SIMCON

A Simulation Control Language at Oregon State University

Explore realms of wonder and Adventure

ERIC L. BEALS
Department of Fisheries and Wildlife
Oregon State University

1981

SIMCON

A Simulation Control Language

July, 1981

E. L. Beals
Department of Fisheries and Wildlife
Oregon State University
Corvallis, Oregon

Sea Grant Publication No. ORESU-H-81-001

Preface.

This is the first of two documents describing the SIMCON system and is further divided into three parts. Part 1 is directed toward all users of SIMCON and contains a general introduction and description of the SIMCON system and a user's guide to running a model under SIMCON. Users need not be familiar with the FORTRAN programming language to read and use this part. Part 2 is a modeler's guide to SIMCON and is designed specifically to aid the FORTRAN programmer construct models utilizing the SIMCON language. Part 3 contains a list of run-time error messages SIMCON is capable of generating. The errors are described and possible solutions to the errors are discussed.

Address comments and requests for documentation to:

Eric Rexstad
Department of Fisheries and Wildlife
Oregon State University
Corvallis, Oregon 97331
(503) 754-4531

Acknowledgments.

The SIMCON system was originally conceived and designed at the Institute of Animal Resource Ecology, University of British Columbia, Vancouver, Canada by Ray Hilborn, Jeffrey Stander, and William Webb. Excerpts are taken liberally from a paper published in Simulation, vol. 20 (May 1973), 172-175 by Ray Hilborn entitled A control system for FORTRAN simulation programming in the Introduction and System Description sections in Part I and the System Architecture section in Part II of this document.

The SIMCON program was obtained by Dr. A.V. Tyler at Oregon State University, Corvallis through the courtesy of the Institute of Animal Resource Ecology and adapted to a CDC CYBER 70 model 73 computer in 1976. Special acknowledgments are due to William Webb who was the author of the first document on SIMCON at Oregon State University and was responsible for program maintenance and upgrades until 1978, and the Sea Grant Program which funded the initial conversion (under the Pleuronectid Project, Sea Grant no. 04-8-M01-144).

SIMCON Error Messages - pages 47-59, was authored by Eric A. Rexstad, Department of Fisheries and Wildlife, Oregon State University, Corvallis.

This work is a result of research sponsored by NOAA Office of Sea Grant, Department of Commerce, under Grant No. NA79AA-D-00106 (Project No. R/OPF-1). The U.S. Government is authorized to produce and distribute reprints for governmental purposes notwithstanding any copyright notation that may appear hereon.

Table of Contents

Part I A User's Guide to the SIMCON Language

Introduction 2
The SIMCON System 3
Summary of Commands 5
Basic SIMCON Commands 6
System Parameters 11
Getting Started 13
Obtaining Output Hardcopy from the CDC CYBER 15
Advanced SIMCON Commands and Features 16
SIMCON Macros 21
Command Files 25
SIMCON Comments 26
SIMCON System Commands 27
Miscellaneous Features 29

Part II The FORTRAN Programmer's Guide to SIMCON

System Architecture 31
Programming Requirements 32
Optional User Routines 36
SIMCON Files and Unit Assignments 39
SIMCON Loading and Execution 40
Program Debugging 41
Conditional Attention Interrupts 42
The SIMCON Utility Library 43

Part III Error Messages 46

Part I. A User's Guide to the SIMCON Language

Introduction.

This guide shows how to utilize the SIMCON command language to control and monitor FORTRAN models previously constructed. Since the majority of programs will probably be simulation models, the illustrations in this manual will be drawn from simulation models now operating under SIMCON. This guide is mostly concerned with descriptions of the SIMCON system commands but also included are sections on how to log on to the CYBER, modes of output, and how to save and print output. Users are also referred to Part II of this document, The FORTRAN Programmer's Guide to SIMCON, which describes the system architecture, programming requirements and restrictions, and the SIMCON utility library.

The SIMCON System.

Computer based simulation modeling often times requires real-time interaction between the model user and the model itself. A major portion of programming effort in interactive digital simulation is usually involved with methods of altering and viewing variables as the simulation proceeds and with obtaining graphical or other forms of output from the program. The SIMCON system has been made completely independent of the model and is capable of handling all input, output, and intervention needs. This can reduce programming and debugging time significantly. Also, any previously programmed self-contained model can be run on the SIMCON system with a minimum of alteration and effort.

The SIMCON system is primarily useful with models that simulate discrete time systems. Its application is thus different from the event oriented simulation systems such as GASP or SIMSCRIPT and continuous system simulation aids. All user programming is done in FORTRAN. The user interacts with the program through very simple commands, of which only two are sufficient to run a model and view its results.

The SIMCON system was conceived and developed with three goals in mind:

1. Selected variables can be viewed as simulation is proceeding.
2. The simulation may be stopped at any point, any variable displayed or modified, then the simulation resumed from the point where it was left.
3. Any variable can be viewed after simulation is complete.

Features of the SIMCON system are:

1. SIMCON contains a set of commands which provide for user control and monitoring functions.
2. Variables may be displayed or modified by name (the SET and DISPLAY commands).
3. Selected variables may be viewed during simulation (PLOT and UNPLOT commands).
4. Any variable may be viewed after simulation (PRINT, VIEW, and GRAPH commands).
5. Since FORTRAN variables are accessible by name, SIMCON is very handy for debugging programs.

6. Printer and line plotter displays may be obtained on any of the remote terminals supported by CYBER as well as the central high speed line printer. Graphics displays can be produced on any remote graphics display device supported by the CYBER COMPLOT graphics package such as any Tektronix terminal or plotter and also the Gerber flatbed plotter.

