. 3) THEN
             EXCMD = .FALSE.
             WRITE (OUNIT,3) DSN, IULAY, LAYCON(IULAY)
             RETURN
          ENDIF
          DO 20 I=1,IULAY-1
          IF (LAYCON(I) .EQ. 2 .OR. LAYCON(I) .EQ. 3)LCDS=LCDS+NCR
   20     CONTINUE
       ELSEIF (DSN .EQ. 'HEAD') THEN
C
C3C --    COMPUTED HEADS MAY NOT CONTAIN ALL LAYERS
          IF (IOFLG (IULAY,1) .EQ. 0) THEN
             EXCMD = .FALSE.
             J = 3
             WRITE (OUNIT,4) DSN,IULAY,IULAY,J,IOFLG(IULAY,1)
             RETURN
          ENDIF
          DO 30 I=1,IULAY-1
             IF (IOFLG (I,1) .NE. 0) LCDS = LCDS + NCR
   30     CONTINUE
       ELSEIF (DSN .EQ. 'DRAWDN') THEN
C
C3D --    COMPUTED DRAWDOWN MAY NOT CONTAIN ALL LAYERS
          IF (IOFLG (IULAY,2) .EQ. 0) THEN
             EXCMD = .FALSE.
             J = 4
             WRITE (OUNIT,4) DSN,IULAY,IULAY,J,IOFLG(IULAY,2)
             RETURN
          ENDIF
          DO 40 I=1,IULAY-1
             IF (IOFLG (I,2) .NE. 0) LCDS = LCDS + NCR
   40     CONTINUE
       ELSE
C
C3E --    ALL OTHER DATA ARRAYS CONTAIN EVERY LAYER
          LCDS = LCDS + (IULAY - 1) * NCR
       ENDIF
       LENDS = NCR
       RETURN
     1 FORMAT (/,10X,'Data set ',A6,' layer',I3,' is too large.',
      1  ' Maximum number of layers =',I3)
     2 FORMAT (/,10X,'Unable to find data set ',A6,', layer',I3,
      1  ' :  two-dimensional;  Retaining complete data set')
     3 FORMAT (/,10X,'No ',A6,' layer',I3,' exists because LAYCON(',
      1  I2,') =',I3)
     4 FORMAT (/,10X,'No ',A6,' layer',I3,' exists because IOFLG(',
      1  I2,',',I2,') =',I3)
       END
      SUBROUTINE MAS1EX (LENDS,NCRL,ARRAY,IULAY,NLAY,IPOINT,
     1 EXCMD,UBIN,LCDS,IBMASK,IZMASK,IUMASK,OUNIT,DSN,IBOUND,UBOUND)
C
C     APPLY UP TO 3 MASKS TO DATA SET IN ARRAY
C       LENDS = NUMBER OF VALUES TO BE MASKED
C       IMKPOS = BEGINNING POINT IN MASK ARRAY
C       NPTS   = NUMBER OF VALUES REMAINING IN ARRAY AFTER MASKING
C
      DOUBLE PRECISION ARRAY
      DIMENSION ARRAY(NCRL)
      INTEGER IBOUND(NCRL),UBOUND(NCRL),IPOINT(NCRL)
      INTEGER OUNIT
      CHARACTER DSN*6
      LOGICAL  IMASKD, UMASKD, EXCMD, UBIN
      EXTERNAL IMASKD, UMASKD
$INSERT TINY.INS
C
C1 -- SET ARRAY POINTERS:  IMKPOS & IEND
      IF (IULAY .EQ. 0) THEN
         IMKPOS = 0
      ELSE
         IMKPOS = NCRL * (IABS(IULAY) - 1) / NLAY
         WRITE (OUNIT,3) IULAY
      ENDIF
      IEND = LCDS + LENDS - 1
C
C2 -- CHECK VALIDITY OF MASK REQUESTS
      IF (IUMASK .NE. 0 .AND. .NOT. UBIN) THEN
         IUMASK = 0
         WRITE (OUNIT,6)
      ENDIF
      IF (IBMASK .LT. -3 .OR. IBMASK .GT. 3) THEN
         WRITE (OUNIT,7) IBMASK
         IBMASK = 0
      ENDIF
      IF (IULAY .GT. NLAY) THEN
         WRITE (OUNIT,2) IULAY
         EXCMD = .FALSE.
         RETURN
      ENDIF
C
C3 -- APPLY THE MASKS
      NULOST = 0
      NBLOST = 0
      NZLOST = 0
      NPTS = 0
      DO 10 I=LCDS,IEND
         IMKPOS = IMKPOS + 1
C
C3A --   ZERO-MASK APPLICATION
         IF (IZMASK .NE. 0 .AND. ABS(ARRAY(I)) .LT. TINY) THEN
            NZLOST = NZLOST + 1
            GOTO 10
         ENDIF
C
C3B --   MODEL BOUNDARY MASK APPLICATION
         IF (IBMASK .NE. 0) THEN
            IF ( IMASKD (IBOUND(IMKPOS),IBMASK) ) THEN
            NBLOST = NBLOST + 1
            GOTO 10
            ENDIF
         ENDIF
```

```
C
C3C --    USER BOUNDARY MASK APPLICATION
          IF (IUMASK .NE. 0) THEN
             IF ( UMASKD (UBOUND(IMKPOS),IUMASK) ) THEN
             NULOST = NULOST + 1
             GOTO 10
             ENDIF
          ENDIF
          NPTS = NPTS + 1
          ARRAY(NPTS) = ARRAY(I)
          IPOINT(NPTS) = IPOINT(I)
   10 CONTINUE
C
C4 -- PRINT A SUMMARY OF THE RESULTS OF THE MASKING PROCESS
      WRITE (OUNIT,4) DSN,NPTS,LENDS
      IF (IZMASK .NE. 0) WRITE (OUNIT,5) NZLOST,
     1  'zero                        '
      IF     (IBMASK .EQ. -3) THEN
         WRITE (OUNIT,5) NBLOST,'active or constant head nodes'
      ELSEIF (IBMASK .EQ. -2) THEN
         WRITE (OUNIT,5) NBLOST,'inactive or active nodes'
      ELSEIF (IBMASK .EQ. -1) THEN
         WRITE (OUNIT,5) NBLOST,'active nodes'
      ELSEIF (IBMASK .EQ.  1) THEN
         WRITE (OUNIT,5) NBLOST,'inactive or constant head nodes'
      ELSEIF (IBMASK .EQ.  2) THEN
         WRITE (OUNIT,5) NBLOST,'inactive nodes'
      ELSEIF (IBMASK .EQ.  3) THEN
         WRITE (OUNIT,5) NBLOST,'constant head nodes'
      ENDIF
      IF (IUMASK .GT. 0) WRITE (OUNIT,5) NULOST,
     1  'outside the user boundary      '
      IF (IUMASK .LT. 0) WRITE (OUNIT,5) NULOST,
     1  'inside the user boundary       '
C
C5 -- RESET THE BEGINNING ARRAY POINTER AND THE ARRAY LENGTH COUNTER
      LENDS = NPTS
      LCDS  = 1
      IF (LENDS .EQ. 0) EXCMD = .FALSE.
      RETURN
C
    2 FORMAT (10X,'Illegal layer specified:  ',I3,' for masking')
    3 FORMAT (10X,'Beginning mask from layer',I3)
    4 FORMAT (/,10X,'Masking was performed on ',A6,//,
     1        I10,' points remain out of',I9,/)
    5 FORMAT (I10,' points excluded that were ',A)
    6 FORMAT (/,10X,'User-boundary mask requested, but user boundary ',
     1               'has not been defined....Mask ignored')
    7 FORMAT (/,10X,'Invalid value for model-boundary mask:  ',I2,
     1               ' ....Mask ignored')
      END
      SUBROUTINE MAS1MV (LENDS,NCRL,NCR,ARRAY,IULAY,NLAY,EXCMD,IOFLG,
     1 UBIN,LCDS,IBMASK,IZMASK,IUMASK,OUNIT,DSN,IBOUND,UBOUND,MISSTR)
C
C APPLY UP TO 3 MASKS TO DATA SET IN ARRAY & REPLACE W/ MISSING INDICATORS
C       LENDS  = LENGTH OF DATA SET
C       IMKPOS = BEGINNING POINT IN MASK ARRAY
C       NPTS   = NUMBER OF VALUES REMAINING IN ARRAY AFTER MASKING
C
      DOUBLE PRECISION ARRAY
      DIMENSION ARRAY (NCRL)
      INTEGER    IBOUND(NCRL), UBOUND(NCRL), IOFLG(NLAY,2)
      INTEGER    OUNIT
      CHARACTER DSN*6,  MISSTR*30
      LOGICAL    IMASKD, UMASKD, EXCMD, UBIN
      EXTERNAL   IMASKD, UMASKD
$INSERT FLWCOM.INS
$INSERT TINY.INS
$INSERT MISVAL.INS
C
C1 -- SET ARRAY POINTERS:  LAYER, IMKPOS, & ISTOP
      IF (IULAY .EQ. 0) THEN
         LAYER  = 0
         IMKPOS = 0
      ELSE
         LAYER  = IULAY - 1
         IMKPOS = NCRL * (IABS(IULAY) - 1) / NLAY
         WRITE (OUNIT,3) IULAY
      ENDIF
      ISTOP = LCDS + LENDS - 1
C
C2 -- CHECK FOR VALIDITY OF MASK REQUESTS
      IF (IUMASK .NE. 0 .AND. .NOT. UBIN) THEN
         IUMASK = 0
         WRITE (OUNIT,6)
      ENDIF
      IF (IBMASK .LT. -3 .OR. IBMASK .GT. 3) THEN
         WRITE (OUNIT,7) IBMASK
         IBMASK = 0
      ENDIF
      IF (IULAY .GT. NLAY) THEN
         WRITE (OUNIT,2) IULAY
         EXCMD = .FALSE.
         RETURN
      ENDIF
C
C3 -- DETERMINE THE MISSING VALUE INDICATORS
   90 CONTINUE
      DO 100 I = 1, 3
         ISTART = 1 + (I-1) * 10
         IEND   = I * 10
         IF (MISSTR(ISTART:IEND) .EQ. '          ') THEN
            READ  (MISVAL(I)          ,1)          MISING(I)
         ELSE
            READ  (MISSTR(ISTART:IEND),1,ERR=9000) MISING(I)
         ENDIF
  100 CONTINUE
```

```
C
C4 -- DETERMINE IF ARRAY IS 'SPARSELY' LAYERED
      IF     (DSN .EQ. 'TOP')    THEN
         IGO = 2
      ELSEIF (DSN .EQ. 'BOT')    THEN
         IGO = 3
      ELSEIF (DSN .EQ. 'HEAD')   THEN
         IGO = 4
      ELSEIF (DSN .EQ. 'DRAWDN') THEN
         IGO = 5
      ELSE
         IGO = 1
      ENDIF
C
C5 -- APPLY THE MASKS
      NULOST = 0
      NBLOST = 0
      NZLOST = 0
      NSLOST = 0
      NPTS   = 0
      DO 1000 I=LCDS,ISTOP
C
C5A --    CALCULATE MASK POSITION AND LAYER
         IMKPOS = IMKPOS + 1
         IF (MOD(I-1,NCR) .EQ. 0) LAYER = LAYER + 1
C
C5B --    CHECK FOR EXISTENCE OF LAYER WHEN 'SPARSELY' LAYERED
         GOTO (900,400,500,600,700) IGO
C
C5B1 --   DETERMINE IF LAYER DOES NOT EXIST FOR A 'TOP' LAYER NUMBER    (IGO=2)
  400    CONTINUE
         IF (LAYCON(LAYER) .NE. 2 .AND. LAYCON(LAYER) .NE. 3) GOTO 800
         GOTO 900
C
C5B2 --   DETERMINE IF LAYER DOES NOT EXIST FOR A 'BOT' LAYER NUMBER    (IGO=3)
  500    CONTINUE
         IF (LAYCON(LAYER) .NE. 1 .AND. LAYCON(LAYER) .NE. 3) GOTO 800
         GOTO 900
C
C5B3 --   DETERMINE IF LAYER DOES NOT EXIST FOR A 'HEAD' LAYER NUMBER   (IGO=4)
  600    CONTINUE
         IF (IOFLG (LAYER,1) .EQ. 0) GOTO 800
         GOTO 900
C
C5B4 --   DETERMINE IF LAYER DOES NOT EXIST FOR A 'DRAWDN' LAYER NUMBER (IGO=5)
  700    CONTINUE
         IF (IOFLG (LAYER,2) .EQ. 0) GOTO 800
         GOTO 900
C
C5B5 --   LAYER DOES NOT EXIST, SET VALUE TO MISSING INDICATOR #1
  800    CONTINUE
         ARRAY (I) = MISING (1)
         NZLOST    = NZLOST + 1
         NSLOST    = NSLOST + 1
         GOTO 1000
C
C5C --    LAYER EXISTS                                                  (IGO=1)
  900    CONTINUE
C
C5C1 --   ZERO-MASK APPLICATION
         IF (IZMASK .NE. 0 .AND. ABS(ARRAY(I)) .LT. TINY) THEN
            NZLOST = NZLOST + 1
            ARRAY(I) = MISING(1)
            GOTO 1000
         ENDIF
```

```
C
C5C2 --   MODEL BOUNDARY MASK APPLICATION
          IF (IBMASK .NE. 0) THEN
             IF ( IMASKD (IBOUND(IMKPOS),IBMASK) ) THEN
                  NBLOST = NBLOST + 1
                  ARRAY(I) = MISING(2)
                  GOTO 1000
             ENDIF
          ENDIF
C
C5C3 --   USER BOUNDARY MASK APPLICATION
          IF (IUMASK .NE. 0) THEN
             IF ( UMASKD (UBOUND(IMKPOS),IUMASK) ) THEN
                  NULOST = NULOST + 1
                  ARRAY(I) = MISING(3)
                  GOTO 1000
             ENDIF
          ENDIF
          NPTS = NPTS + 1
 1000 CONTINUE
C
C6 -- PRINT A SUMMARY OF THE RESULTS OF THE MASKING PROCESS
      WRITE (OUNIT,4) DSN,NPTS,LENDS
      IF (IZMASK .NE. 0) WRITE (OUNIT,5) NZLOST,
     1  'zero                                    '
      IF     (IBMASK .EQ. -3) THEN
         WRITE (OUNIT,5) NBLOST,'active or constant head nodes'
      ELSEIF (IBMASK .EQ. -2) THEN
         WRITE (OUNIT,5) NBLOST,'inactive or active nodes'
      ELSEIF (IBMASK .EQ. -1) THEN
         WRITE (OUNIT,5) NBLOST,'active nodes'
      ELSEIF (IBMASK .EQ.  1) THEN
         WRITE (OUNIT,5) NBLOST,'inactive or constant head nodes'
      ELSEIF (IBMASK .EQ.  2) THEN
         WRITE (OUNIT,5) NBLOST,'inactive nodes'
      ELSEIF (IBMASK .EQ.  3) THEN
         WRITE (OUNIT,5) NBLOST,'constant head nodes'
      ENDIF
      IF (IUMASK .GT. 0) WRITE (OUNIT,5) NULOST,
     1  'outside the user boundary       '
      IF (IUMASK .LT. 0) WRITE (OUNIT,5) NULOST,
     1  'inside the user boundary        '
      IF (IGO    .GT. 1) WRITE (OUNIT,5) NSLOST,
     1  'on a non-existent layer (included in zero-mask total)'
      RETURN
C
9000  WRITE (OUNIT,8) I, MISSTR(ISTART:IEND), MISVAL(I)
      MISSTR(ISTART:IEND) = '               '
      GOTO 90
    1 FORMAT (F10.0)
    2 FORMAT (10X,'Illegal layer specified:  ',I3,' for masking')
    3 FORMAT (10X,'Beginning mask from layer',I3)
    4 FORMAT (/,10X,'Masking was performed on ',A6,//,
     1         I10,' values unmasked out of',I9,/)
    5 FORMAT (I10,' points masked that were ',A)
    6 FORMAT (/,10X,'User-boundary mask requested, but user boundary ',
     1              'has not been defined....Mask ignored')
    7 FORMAT (/,10X,'Invalid value for model-boundary mask:  ',I2,
     1              ' ....Mask ignored')
    8 FORMAT (/,10X,'Illegal missing-value indicator #',I1,':  ',A,
     1        /,10X,' Set to default indicator:            ',A)
      END
      LOGICAL FUNCTION UMASKD (MASK,MASKEY)
C
C     DETERMINE IF MASK VALUE REPRESENTS A MASK:  (TRUE)
C       BASED UPON THE ACTION SPECIFIED BY THE MASKEY
C
C TRUTH-TABLE | MASKEY < 0    MASKEY = 0     MASKEY > 0
C ------------+------------+--------------+-------------
C   MASK <= 0 |  FALSE          FALSE           TRUE
C             +
C   MASK >  0 |   TRUE          FALSE          FALSE
C
      UMASKD = .FALSE.
      IF (MASKEY) 10,40,20
   10 IF (MASK) 40,40,30
   20 IF (MASK) 30,30,40
   30 UMASKD = .TRUE.
   40 RETURN
      END

      LOGICAL FUNCTION IMASKD (MASVAL,MASKEY)
C
C DETERMINE WHETHER A MASK HAS OCCURRED USING A MASK KEY AND THE MODEL BOUNDARY
C
C1 -- ASSUME THE VALUE IS UNMASKED, THEN FIND CORRECT MASK KEY
      IMASKD = .FALSE.
C
C1A -- UNMASKED IF INACTIVE MODEL NODE
      IF     (MASKEY .EQ. -3) THEN
          IF (MASVAL .EQ. 0) RETURN
C
C1B -- UNMASKED IF CONSTANT HEAD NODE
      ELSEIF (MASKEY .EQ. -2) THEN
          IF (MASVAL .LT. 0) RETURN
C
C1C -- UNMASKED IF INACTIVE OR CONSTANT HEAD NODE
      ELSEIF (MASKEY .EQ. -1) THEN
          IF (MASVAL .LE. 0) RETURN
C
C1D -- ALL NODES UNMASKED
      ELSEIF (MASKEY .EQ. 0) THEN
                               RETURN
C
C1E -- UNMASKED IF ACTIVE NODE
      ELSEIF (MASKEY .EQ.  1) THEN
          IF (MASVAL .GT. 0) RETURN
C
C1F -- UNMASKED IF ACTIVE OR CONSTANT HEAD NODE
      ELSEIF (MASKEY .EQ.  2) THEN
          IF (MASVAL .NE. 0) RETURN
C
C1G -- UNMASKED IF ACTIVE OR INACTIVE NODE
      ELSEIF (MASKEY .EQ. 3) THEN
          IF (MASVAL .GE. 0) RETURN
      ENDIF
C
C2 -- VALUE HAS BEEN MASKED
      IMASKD = .TRUE.
      RETURN
      END
      SUBROUTINE UBUBLE (LCSTK,DSNSTK,NAMSTK,ISPSTK,ITSSTK,CMD,
     1                  FLAG3D,OUNIT,DSNNEW,NAMNEW,ISPNEW,ITSNEW)
C
C     BUBBLE THE STACK DSN'S, LOCATIONS, AND DESCRIPTORS
C
$INSERT STKSIZE.INS
      INTEGER LCSTK(ISTKSZ),ISPSTK(ISTKSZ),ITSSTK(ISTKSZ),OUNIT
      CHARACTER DSNSTK(ISTKSZ)*6,NAMSTK(ISTKSZ)*24,DSNNEW*6,NAMNEW*24
      LOGICAL FLAG3D
C
C1 -- WRITE AN INFORMATIONAL MESSAGE, AND SAVE LOCATION OF LAST STACK POSITION
      IF ( FLAG3D ) THEN
         WRITE (OUNIT,1) 'THREE', CMD
      ELSE
         WRITE (OUNIT,1) '  TWO',CMD
      ENDIF
      LCTEMP = LCSTK(ISTKSZ)
C
C2 -- MOVING DOWNWARD THROUGH THE STACK, SHIFT THE LOCATION, NAME, AND
C       DESCRIPTION OF EACH ARRAY UPWARD ONE STACK POSITION.
C       IF 3-D STACK, ALSO MOVE STRESS-PERIOD AND TIME-STEP UPWARD.
      DO 10 I=ISTKSZ-1,1,-1
      J = I + 1
      LCSTK(J)  = LCSTK(I)
      DSNSTK(J) = DSNSTK(I)
      NAMSTK(J) = NAMSTK(I)
      IF (FLAG3D) THEN
         ISPSTK(J) = ISPSTK(I)
         ITSSTK(J) = ITSSTK(I)
         WRITE (OUNIT,2) J,DSNSTK(J),ISPSTK(J),ITSSTK(J),NAMSTK(J)
      ELSE
         WRITE (OUNIT,3) J,DSNSTK(J),NAMSTK(J)
      ENDIF
   10 CONTINUE
C
C3 -- PUT LOCATION OF SAVED STACK POSITION INTO THE BOTTOM PLACE-HOLDER,
C       AND COPY THE NAME, DESCRIPTION, (& IF 3-D: STRESS-PERIOD & TIME-STEP)
C       OF THE NEW DATA ARRAY INTO THE FIRST STACK POSITION.
      LCSTK(1)  = LCTEMP
      DSNSTK(1) = DSNNEW
      NAMSTK(1) = NAMNEW
      IF (FLAG3D) THEN
         ISPSTK(1) = ISPNEW
         ITSSTK(1) = ITSNEW
         WRITE (OUNIT,2) 1,DSNSTK(1),ISPSTK(1),ITSSTK(1),NAMSTK(1)
      ELSE
         WRITE (OUNIT,3) 1,DSNSTK(1),NAMSTK(1)
      ENDIF
      RETURN
    1 FORMAT (//,7X,A5,'-DIMENSIONAL STACK CONTENTS AFTER ',A4,
     1 ' COMMAND',//,11X,'STACK   DATA SET STRESS TIME',/,10X,
     2 'POSITION   NAME   PERIOD STEP DESCRIPTION',/)
    2 FORMAT (14X,I1,5X,A6,4X,I2,4X,I2,2X,A24)
    3 FORMAT (14X,I1,5X,A6,4X,'--',4X,'--',2X,A24)
      END
      CHARACTER*6 FUNCTION UTRMUP (STRING,LEN)
      CHARACTER STRING*6, HOLD*6
      PARAMETER (ILCA=97,ILCZ=122,IOFSET=32)
      INTRINSIC CHAR, ICHAR
C
C1 -- COPY INPUT STRING, FIND 1ST NON-BLANK CHARACTER FROM RIGHT END
      HOLD = STRING(1:LEN)
      DO 10 I=LEN,1,-1
         IF (HOLD(I:I) .NE. ' ') GOTO 20
   10 CONTINUE
         UTRMUP = ''
         RETURN
C
C2 -- FIND 1ST NON-BLANK CHARACTER FROM LEFT END, CONVERT LOWER TO UPPER-CASE
  20  L = 0
      K = 0
      DO 30 J=1,I
         IF (HOLD(J:J) .EQ. ' ') THEN
            IF (K .EQ. 0) L = J
         ELSE
            K = 1
            IC = MOD (ICHAR (HOLD(J:J) ) ,128)
            IF (IC .GE. ILCA .AND. IC .LE. ILCZ) THEN
               IC = IC - IOFSET
               HOLD (J:J) = CHAR (IC)
            ENDIF
         ENDIF
   30 CONTINUE
C
C3 -- REMOVE LEADING BLANKS AND RETURN LEFT-JUSTIFIED, UPPER-CASE STRING
      UTRMUP = HOLD (L+1:I)
      RETURN
      END

      SUBROUTINE ULC1ND (LCPT,NCOL,NCR,I,J,K)
C
C     FINDS LOCATION IN GRID FROM STORAGE LOCATION IN MATRIX
C
C1 -- CALCULATE LAYER
      IRND = MOD (LCPT,NCR)
        K  = LCPT / NCR
      IF (IRND .NE. 0) K = K + 1
C
C2 -- CALCULATE ROW
      ITOP = LCPT - (K-1) * NCR
        I  = ITOP / NCOL
C
C3 -- CALCULATE COLUMN
      IRND = MOD (ITOP,NCOL)
      IF (IRND .NE. 0) I = I + 1
        J  = LCPT - ( (K-1) * NCR + (I-1) * NCOL)
      RETURN
      END
      SUBROUTINE UPOS1G (INODE,JNODE,KNODE,NROW,NCOL,NLAY,FWD,
     1 DELC,DELR,DELL,XNODE,YNODE,ZNODE)
C
C CALCULATE ENGINEERING UNITS FROM GRID CELL LOCATION (FWD=TRUE)
C      OR:  GRID CELL LOCATION FROM ENGINEERING UNITS (FWD=FALSE)
C
C INODE = MODEL GRID COLUMN        XNODE = ENGINEERING UNITS EAST
C JNODE = MODEL GRID ROW           YNODE = ENGINEERING UNITS SOUTH
C KNODE = MODEL GRID LAYER         ZNODE = ENGINEERING UNITS DOWN
C
      DIMENSION DELC(NROW), DELR(NCOL), DELL(NLAY)
      LOGICAL FWD
C
C1 -- FORWARD CALCULATION FROM MODEL GRID TO ENGINEERING UNITS
      IF (FWD) THEN
C
C1A -- CALCULATE THE DISTANCE EAST
         XNODE = DELR (INODE) / 2.0
         DO 10 I=1,INODE-1
10          XNODE = XNODE + DELR (I)
C
C1B -- CALCULATE THE DISTANCE SOUTH
         YNODE = DELC (JNODE) / 2.0
         DO 20 I=1,JNODE-1
20          YNODE = YNODE + DELC (I)
C
C1C -- CALCULATE THE DISTANCE DOWN
         ZNODE = DELL(KNODE) / 2.0
         DO 30 I=1,KNODE-1
30          ZNODE = ZNODE + DELL (I)
C
C2 -- REVERSE CALCULATION FROM ENGINEERING UNITS TO MODEL GRID
      ELSE
C
C2A -- CALCULATE THE NUMBER OF CELLS EAST
         HOLD  = 0.
         PRIOR = 0.
         DO 110 I=1,NCOL
            POINT = HOLD + DELR (I) / 2.0
            IF (POINT .GT. XNODE) GOTO 115
            PRIOR = POINT
110         HOLD  = POINT + DELR (I) / 2.0
         INODE = NCOL
         IF (XNODE .GT. HOLD) INODE = INODE + 1
         GOTO 118
115      DIFF1 = XNODE - PRIOR
         DIFF2 = POINT - XNODE
         INODE = I
         IF (DIFF1 .LT. DIFF2 .AND. INODE .NE. 1) INODE = I - 1
C
C2B -- CALCULATE THE NUMBER OF CELLS SOUTH
118      HOLD  = 0.
         PRIOR = 0.
         DO 120 I=1,NROW
            POINT = HOLD + DELC (I) / 2.0
            IF (POINT .GT. YNODE) GOTO 125
            PRIOR = POINT
120         HOLD  = POINT + DELC (I) / 2.0
         JNODE = NROW
         IF (YNODE .GT. HOLD) JNODE = JNODE + 1
         GOTO 128
125      DIFF1 = YNODE - PRIOR
         DIFF2 = POINT - YNODE
         JNODE = I
         IF (DIFF1 .LT. DIFF2 .AND. JNODE .NE. 1) JNODE = I - 1
```

```
C
C2C -- CALCULATE THE NUMBER OF CELLS DOWN
128        HOLD  = 0.
           PRIOR = 0.
           DO 130 I=1,NLAY
              POINT = HOLD + DELL (I) / 2.0
              IF (POINT .GT. ZNODE) GOTO 135
              PRIOR = POINT
130           HOLD  = POINT + DELL (I) / 2.0
           KNODE = NLAY
           IF (ZNODE .GT. HOLD) KNODE = KNODE + 1
           RETURN
135        DIFF1 = ZNODE - PRIOR
           DIFF2 = POINT - ZNODE
           KNODE = I
           IF (DIFF1 .LT. DIFF2 .AND. KNODE .NE. 1) KNODE = I - 1
        ENDIF
        RETURN
        END

        SUBROUTINE U3DREL (DSN,KPER,KSTP,IBDCHN,BUFF,NCOL,NROW,
     1                    TEXTEX,NLAY,OUNIT,EXCMD)
C
C       3-DIMENSIONAL UNFORMATTED ARRAY READER FOR MODEL POST-PROCESSING
C
        CHARACTER DSN*6
        DOUBLE PRECISION TEXTEX, TXTFND
        DIMENSION TEXT(4),BUFF(NCOL,NROW,NLAY)
        INTEGER OUNIT
        LOGICAL EXCMD, TRIP
        EQUIVALENCE (TXTFND,TEXT(3))
C
C1 -- SET FLAG AND READ HEADER FROM FILE
        TRIP = .FALSE.
   10 READ (IBDCHN,ERR=98,END=99) IKSTP,IKPER,TEXT,INCOL,INROW,INLAY
C
C2 -- CHECK FOR ILLEGAL ARRAY SIZE WHICH COULD DESTROY STACK CONTENTS
        IF (INCOL.NE.NCOL .OR. INROW.NE.NROW .OR. INLAY.NE.NLAY) THEN
           EXCMD = .FALSE.
           WRITE (OUNIT,1) DSN,IBDCHN,INROW,INCOL,INLAY,NROW,NCOL,NLAY
     1     FORMAT (' Array size error occurred reading ',A6,' on unit',
     1     I3,/,'    SIZE READ          SIZE EXPECTED',
     2        /,' NROW NCOL NLAY      NROW NCOL NLAY',/,3I5,4X,3I5)
           RETURN
        ENDIF
C
C3 -- READ THE ARRAY INTO A BUFFER
        READ (IBDCHN,ERR=98,END=99) BUFF
C
C4 -- IF STRESS-PERIOD & TIME-STEP WERE LESS THAN THE ONE REQUESTED, OR
C         DATA SET TEXT DOES NOT AGREE WITH THE EXPECTED TEXT; THEN KEEP READING.
        IF((IKPER.LT.KPER .OR. (IKPER.EQ.KPER .AND. IKSTP.LT.KSTP)) .OR.
     1    (IKPER.EQ.KPER .AND. IKSTP.EQ.KSTP .AND. TXTFND.NE.TEXTEX))THEN
              WRITE (OUNIT,2) (TEXT(I),I=1,4),IKPER,IKSTP
     2        FORMAT (/,10X,'--> FAST-FORWARDING... ',
     1          ' found ',4A4,':  stress period',I3,', time step',I3)
              GOTO 10
        ENDIF
C
C5 -- FOUND THE CORRECT DATA SET
        IF(IKSTP.EQ.KSTP .AND. IKPER.EQ.KPER .AND. TEXTEX.EQ.TXTFND)RETURN
        IF (TRIP) THEN
```

```
C
C6 --     ALREADY REWOUND THIS UNIT ONCE, ABORT
          EXCMD = .FALSE.
          WRITE (OUNIT,3) DSN,IBDCHN,IKPER,IKSTP,KPER,KSTP
    3     FORMAT (/,' Error occurred reading ',A6,' on unit',I3,
     1     /,'    FOUND:  stress period',I3,' time step',I3,
     2     /,' EXPECTED:  stress period',I3,' time step',I3)
          RETURN
       ENDIF
C
C7 -- PAST THE REQUESTED STRESS PERIOD & TIME STEP, REWIND & TRY AGAIN
20     TRIP = .TRUE.
       WRITE (OUNIT,4) IBDCHN
    4  FORMAT (/,10X,'--> REWINDING UNIT ',I3)
       REWIND (IBDCHN)
       GOTO 10
C
   98  WRITE (OUNIT,5) IBDCHN
       EXCMD = .FALSE.
    5  FORMAT (' U3DREL:  Error reading from unit:  ',I2)
       RETURN
   99  IF (.NOT. TRIP) GOTO 20
       WRITE (OUNIT,6) IBDCHN
       EXCMD = .FALSE.
    6  FORMAT (' U3DREL:  Unexpected E-O-F on unit:  ',I2)
       RETURN
       END

       SUBROUTINE ULYREL (DSN,KPER,KSTP,IBDCHN,BUFF,NCOL,NROW,
     1                    NLAY,TEXTEX,PERTIM,TOTIM,OUNIT,EXCMD)
C
C      2-DIMENSIONAL UNFORMATTED ARRAY READER FOR MODEL POST-PROCESSING
C
       DOUBLE PRECISION TEXTEX,TXTFND
       DIMENSION TEXT(4),BUFF(NCOL,NROW)
       INTEGER OUNIT
       LOGICAL EXCMD,TRIP
       EQUIVALENCE (TXTFND,TEXT(3))
C
C1 -- SET FLAG & READ THE HEADER RECORD
       TRIP = .FALSE.
   10  READ (IBDCHN,ERR=98,END=99) IKSTP,IKPER,PERTIM,TOTIM,TEXT,INCOL,
     1    INROW,INLAY
C
C2 -- CHECK FOR ILLEGAL ARRAY SIZE...COULD DESTROY STACK CONTENTS
       IF (INCOL.GT.NCOL .OR. INROW.GT.NROW .OR. INLAY.GT.NLAY) THEN
          EXCMD = .FALSE.
          WRITE (OUNIT,1) DSN,IBDCHN,INROW,INCOL,INLAY,NROW,NCOL,NLAY
    1     FORMAT (' Array size error occurred reading ',A6,' on unit',
     1    I3,/,'   SIZE READ            SIZE EXPECTED',
     2       /,' NROW NCOL NLAY      NROW NCOL NLAY',/,3I5,4X,3I5)
          RETURN
       ENDIF
C
C3 -- READ THE ARRAY INTO A BUFFER
       READ (IBDCHN,ERR=98,END=99) BUFF
```

```
C
C4 -- CHECK IF STRESS-PERIOD & TIME-STEP WERE LESS THAN THE ONE REQUESTED, OR
C      DATA SET TEXT DOES NOT AGREE WITH THE EXPECTED TEXT; THEN KEEP READING.
      IF((IKPER.LT.KPER .OR. (IKPER.EQ.KPER .AND. IKSTP.LT.KSTP)) .OR.
     1   (IKPER.EQ.KPER .AND. IKSTP.EQ.KSTP .AND. TXTFND.NE.TEXTEX))THEN
            WRITE (OUNIT,2) (TEXT(I),I=1,4),IKPER,IKSTP
    2       FORMAT (/,10X,'--> FAST-FORWARDING... ',
     1        'FOUND ',4A4,':  STRESS PERIOD',I3,', TIME STEP',I3)
         GOTO 10
      ENDIF
C
C5 -- FOUND THE CORRECT DATA SET
      IF(IKSTP.EQ.KSTP .AND. IKPER.EQ.KPER .AND. TEXTEX.EQ.TXTFND)RETURN
      IF (TRIP) THEN
C
C6 --    ALREADY REWOUND THIS UNIT ONCE, ABORT
         EXCMD = .FALSE.
         WRITE (OUNIT,3) DSN,IBDCHN,IKPER,IKSTP,KPER,KSTP
    3    FORMAT (/,' Error occurred reading ',A6,' on unit',I3,
     1     /,'    FOUND:  stress period',I3,' time step',I3,
     2     /,'EXPECTED:  stress period',I3,' time step',I3)
      ENDIF
      RETURN
C
C7 -- PAST THE REQUESTED STRESS PERIOD & TIME STEP, REWIND AND TRY AGAIN
   20 TRIP = .TRUE.
      WRITE (OUNIT,4) IBDCHN
    4 FORMAT (/,10X,'--> REWINDING UNIT ',I3)
      REWIND (IBDCHN)
      GOTO 10
C
   98 WRITE (OUNIT,5) IBDCHN
      EXCMD = .FALSE.
    5 FORMAT (' ULYREL:  Error reading from unit:  ',I2)
      RETURN
   99 IF (.NOT. TRIP) GOTO 20
      WRITE (OUNIT,6) IBDCHN
      EXCMD = .FALSE.
    6 FORMAT (' ULYREL:  Unexpected E-O-F on unit:  ',I2)
      RETURN
      END
      SUBROUTINE UIO1FG (NPER,NLAY,NTS,IUSP,IUTS,IOFLG,ITSOC,INOC,
     1 OUNIT,EXCMD)
C
C RETRIEVE THE HEAD/DRAWDOWN SAVE FLAGS FOR A STRESS-PERIOD AND TIME-STEP
C
      INTEGER NTS(NPER),OUNIT,IOFLG(NLAY,2)
      LOGICAL EXCMD
C
C1 -- IF OUTPUT CONTROL UNIT IS ZERO, THEN HEAD AND DRAWDOWN ARE NOT AVAILABLE
      IF (INOC .EQ. 0) THEN
         WRITE (OUNIT,1)
    1    FORMAT (/,' HEAD and DRAWDOWN are not available with default'
     1             ' output control')
         EXCMD = .FALSE.
         RETURN
      ENDIF
C
C2 -- CALCULATE THE TOTAL NUMBER OF TIME STEPS FROM THE START OF THE SIMULATION
      NUMUTS = IUTS
      IF (IUSP .GT. 1) THEN
         DO 10 I=1,IUSP-1
            NUMUTS = NUMUTS + NTS(I)
10       CONTINUE
      ENDIF
C
C3 -- IS THE TIME STEP: BEHIND, AT, OR AHEAD OF THE OUTPUT CONTROL FILE POINTER
      IDIFF = NUMUTS - ITSOC
      IF (IDIFF) 100,400,200
100   CONTINUE
C
C3A --    REVERSING TO A PREVIOUS TIME-STEP
         REWIND (INOC)
         READ (INOC,2) IPAD1
         NTSADV = NUMUTS
         GOTO 300
200   CONTINUE
C
C3B --   ADVANCING TO A TIME-STEP
         NTSADV = IDIFF
         GOTO 300
C
C3C --   POSITION AND READ IOFLG FOR THE CORRECT TIME-STEP
300   CONTINUE
         ITSOC = NUMUTS
         DO 350 I=1,NTSADV
            READ (INOC,2,ERR=98,END=99) INCODE
            IF (INCODE .LT. 0) GOTO 350
            IF (INCODE .EQ. 0) THEN
               READ (INOC,3,ERR=98,END=99) IPAD1,IPAD2,IPAD3,IPAD4
               DO 310 K=1,NLAY
                  IOFLG(K,1) = IPAD3
                  IOFLG(K,2) = IPAD4
  310          CONTINUE
            ELSE
               DO 330 K=1,NLAY
                  READ (INOC,3,ERR=98,END=99)
     1                                IPAD1,IPAD2,(IOFLG(K,M), M=1,2)
  330          CONTINUE
            ENDIF
  350    CONTINUE
  400 CONTINUE
C
C3D --   POSITIONED AT THE CORRECT TIME-STEP, IOFLG IS STORED
      RETURN
```

```
C
   98 WRITE (OUNIT,998) INOC
  998 FORMAT (' UIO1FG:  Error reading from unit:  ',I2)
      STOP
   99 WRITE (OUNIT,999) INOC
  999 FORMAT (' UIO1FG:  Unexpected E-O-F on unit:  ',I2)
      STOP
    2 FORMAT (I10)
    3 FORMAT (4I10)
      END

      SUBROUTINE UXFERR (DARRAY,ARRAY,NUM)
      DOUBLE PRECISION      DARRAY
      DIMENSION ARRAY(NUM), DARRAY(NUM)
      DO 10 I = 1, NUM
         DARRAY (I) = DBLE (ARRAY (I))
10    CONTINUE
      RETURN
      END

      SUBROUTINE UXFERI (DARRAY,IARRAY,NUM)
      DOUBLE PRECISION DARRAY
      DIMENSION        DARRAY(NUM)
      INTEGER          IARRAY(NUM)
      DO 10 I = 1, NUM
         DARRAY (I) = DBLE (IARRAY (I))
10    CONTINUE
      RETURN
      END

      REAL FUNCTION UXFERD (DPHOLD)
      DOUBLE PRECISION DPHOLD
      UXFERD = DPHOLD
      RETURN
      END

      SUBROUTINE ULAPRS(BUF,TEXT,KSTP,KPER,NCOL,NROW,ILAY,IPRN,IOUT)
C
C     PRINT A 1-LAYER ARRAY IN STRIPS
C     MODIFIED TO PROVIDE 80 COLUMN FORMATS
C
      DIMENSION BUF(NCOL,NROW),TEXT(4)
C
C1 -- MAKE SURE THE FORMAT CODE (IP OR IPRN) IS BETWEEN 1 AND 20
      IP=IPRN
      IF(IP.LT.1 .OR. IP.GT.20) IP=12
C
C2 -- DETERMINE THE NUMBER OF VALUES (NCAP) PRINTED ON ONE LINE.
      IF(IP.EQ.1) NCAP=11
      IF(IP.EQ.2) NCAP=9
      IF(IP.GT.2 .AND. IP.LT.7) NCAP=15
      IF(IP.GT.6 .AND. IP.LT.12) NCAP=20
      IF(IP.EQ.12) NCAP=10
      IF(IP.GT.12) NCAP=8
C
C3 -- CALCULATE THE NUMBER OF STRIPS (NSTRIP).
      NCPF=129/NCAP
      IF (IP .GT. 12) NCPF = 9
      ISP=0
      IF(NCAP.GT.12) ISP=3
      NSTRIP=(NCOL-1)/NCAP + 1
      J1=1-NCAP
      J2=0
```

```
C
C4 -- LOOP THROUGH THE STRIPS.
      DO 2000 N=1,NSTRIP
C
C5 -- CALCULATE THE FIRST(J1) & THE LAST(J2) COLUMNS FOR THIS STRIP
      J1=J1+NCAP
      J2=J2+NCAP
      IF(J2.GT.NCOL) J2=NCOL
C
C6 -- PRINT TITLE ON EACH STRIP
      WRITE(IOUT,1) TEXT,ILAY,KSTP,KPER
    1 FORMAT(1H1,10X,4A4,' IN LAYER',I3,' AT END OF TIME STEP',I3,
     1      ' IN STRESS PERIOD',I3/11X,71('-'))
C
C7 -- PRINT COLUMN NUMBERS ABOVE THE STRIP
      CALL UCOLNO(J1,J2,ISP,NCAP,NCPF,IOUT)
C
C8 -- LOOP THROUGH THE ROWS PRINTING COLS J1 THRU J2 WITH FORMAT IP
      DO 1000 I=1,NROW
      GO TO(10,20,30,40,50,60,70,80,90,100,110,120,130,140,150,160,
     1 170,180,190,200), IP
C
C8A --   FORMAT 10G10.3
   10    WRITE(IOUT,11) I,(BUF(J,I),J=J1,J2)
   11    FORMAT(1X,I3,2X,1PG10.3,10(1X,G10.3))
         GO TO 1000
C
C8B --   FORMAT 8G13.6
   20    WRITE(IOUT,21) I,(BUF(J,I),J=J1,J2)
   21    FORMAT(1X,I3,2X,1PG13.6,8(1X,G13.6))
         GO TO 1000
C
C8C --   FORMAT 15F7.1
   30    WRITE(IOUT,31) I,(BUF(J,I),J=J1,J2)
   31    FORMAT(1X,I3,1X,15(1X,F7.1))
         GO TO 1000
C
C8D --   FORMAT 15F7.2
   40    WRITE(IOUT,41) I,(BUF(J,I),J=J1,J2)
   41    FORMAT(1X,I3,1X,15(1X,F7.2))
         GO TO 1000
C
C8E --   FORMAT 15F7.3
   50    WRITE(IOUT,51) I,(BUF(J,I),J=J1,J2)
   51    FORMAT(1X,I3,1X,15(1X,F7.3))
         GO TO 1000
C
C8F --   FORMAT 15F7.4
   60    WRITE(IOUT,61) I,(BUF(J,I),J=J1,J2)
   61    FORMAT(1X,I3,1X,15(1X,F7.4))
         GO TO 1000
C
C8G --   FORMAT 20F5.0
   70    WRITE(IOUT,71) I,(BUF(J,I),J=J1,J2)
   71    FORMAT(1X,I3,1X,20(1X,F5.0))
         GO TO 1000
C
C8H --   FORMAT 20F5.1
   80    WRITE(IOUT,81) I,(BUF(J,I),J=J1,J2)
   81    FORMAT(1X,I3,1X,20(1X,F5.1))
         GO TO 1000
```

```
C
C8I --   FORMAT 20F5.2
   90    WRITE(IOUT,91) I,(BUF(J,I),J=J1,J2)
   91    FORMAT(1X,I3,1X,20(1X,F5.2))
         GO TO 1000
C
C8J --   FORMAT 20F5.3
  100    WRITE(IOUT,101) I,(BUF(J,I),J=J1,J2)
  101    FORMAT(1X,I3,1X,20(1X,F5.3))
         GO TO 1000
C
C8K --   FORMAT 20F5.4
  110    WRITE(IOUT,111) I,(BUF(J,I),J=J1,J2)
  111    FORMAT(1X,I3,1X,20(1X,F5.4))
         GO TO 1000
C
C8L --   FORMAT 9G11.4
  120    WRITE(IOUT,121) I,(BUF(J,I),J=J1,J2)
  121    FORMAT(1X,I3,2X,1PG11.4,9(1X,G11.4))
         GO TO 1000
C
C8M --   FORMAT 8G9.0
  130    WRITE(IOUT,131) I,(BUF(J,I),J=J1,J2)
  131    FORMAT(1X,I3,1X,1PG9.0,7G9.0)
         GO TO 1000
C
C8N --   FORMAT 8G9.1
  140    WRITE(IOUT,141) I,(BUF(J,I),J=J1,J2)
  141    FORMAT(1X,I3,1X,1PG9.1,7G9.1)
         GOTO 1000
C
C8O --   FORMAT 8G9.2
  150    WRITE(IOUT,151) I,(BUF(J,I),J=J1,J2)
  151    FORMAT(1X,I3,1X,1PG9.2,7G9.2)
         GOTO 1000
C
C8P --   FORMAT 8G9.3
  160    WRITE(IOUT,161) I,(BUF(J,I),J=J1,J2)
  161    FORMAT(1X,I3,1X,1PG9.3,7G9.3)
         GOTO 1000
C
C8Q --   FORMAT 8F9.0
  170    WRITE(IOUT,171) I,(BUF(J,I),J=J1,J2)
  171    FORMAT(1X,I3,1X,8F9.0)
         GOTO 1000
C
C8R --   FORMAT 8F9.1
  180    WRITE(IOUT,181) I,(BUF(J,I),J=J1,J2)
  181    FORMAT(1X,I3,1X,8F9.1)
         GOTO 1000
C
C8S --   FORMAT 8F9.2
  190    WRITE(IOUT,191) I,(BUF(J,I),J=J1,J2)
  191    FORMAT(1X,I3,1X,8F9.2)
         GOTO 1000
C
C8T --   FORMAT 8F9.3
  200    WRITE(IOUT,201) I,(BUF(J,I),J=J1,J2)
  201    FORMAT(1X,I3,1X,8F9.3)
         GOTO 1000
C
 1000 CONTINUE
 2000 CONTINUE
      RETURN
      END
      SUBROUTINE ULAPRW(BUF,TEXT,KSTP,KPER,NCOL,NROW,ILAY,IPRN,IOUT)
C
C     PRINT 1-LAYER ARRAY
C     MODIFIED TO USE PRINT FORMAT CODES 13-20 FOR 80-COLUMN DEVICES
C
      DIMENSION BUF(NCOL,NROW),TEXT(4)
C
C1 -- PRINT A HEADER
      IF(ILAY.LE.0) GO TO 5
      WRITE(IOUT,1) TEXT,ILAY,KSTP,KPER
    1 FORMAT(1H1,10X,4A4,' IN LAYER',I3,' AT END OF TIME STEP',I3,
     1      ' IN STRESS PERIOD',I3/11X,71('-'))
C
C2 -- MAKE SURE THE FORMAT CODE (IP OR IPRN) IS BETWEEN 1 AND 20.
    5 IP=IPRN
      IF(IP.LT.1 .OR. IP.GT.20) IP=12
C
C3 -- CALL THE UTILITY MODULE UCOLNO TO PRINT COLUMN NUMBERS.
      IF(IP.EQ.1) CALL UCOLNO(1,NCOL,0,11,11,IOUT)
      IF(IP.EQ.2) CALL UCOLNO(1,NCOL,0,9,14,IOUT)
      IF(IP.GT.2 .AND. IP.LT.7) CALL UCOLNO(1,NCOL,3,15,8,IOUT)
      IF(IP.GT.6 .AND. IP.LT.12) CALL UCOLNO(1,NCOL,3,20,6,IOUT)
      IF(IP.EQ.12) CALL UCOLNO(1,NCOL,0,10,12,IOUT)
      IF(IP.GT.12) CALL UCOLNO(1,NCOL,0,8,9,IOUT)
C
C4 -- LOOP THROUGH THE ROWS PRINTING EACH ONE IN ITS ENTIRETY.
      DO 1000 I=1,NROW
      GO TO(10,20,30,40,50,60,70,80,90,100,110,120,130,140,150,160,
     1 170,180,190,200), IP
C
C4A --   FORMAT 11G10.3
   10    WRITE(IOUT,11) I,(BUF(J,I),J=1,NCOL)
   11    FORMAT(1X,I3,2X,1PG10.3,10(1X,G10.3)/(5X,11(1X,G10.3)))
         GO TO 1000
C
C4B --   FORMAT 9G13.6
   20    WRITE(IOUT,21) I,(BUF(J,I),J=1,NCOL)
   21    FORMAT(1X,I3,2X,1PG13.6,8(1X,G13.6)/(5X,9(1X,G13.6)))
         GO TO 1000
C
C4C --   FORMAT 15F7.1
   30    WRITE(IOUT,31) I,(BUF(J,I),J=1,NCOL)
   31    FORMAT(1X,I3,1X,15(1X,F7.1)/(5X,15(1X,F7.1)))
         GO TO 1000
C
C4D --   FORMAT 15F7.2
   40    WRITE(IOUT,41) I,(BUF(J,I),J=1,NCOL)
   41    FORMAT(1X,I3,1X,15(1X,F7.2)/(5X,15(1X,F7.2)))
         GO TO 1000
C
C4E --   FORMAT 15F7.3
   50    WRITE(IOUT,51) I,(BUF(J,I),J=1,NCOL)
   51    FORMAT(1X,I3,1X,15(1X,F7.3)/(5X,15(1X,F7.3)))
         GO TO 1000
C
C4F --   FORMAT 15F7.4
   60    WRITE(IOUT,61) I,(BUF(J,I),J=1,NCOL)
   61    FORMAT(1X,I3,1X,15(1X,F7.4)/(5X,15(1X,F7.4)))
         GO TO 1000
C
C4G --   FORMAT 20F5.0
   70    WRITE(IOUT,71) I,(BUF(J,I),J=1,NCOL)
   71    FORMAT(1X,I3,1X,20(1X,F5.0)/(5X,20(1X,F5.0)))
         GO TO 1000
```

```
C
C4H --   FORMAT 20F5.1
   80    WRITE(IOUT,81) I,(BUF(J,I),J=1,NCOL)
   81    FORMAT(1X,I3,1X,20(1X,F5.1)/(5X,20(1X,F5.1)))
         GO TO 1000
C
C4I --   FORMAT 20F5.2
   90    WRITE(IOUT,91) I,(BUF(J,I),J=1,NCOL)
   91    FORMAT(1X,I3,1X,20(1X,F5.2)/(5X,20(1X,F5.2)))
         GO TO 1000
C
C4J --   FORMAT 20F5.3
  100    WRITE(IOUT,101) I,(BUF(J,I),J=1,NCOL)
  101    FORMAT(1X,I3,1X,20(1X,F5.3)/(5X,20(1X,F5.3)))
         GO TO 1000
C
C4K --   FORMAT 20F5.4
  110    WRITE(IOUT,111) I,(BUF(J,I),J=1,NCOL)
  111    FORMAT(1X,I3,1X,20(1X,F5.4)/(5X,20(1X,F5.4)))
         GO TO 1000
C
C8L --   FORMAT 10G11.4
  120    WRITE(IOUT,121) I,(BUF(J,I),J=1,NCOL)
  121    FORMAT(1X,I3,2X,1PG11.4,9(1X,G11.4)/(5X,10(1X,G11.4)))
         GOTO 1000
C
C8M --   FORMAT 8G9.0
  130    WRITE(IOUT,131) I,(BUF(J,I),J=1,NCOL)
  131    FORMAT(1X,I3,1X,1PG9.0,7G9.0,/(5X,8G9.0))
         GOTO 1000
C
C8N --   FORMAT 8G9.1
  140    WRITE(IOUT,141) I,(BUF(J,I),J=1,NCOL)
  141    FORMAT(1X,I3,1X,1PG9.1,7G9.1,/(5X,8G9.1))
         GOTO 1000
C
C8O --   FORMAT 8G9.2
  150    WRITE(IOUT,151) I,(BUF(J,I),J=1,NCOL)
  151    FORMAT(1X,I3,1X,1PG9.2,7G9.2,/(5X,8G9.2))
         GOTO 1000
C
C8P --   FORMAT 8G9.3
  160    WRITE(IOUT,161) I,(BUF(J,I),J=1,NCOL)
  161    FORMAT(1X,I3,1X,1PG9.3,7G9.3,/(5X,8G9.3))
         GOTO 1000
C
C8Q --   FORMAT 8F9.0
  170    WRITE(IOUT,171) I,(BUF(J,I),J=1,NCOL)
  171    FORMAT(1X,I3,1X,8F9.0/(5X,8F9.0))
         GOTO 1000
C
C8R --   FORMAT 8F9.1
  180    WRITE(IOUT,181) I,(BUF(J,I),J=1,NCOL)
  181    FORMAT(1X,I3,1X,8F9.1/(5X,8F9.1))
         GOTO 1000
C
C8S --   FORMAT 8F9.2
  190    WRITE(IOUT,191) I,(BUF(J,I),J=1,NCOL)
  191    FORMAT(1X,I3,1X,8F9.2/(5X,8F9.2))
         GOTO 1000
C
C8T --   FORMAT 8F9.3
  200    WRITE(IOUT,201) I,(BUF(J,I),J=1,NCOL)
  201    FORMAT(1X,I3,1X,8F9.3/(5X,8F9.3))
         GOTO 1000
```

```
C
 1000 CONTINUE
      RETURN
      END

      SUBROUTINE U1DREL(A,ANAME,JJ,IN,IOUT,NOPRT)
C
C     ROUTINE TO INPUT 1-D REAL DATA MATRICES
C     MODIFIED TO ALLOW SUPPRESSION OF ALL PRINTING
C       A IS ARRAY TO INPUT
C       ANAME IS 24 CHARACTER DESCRIPTION OF A
C       JJ IS NO. OF ELEMENTS
C       IN IS INPUT UNIT
C       IOUT IS OUTPUT UNIT
C       NOPRT IS A FLAG (<0 MEANS SUPPRESS ALL PRINTING
C
      DIMENSION A(JJ),ANAME(6),FMTIN(5)
$INSERT TINY.INS
C
C1 -- READ ARRAY CONTROL RECORD.
      READ (IN,1) LOCAT,CNSTNT,FMTIN,IPRN
    1 FORMAT(I10,F10.0,5A4,I10)
C
C2 -- USE LOCAT TO SEE WHERE ARRAY VALUES COME FROM.
      IF(LOCAT.GT.0) GO TO 90
C
C3 -- IF LOCAT=0 THEN SET ALL ARRAY VALUES EQUAL TO CNSTNT. RETURN
      DO 80 J=1,JJ
   80 A(J)=CNSTNT
      IF(NOPRT.GE.0)WRITE(IOUT,3) ANAME,CNSTNT
    3 FORMAT(1X,52X,6A4,' =',G15.7)
      RETURN
C
C4 -- IF LOCAT>0 THEN READ FORMATTED RECORDS USING FORMAT FMTIN.
   90 IF (NOPRT.GE.0)WRITE(IOUT,5) ANAME,LOCAT,FMTIN
    5 FORMAT(1X,///30X,6A4,' WILL BE READ ON UNIT',I3,
     1          ' USING FORMAT: ',5A4/30X,79('-')/)
      READ (LOCAT,FMTIN) (A(J),J=1,JJ)
C
C5 -- IF CNSTNT NOT ZERO THEN MULTIPLY ARRAY VALUES BY CNSTNT.
      IF(ABS(CNSTNT).LE.TINY) GO TO 120
      DO 100 J=1,JJ
  100 A(J)=A(J)*CNSTNT
C
C6 -- IF PRINT CODE (IPRN) =>0 THEN PRINT ARRAY VALUES.
  120 IF(IPRN.LT.0.OR.NOPRT.LT.0) RETURN
      WRITE(IOUT,1001) (A(J),J=1,JJ)
 1001 FORMAT((1X,1PG12.5,9(1X,G12.5)))
      RETURN
      END
      SUBROUTINE U2DREL(A,ANAME,II,JJ,K,IN,IOUT,NOPRT)
C
C     ROUTINE TO INPUT 2-D REAL DATA MATRICES
C     MODIFIED TO ALLOW SUPPRESSION OF PRINTING
C       ANAME IS 24 CHARACTER DESCRIPTION OF A
C       II IS NO. OF ROWS
C       JJ IS NO. OF COLS
C       K IS LAYER NO. (USED WITH NAME TO TITLE PRINTOUT UNLESS K IS 0)
C       IN IS INPUT UNIT
C       IOUT IS OUTPUT UNIT
C       NOPRT IS A FLAG (<0 MEANS SUPPRESS ALL PRINTING)
C
      DIMENSION A(JJ,II),ANAME(6),FMTIN(5)
$INSERT TINY.INS
C
C1 -- READ ARRAY CONTROL RECORD.
      READ (IN,1) LOCAT,CNSTNT,FMTIN,IPRN
    1 FORMAT(I10,F10.0,5A4,I10)
C
C2 -- USE LOCAT TO SEE WHERE ARRAY VALUES COME FROM.
      IF(LOCAT) 200,50,90
C
C3 -- IF LOCAT=0 THEN SET ALL ARRAY VALUES EQUAL TO CNSTNT. RETURN
   50 DO 80 I=1,II
      DO 80 J=1,JJ
   80 A(J,I)=CNSTNT
      IF(K.GT.0.AND.NOPRT.GE.0) WRITE(IOUT,2) ANAME,CNSTNT,K
    2 FORMAT(1X,52X,6A4,' =',G15.7,' FOR LAYER',I3)
      IF(K.LE.0.AND.NOPRT.GE.0) WRITE(IOUT,3) ANAME,CNSTNT
    3 FORMAT(1X,52X,6A4,' =',G15.7)
      RETURN
C
C4 -- IF LOCAT>0 THEN READ FORMATTED RECORDS USING FORMAT FMTIN.
   90 IF(K.GT.0.AND.NOPRT.GE.0) WRITE(IOUT,4) ANAME,K,LOCAT,FMTIN
    4 FORMAT(1X,///30X,6A4,' FOR LAYER',I3,' WILL BE READ ON UNIT',
     1        I3,' USING FORMAT: ',5A4/30X,96('-'))
      IF(K.LE.0.AND.NOPRT.GE.0) WRITE(IOUT,5) ANAME,LOCAT,FMTIN
    5 FORMAT(1X,///30X,6A4,' WILL BE READ ON UNIT',
     1        I3,' USING FORMAT: ',5A4/30X,83('-'))
      DO 100 I=1,II
      READ (LOCAT,FMTIN) (A(J,I),J=1,JJ)
  100 CONTINUE
      GO TO 300
C
C5 -- LOCAT<0 THEN READ UNFORMATTED RECORD CONTAINING ARRAY VALUES
  200 LOCAT=-LOCAT
      IF(K.GT.0.AND.NOPRT.GE.0) WRITE(IOUT,201) ANAME,K,LOCAT
  201 FORMAT(1X,///30X,6A4,', LAYER',I3,
     1     ' WILL BE READ UNFORMATTED ON UNIT',I3/30X,73('-'))
      IF(K.LE.0.AND.NOPRT.GE.0) WRITE(IOUT,202) ANAME,LOCAT
  202 FORMAT(1X,///30X,
     1     ' WILL BE READ UNFORMATTED ON UNIT',I3/30X,60('-'))
      READ(LOCAT)
      READ(LOCAT) A
C
C6 -- IF CNSTNT NOT ZERO THEN MULTIPLY ARRAY VALUES BY CNSTNT.
  300 IF(ABS(CNSTNT).LE.TINY) GO TO 320
      DO 310 I=1,II
      DO 310 J=1,JJ
      A(J,I)=A(J,I)*CNSTNT
  310 CONTINUE
```

```
C
C7 -- IF PRINT CODE (IPRN) =>0 THEN PRINT ARRAY VALUES.
  320 IF(IPRN.LT.0.OR.NOPRT.LT.0) RETURN
      CALL ULAPRW(A,ANAME,0,0,JJ,II,0,IPRN,IOUT)
      RETURN
      END

      SUBROUTINE U2DINT(IA,ANAME,II,JJ,K,IN,IOUT,NOPRT)
C
C     ROUTINE TO INPUT 2-D INTEGER DATA MATRICES
C     MODIFIED TO ALLOW SUPPRESSION OF PRINTING
C       IA IS ARRAY TO INPUT
C       ANAME IS 24 CHARACTER DESCRIPTION OF IA
C       II IS NO. OF ROWS
C       JJ IS NO. OF COLS
C       K IS LAYER NO. (USED WITH NAME TO TITLE PRINTOUT UNLESS K IS 0)
C       IN IS INPUT UNIT
C       IOUT IS OUTPUT UNIT
C       NOPRT IS A FLAG (<0 MEANS SUPPRESS ALL PRINTING)
C
      DIMENSION IA(JJ,II),ANAME(6),FMTIN(5)
C
C1 -- READ ARRAY CONTROL RECORD.
      READ (IN,1) LOCAT,ICONST,FMTIN,IPRN
    1 FORMAT(I10,I10,5A4,I10)
C
C2 -- USE LOCAT TO SEE WHERE ARRAY VALUES COME FROM.
      IF(LOCAT) 200,50,90
C
C3 -- IF LOCAT=0 THEN SET ALL ARRAY VALUES EQUAL TO ICONST. RETURN
   50 DO 80 I=1,II
      DO 80 J=1,JJ
   80 IA(J,I)=ICONST
      IF(K.GT.0.AND.NOPRT.GE.0) WRITE(IOUT,2) ANAME,ICONST,K
    2 FORMAT(1X,52X,6A4,' =',I15,' FOR LAYER',I3)
      IF(K.LE.0.AND.NOPRT.GE.0) WRITE(IOUT,3) ANAME,ICONST
    3 FORMAT(1X,52X,6A4,' =',I15)
      RETURN
C
C4 -- IF LOCAT>0 THEN READ FORMATTED RECORDS USING FORMAT FMTIN.
   90 IF(K.GT.0.AND.NOPRT.GE.0) WRITE(IOUT,4) ANAME,K,LOCAT,FMTIN
    4 FORMAT(1X,///30X,6A4,' FOR LAYER',I3,' WILL BE READ ON UNIT',
     1       I3,' USING FORMAT: ',5A4/30X,96('-'))
      IF(K.LE.0.AND.NOPRT.GE.0) WRITE(IOUT,5) ANAME,LOCAT,FMTIN
    5 FORMAT(1X,///30X,6A4,' WILL BE READ ON UNIT',
     1         I3,' USING FORMAT: ',5A4/30X,83('-'))
      DO 100 I=1,II
      READ (LOCAT,FMTIN) (IA(J,I),J=1,JJ)
  100 CONTINUE
      GO TO 300
C
C5 -- LOCAT<0 THEN READ UNFORMATTED RECORD CONTAINING ARRAY VALUES
  200 LOCAT=-LOCAT
      IF(K.GT.0.AND.NOPRT.GE.0) WRITE(IOUT,201) ANAME,K,LOCAT
  201 FORMAT(1X,///30X,6A4,', LAYER',I3,
     1     ' WILL BE READ UNFORMATTED ON UNIT',I3/30X,73('-'))
      IF(K.LE.0.AND.NOPRT.GE.0) WRITE(IOUT,202) ANAME,LOCAT
  202 FORMAT(1X,///30X,6A4,
     1     ' WILL BE READ UNFORMATTED ON UNIT',I3/30X,60('-'))
      READ(LOCAT)
      READ(LOCAT) IA
```

```
C
C6 -- IF ICONST NOT ZERO THEN MULTIPLY ARRAY VALUES BY ICONST.
  300 IF(ICONST.EQ.0) GO TO 320
      DO 310 I=1,II
      DO 310 J=1,JJ
      IA(J,I)=IA(J,I)*ICONST
  310 CONTINUE
C
C7 -- IF PRINT CODE (IPRN) =>0 THEN PRINT ARRAY VALUES.
  320 IF(IPRN.LT.0.OR.NOPRT.LT.0) RETURN
      IF(IPRN.GT.5) IPRN=0
      IPRN=IPRN+1
C
C8 -- PRINT COLUMN NUMBERS AT TOP OF PAGE.
      IF(IPRN.EQ.1) CALL UCOLNO(1,JJ,0,10,12,IOUT)
      NL=125/IPRN/5*5
      IF(IPRN.GT.1) CALL UCOLNO(1,JJ,4,NL,IPRN,IOUT)
C
C9 -- PRINT EACH ROW IN THE ARRAY.
      DO 110 I=1,II
C
C10 -- SELECT THE FORMAT
        GO TO(101,102,103,104,105,106), IPRN
C
C10A -- FORMAT 10I11
  101   WRITE(IOUT,1001) I,(IA(J,I),J=1,JJ)
 1001   FORMAT(1X,I3,2X,I11,9(1X,I11)/(5X,10(1X,I11)))
        GO TO 110
C
C10B -- FORMAT 60I1
  102   WRITE(IOUT,1002) I,(IA(J,I),J=1,JJ)
 1002   FORMAT(1X,I3,1X,60(1X,I1)/(5X,60(1X,I1)))
        GO TO 110
C
C10C -- FORMAT 40I2
  103   WRITE(IOUT,1003) I,(IA(J,I),J=1,JJ)
 1003   FORMAT(1X,I3,1X,40(1X,I2)/(5X,40(1X,I2)))
        GO TO 110
C
C10D -- FORMAT 30I3
  104   WRITE(IOUT,1004) I,(IA(J,I),J=1,JJ)
 1004   FORMAT(1X,I3,1X,30(1X,I3)/(5X,30(1X,I3)))
        GO TO 110
C
C10E -- FORMAT 25I4
  105   WRITE(IOUT,1005) I,(IA(J,I),J=1,JJ)
 1005   FORMAT(1X,I3,1X,25(1X,I4)/(5X,25(1X,I4)))
        GO TO 110
C
C10F -- FORMAT 20I5
  106   WRITE(IOUT,1006) I,(IA(J,I),J=1,JJ)
 1006   FORMAT(1X,I3,1X,20(1X,I5)/(5X,20(1X,I5)))
  110   CONTINUE
      RETURN
      END
```

Attachment C--Vector program listing.

```
/* VECTOR.AML -- Create ARC/INFO coverage of flow vectors from the
/*                Modular Model Statistical Processor program,
/*                VECT command output file.
&ARGS GEN.FILE~
      COVERAGE~
      NO.ROWS~
      NO.COLS

&TYPE '[VECTOR Revision 1.0]'
&TYPE

/* TEST ARGUMENTS FOR ERRORS -----------------------------------------------------
&S ERR :=
&IF [NULL %GEN.FILE%] | [NULL %COVERAGE%] &THEN ~
    &S ERR := 'Missing argument'
&IF [TYPE %NO.ROWS%] ^= -1 | [TYPE %NO.COLS%] ^= -1 &THEN ~
    &S ERR := 'Nrow and Ncol must be positive integers'
&IF %NO.ROWS% <= 0 | %NO.COLS% <= 0 &THEN ~
    &S ERR := 'Nrow and Ncol must be positive integers'
&IF [EXISTS %COVERAGE% -FILE] | [EXISTS %COVERAGE% -DIR] | ~
    [EXISTS %COVERAGE% -COVERAGE] &THEN ~
    &S ERR := %COVERAGE%' already exists.'
&IF ^ [EXISTS %GEN.FILE% -FILE] &THEN ~
    &S ERR := %GEN.FILE%' does not exist.'
&IF ^ [NULL %ERR%] &THEN &GOTO HELP

/* SET METHOD FOR INVOKING ARCNUM LOAD MODULE ON THIS COMPUTER SYSTEM --------
&S OS.COMMAND := RESUME ARCNUM.RUN

/* CREATE THE COVERAGE AND GENERATE LINE TOPOLOGY ----------------------------
GENERATE %COVERAGE%
INPUT %GEN.FILE%
LINES
QUIT

BUILD %COVERAGE% LINE

/* ADD AND DEFINE ITEMS FOR ROW, COLUMN, AND LAYER ---------------------------
ADDITEM %COVERAGE%.AAT %COVERAGE%.AAT ROW    4 4 I
ADDITEM %COVERAGE%.AAT %COVERAGE%.AAT COLUMN 4 4 I
ADDITEM %COVERAGE%.AAT %COVERAGE%.AAT LAYER  4 4 I

&DATA %OS.COMMAND%
[DIR [PATHNAME *]]>INFO
%COVERAGE%
%NO.ROWS%
%NO.COLS%
&END

&RETURN

&LABEL HELP
&TYPE %ERR%
&TYPE
&TYPE 'Usage:  &RUN VECTOR Vect_file Coverage Nrow Ncol'
&TYPE
&TYPE 'where:  Vect_file = name of file created by MMSP VECT command'
&TYPE '        Coverage  = name of the line coverage to be created'
&TYPE '          Nrow    = number of rows in the model simulation grid'
&TYPE '          Ncol    = number of columns in the model simulation grid'
&TYPE
&RETURN
```

```
      PROGRAM ARCNUM
C
C CALCULATE ROW, COLUMN, & LAYER FROM GRID CELL LOCATION IN FLOW VECTOR COVERAGE
C JONATHON SCOTT, SEPTEMBER 30, 1988
C
      CHARACTER  DIRECT*128, ARCCOV*32, FILE*32
      CHARACTER  USER*4,     STRING*16, NAMITM*16
C
      INTEGER    IPNTAR(4),  IROWAR(4), ICOLAR(4), ILAYAR(4)
      INTEGER    INREC(1024)
      DOUBLE PRECISION REALLY
C
      PARAMETER (USER='ARC',STRING='')
      PARAMETER (ITWO=2)
C
C1 -- INITIALIZE ARC LIBRARY ROUTINES
      CALL LUNINI
      CALL MINIT
      CALL INFINT
C
C2 -- GET PATHNAME, COVERAGE NAME, & NUMBER OF ROWS / COLUMNS IN THE MODEL GRID
      READ (*,'(A)') DIRECT, ARCCOV
      READ (*,*) NROW,NCOL
      NCR = NROW * NCOL
C
C3 -- OPEN THE AAT FOR THE MODEL COVERAGE & DEFINE ITEM MATRICIES
      FILE = ARCCOV (1:LENGTH(ARCCOV,32)) // '.AAT'
      CALL INFOPN (FILE,DIRECT,USER,ITWO,NFARC,NUMARC,IARCLN,IER)
      IF (IER .NE. 0) THEN
          WRITE (*,'(2A)') 'UNABLE TO OPEN AAT FOR: ',ARCCOV
          GOTO 991
      ENDIF
C
C4 -- EXTRACT THE ARRAY DESCRIBING THE COVER-ID ITEM
      NAMITM = ARCCOV (1:LENGTH (ARCCOV,32)) // '-ID'
      CALL INFEXI (NFARC,NAMITM,IPNTAR,IEXIST)
      IF (IEXIST .NE. 1) THEN
          WRITE (*,'(A,I4)') 'ERROR DURING INFEXI OF -ID',IER
          GOTO 991
      ENDIF
C
C5 -- EXTRACT THE ARRAY DESCRIBING THE ROW ITEM
      NAMITM = 'ROW'
      CALL INFEXI (NFARC,NAMITM,IROWAR,IEXIST)
      IF (IEXIST .NE. 1) THEN
          WRITE (*,'(A,I4)') 'ERROR DURING INFEXI OF ROW',IER
          GOTO 991
      ENDIF
C
C6 -- EXTRACT THE ARRAY DESCRIBING THE COLUMN ITEM
      NAMITM = 'COLUMN'
      CALL INFEXI (NFARC,NAMITM,ICOLAR,IEXIST)
      IF (IEXIST .NE. 1) THEN
          WRITE (*,'(A,I4)') 'ERROR DURING INFEXI OF COLUMN',IER
          GOTO 991
      ENDIF
C
C7 -- EXTRACT THE ARRAY DESCRIBING THE LAYER ITEM
      NAMITM = 'LAYER'
      CALL INFEXI (NFARC,NAMITM,ILAYAR,IEXIST)
      IF (IEXIST .NE. 1) THEN
          WRITE (*,'(A,I4)') 'ERROR DURING INFEXI OF LAYER',IER
          GOTO 991
      ENDIF
```

```
C
C8 -- LOOP THROUGH THE ARCS, COMPUTING & WRITING ROW, COLUMN, & LAYER
      DO 20 IPNT=1,NUMARC
C
C8A --     RETRIEVE A RECORD FROM THE ARC ATTRIBUTE TABLE
          IRECNO = IPNT
          CALL INFGET (NFARC,IRECNO,INREC,IER)
          IF (IER .NE. 0) THEN
              WRITE (*,'(A,I4)') 'ERROR DURING INFGET ',IER
              GOTO 991
          ENDIF
C
C8B --     EXTRACT THE COVER-ID DATA FROM THE RECORD
          CALL INFDEC (INREC,IARCLN,IPNTAR,REALLY,STRING,IER)
          IF (IER .NE. 0) THEN
              WRITE (*,'(A,I4)') 'ERROR DURING INFDEC ',IER
              GOTO 991
          ENDIF
C
C8C --     CALCULATE THE ROW, COLUMN, & LAYER LOCATION
          LCPT = IDINT(REALLY)
          CALL ULC1ND (LCPT,NCOL,NCR,I,J,K)
C
C8D --     ENCODE ON THE RECORD THE ROW, COLUMN, & LAYER
          REALLY = DBLE(I)
          CALL INFENC (IROWAR,REALLY,STRING,IARCLN,INREC,IER)
          IF (IER .NE. 0) THEN
              WRITE (*,'(A,I4)') 'ERROR DURING INFENC OF ROW ',IER
              GOTO 991
          ENDIF
          REALLY = DBLE(J)
          CALL INFENC (ICOLAR,REALLY,STRING,IARCLN,INREC,IER)
          IF (IER .NE. 0) THEN
              WRITE (*,'(A,I4)') 'ERROR DURING INFENC OF COLUMN ',IER
              GOTO 991
          ENDIF
          REALLY = DBLE(K)
          CALL INFENC (ILAYAR,REALLY,STRING,IARCLN,INREC,IER)
          IF (IER .NE. 0) THEN
              WRITE (*,'(A,I4)') 'ERROR DURING INFENC OF LAYER ',IER
              GOTO 991
          ENDIF
C
C8E --     WRITE THE UPDATED RECORD TO THE ARC ATTRIBUTE TABLE
          IRECNO = IPNT
          CALL INFPUT (NFARC,IRECNO,INREC,IER)
          IF (IER .NE. 0) THEN
              WRITE (*,'(A,I4)') 'ERROR DURING INFPUT ',IER
              GOTO 991
          ENDIF
20    CONTINUE
C
C9 -- FINISHED WITH ALL THE ARCS
      WRITE (*,'(A,I4)') 'Number of arcs defined = ',NUMARC
991   CALL INFCLS (NFARC)
      STOP
      END
```

```
C
      INTEGER FUNCTION LENGTH (STRING,LEN)
C
C  FUNCTION TO DETERMINE THE LENGTH OF A CHARACTER VARIABLE
C
      CHARACTER*(*) STRING
      DO 10 I=LEN,1,-1
10      IF (STRING(I:I) .NE. ' ') GOTO 20
      I = 0
20    LENGTH = I
      RETURN
      END
C
      SUBROUTINE ULC1ND (LCPT,NCOL,NCR,I,J,K)
C
C     FINDS LOCATION IN GRID FROM STORAGE LOCATION IN MATRIX
C
      IREM = MOD (LCPT,NCR)
        K  = LCPT / NCR
      IF (IREM .NE. 0) K = K + 1
      ITOP = LCPT - (K-1) * NCR
        I  = ITOP / NCOL
      IREM = MOD (ITOP,NCOL)
      IF (IREM .NE. 0) I = I + 1
        J  = LCPT - ( (K-1) * NCR + (I-1) * NCOL)
      RETURN
      END
```

APPENDIX
ABBREVIATED INPUT INSTRUCTIONS

The instructions supplied here are intended as a quick reference for experienced users. Detailed information for each command is provided in chapter 3. The command descriptions in this appendix provide terse explanations of the data requirements for each command. Acronyms are used with each command description to provide a consistent, abbreviated description of the required format. The explanations of the acronyms are provided in alphabetical order after the command listing printed below.

```
Starting
  Column:  1    6
  Fields:  **** COMMENT
```

```
Starting
  Column:  1    6
  Fields:  TITL TITLE
```

```
Starting
  Column:  1    6   13   16   18   20 23 26
  Fields:  READ DSN UNIT NDIM TYPE SP TS ARRAY-NAME
```

when reading histogram cut points

```
Starting
  Column:  1    6     13   16
  Fields:  READ CLASS UNIT NCUT
```

```
Starting
  Column:  1    6   13    16   23       27
  Fields:  PRIN DSN LAYER MASK FMT-CODE MISSING-VALUES
```

```
Starting
  Column:  1    6   13    16   23   26       30
  Fields:  WRIT DSN LAYER MASK UNIT FMT-CODE MISSING-VALUES
```

```
Starting
  Column:  1    6   13    16
  Fields:  STAT DSN LAYER MASK
```

```
Starting
  Column:  1    6   13    16   23
  Fields:  HIST DSN LAYER MASK NCLASS
```

```
Starting
  Column:  1    6    13    16       19   26    29   36
  Fields:  COMP DSN1 LAYER OPERATOR DSN2 LAYER MASK LIMIT
```

```
Starting
 Column:  1    6    13    16       19   26    29   36
 Fields:  MATH DSN1 LAYER OPERATOR DSN2 LAYER DSN3 ARRAY-NAME
```

```
Starting
 Column:  1    6    9   13  17  21  25  29  33  37  41  45  49  53  57  61  65  69  73  77
 Fields:  HEAD UNIT ROW COL LAY ROW COL LAY ROW COL LAY ROW COL LAY ROW COL LAY ROW COL LAY
```

```
Starting
 Column:  1    6  9
 Fields:  REBO SP TS
```

```
Starting
 Column:  1    6         16        26        36        46        56        66
 Fields:  THIC THICK1    THICK2    THICK3    THICK4    THICK5    THICK6    THICK7
```

```
Starting
 Column:  1    6   10  14  18  22  26  30  34  37
 Fields:  SLIC ROW COL LAY ROW COL LAY ROW COL LAY
```

```
Starting
 Column:  1    6    9           15
 Fields:  VECT UNIT ORIENTATION SCALE-FACTOR
```

Acronym Explanations

ARRAY-NAME: is a 24-character name given to a data array. ARRAY-NAME is not explicitly used by the MMSP program, but is printed to identify the contents of a data array.

COL: is a three-digit integer specifying a column number in the simulation grid.

COMMENT: is a 75-character field used for internal documentation of an MMSP command file.

DSN, DSN1, DSN2, DSN3: is a six-character data set name given to a two- or three-dimensional data array. Some DSNs are reserved for particular data arrays. The reserved DSNs are listed below, see table 7.

STRT	SC1	TOP	BOT
WELL	RECH	RECHF	AREA
HEAD	DRAWDN	STORAG	CNHEAD
RIFACE	FRFACE	LOFACE	CBCRIV
CBCRCH	CBCWEL	CBCDRN	CBCEVT
CBCGHB	UBOUND	CLASS	

FMT-CODE: is a two-digit integer identifying a Fortran format to use when printing or writing a data array.

FMT-CODE	FORTRAN FORMAT	FMT-CODE	FORTRAN FORMAT	FMT-CODE	FORTRAN FORMAT
0	(10G11.4)	7	(20F 5.0)	14	(8G 9.1)
1	(11G10.3)	8	(20F 5.1)	15	(8G 9.2)
2	(9G13.6)	9	(20F 5.2)	16	(8G 9.3)
3	(15F 7.1)	10	(20F 5.3)	17	(8F 9.0)
4	(15F 7.2)	11	(20F 5.4)	18	(8F 9.1)
5	(15F 7.3)	12	(10G11.4)	19	(8F 9.2)
6	(15F 7.4)	13	(8G 9.0)	20	(8F 9.3)

LAYER: is a two-digit integer specifying a layer number of the simulation grid. When LAYER is zero, all layers are processed by the command. When LAYER is specified for a two-dimensional data array, the field identifies which boundary-array layer to use for masking the data array.

LAY: is a three-digit integer indicating a layer in the simulation grid.

LIMIT: is a four-digit integer controlling the maximum number of nodes that will be printed when a COMP command is performed and the comparison condition is true.

MASK: consists of three two-digit mask fields. Specifying a non-zero number in the first mask field causes zeroes in the data array to be masked. The second mask field controls the model-boundary mask as follows.

Mask field contents	Nodes masked
-3	Active and constant head nodes
-2	Inactive and active nodes
-1	Active nodes
0	None
1	Inactive and constant head nodes
2	Inactive nodes
3	Constant head nodes

The third mask field controls the user-boundary mask. The user-boundary must have been previously defined with the READ or SLIC command prior to using a user-boundary mask. Specifying a positive number causes values in the data array to be masked when the corresponding node in the user-boundary array is less than or equal to zero. Specifying a negative number masks the complementary set of nodes. Specifying a zero disables the user-boundary mask.

MISSING-VALUES: are three real numbers placed in a 30-column field, according to the Fortran format (3F10.0). The three numbers over-ride the default missing values shown in table 1 when executing the WRIT or PRIN commands. The first missing value is written by these commands when a value in the data array is masked by the zero mask, the second missing value is written when a value is masked by the model boundary mask, and the third missing value is written when a value is masked by the user boundary mask.

NCLASS: is a two-digit integer specifying the number of frequency classes. When between -3 and -20, classes are computed by MMSP using logarithmic scaling. When between 3 and 20, classes are computed by MMSP using arithmetic scaling. Specifying NCLASS as a number between -3 and 3, causes MMSP to use cut points for frequency classes provided previously with the READ command.

NCUT: is a two-digit integer specifying the number of cut points to be read for later use with the HIST command.

NDIM: is a one-digit integer (either 2 or 3) that specifies the number of dimensions when reading data arrays. NDIM is ignored when a reserved DSN is used.

OPERATOR: is a two-character field indicating which mathematical operation or logical comparison to perform with the MATH or COMP command. Each command has different operators as shown below.

MATH command operators	COMP command operators
+ or AD	= or EQ
- or SU	^= or NE
* or MU	< or LT
/ or DI	> or GT
** or EX	<= or =< or LE
|| or AB	>= or => or GE

ORIENTATION: is a five-character field specifying a viewing orientation for an oblique slice of the simulation grid. ORIENTATION must be one of "TOP", "SIDE", or "FRONT".

ROW: is a three-digit integer that specifies a row of the simulation grid.

SCALE-FACTOR: is a scale factor used to modify computed vector lengths. When positive, the vector lengths are increased by multiplying by the SCALE-FACTOR. When negative, the vector lengths are computed by multiplying the vector lengths by the smallest power of ten necessary to make the shortest vector length greater than one, taking the logarithm of the product, and multiplying the result by the absolute value of the SCALE-FACTOR.

SP: is a two-digit integer that specifies a stress-period number.

THICK: is a 70-column field used for entering as many as seven layer thicknesses according to the Fortran format (7F10.0). The THIC command should be repeated as many times as necessary to enter a thickness value for every layer in the simulation grid. Thicknesses are used only for determining the coordinates for flow vectors displayed on a plane that slices more than one layer of the simulation grid.

TITLE: is a 75-character field used for altering the fourth title line.

TS: is a two-digit integer that specifies a time-step number.

TYPE: is a character (either "R" or "I") that specifies whether real or integer data are being read. TYPE is ignored when reserved data set names are used.

UNIT: is a two-digit integer identifying the Fortran unit number that has been opened by job control statements that precede the running of the MMSP program. The unit number corresponds to a file that will be read or written by the MMSP program.