7. At the option of the user, SIMCON has the capability to restore the state of the model to any previous point in simulated time (the TIME command). For example, while simulating from the year 1980 to the year 2000, the user may stop at 1990 and explore several alternatives from 1990 to 2000 without having to restart the model from 1980 each time. The user may also return to the initial state of the model from any point without having to use expensive FORTRAN input/output to reread initial values.

8. Beginning and ending time periods are easily specified for a simulation run (the SIMULATE command).

9. Synonyms may be defined for FORTRAN variable names. For example, the variable BARO can be referred to by the more descriptive name BAROMETRIC-PRESSURE.

10. A series of SIMCON commands may be stored on a file and executed as a group (BATCH files). A BATCH file is a convenient way to initialize variables.

11. SIMCON commands may be combined into groups called macros. These macros can be created and executed interactively to automatically perform combinations of functions and can be conveniently maintained on a permanent library file.

12. Commands may be defined to intervene at a specified point in simulated time. Such a command may automatically alter a variable at a given point in the simulation (AT commands).

13. The SIMCON system has the ability to call user provided subroutines upon command. The programmer may write his or her own special output routines or provide instructions or promptings to novice users.

A Summary of Commands and Abbreviations.

Command	Abbr.	Description	Page
AT	A	Specifies a future intervention.	9
CALL	CA	Calls a SIMCON macro explicitly.	21
CLEAR	CL	Clears the output work area.	17
COMMAND	COM	Defines an alternate name for a SIMCON or user command.	19
CONTINUE	C	Resumes the simulation.	9
DISPLAY	D	Displays the value of a variable.	7
FILE	F	Writes output onto a file.	16
GET	GE	Gets a variable for viewing.	16
GO	GO	An alternate form of the CONTINUE command.	9
GRAPH	G	Produces a graphics display.	8
HDUMP	H	Displays variable attributes and the symbol table (primarily for debugging).	20
INITIALIZE	I	Calls the user's initialization subroutine.	17
MACRO	M	Allows the user to create a SIMCON macro as a set of other SIMCON commands.	21
NAME	N	Defines a synonym for a variable name.	19
ONSTAT	O	Defines a list of variables on which to perform simple statistical monitoring during simulation.	16
PLOT	PL	Provides for automatic output to be generated for selected variables during simulation.	17
PRINT	P	Produces a table of the values of variables across time.	8
QUIT	Q	Terminates SIMCON execution.	10
READ	R	Causes commands to be read from a file.	25
RESET	RE	Restores a system state previously saved	20
SAVE	SA	Saves the current system state.	20
SET	S	Changes the value of a variable.	6
SIMULATE	SI	Sets simulation times and starts the simulation.	7
STATS	ST	Produces a statistical summary of variables over time.	16
TIME	T	Restores any model state.	9
UCOMAN	U	Calls a user defined command subroutine.	19
UNPLOT	UN	Reverses the action of the PLOT command.	17
VIEW	V	Produces a printer plot display.	8

Basic SIMCON Commands.

All that is necessary to manipulate a model are a means of iterating the model a specified number of times and of setting and displaying the values of variables. The following discussion describes the simplest and most basic set of SIMCON commands sufficient for the usual needs of simulation modelling.

Variable Naming Conventions. Many SIMCON commands reference the values of the FORTRAN variables of the model program. Both subscripted variables (arrays) and non-subscripted variables (scalars) may be referenced in commands. For example, the command "DISPLAY XVAL" would cause the value of the variable XVAL to be displayed at the terminal. Single elements of subscripted arrays are referenced in standard FORTRAN notation (see examples below). Subscript ranges may also be referenced. The follwing examples are all legitimate methods to reference variables:

the command:	references:
DISPLAY POP(10,2)	the single element in column 10 and row 2 of the array POP
DISPLAY MAX(ALL)	all the elements of the singly subscripted array MAX
DISPLAY SURF(5...10,ALL)	all the elements in columns 5 through 10 inclusive
DISPLAY MAX(10)	the tenth element only
DISPLAY MAX	the first element only

Any combination of the above referencing methods may be used.

The Basic SIMCON Commands.

SET variable=value

The named variable is set to the specified value. The value may be an integer or a real number or one of the special mnemonics "ON" and "OFF". Examples:

```
SET XVAL=1.7
SET MAX(10)=23
SET IFLAG=ON
```

Also,

```
SET FTYPE(1...5)=1 2 3 4 5
```

will set the first 5 elements of the array FTYPE.

```
SET POP(1...5)=10 20 30
```

assigns the first 5 elements of the array POP to the values 10 20 30 30 30. Note that the last value is repeated until all the elements are filled. A value may also be repeated using the "*" character. The following two commands have identical results:

```
SET MAX(1...10)=3*1 3*2 3*3 4
SET MAX(1...10)=1 1 1 2 2 2 3 3 3 4
```

In the SET command, the words "ON" and "OFF" are synonyms for the values 1 and 0, respectively. Also, the word "ALL" may be used to specify array elements. More examples:

```
SET POP(ALL)=0
SET MAT(ALL,4...6)=1
SET INUM(1,1,1...4)=1 2 3 4
SET VAL(1,1...3,1...3)=3*2.6 3*7.2 3*8.9
```

In the last example, the elements of the three dimensional array VAL are filled such that the left most subscript changes through its range most quickly (in column major order). Note that nine elements were referenced and assigned values.

DISPLAY variable(s)

The current values of the named variables are displayed at the terminal. For example, elements from three variables (two arrays and a scalar) are displayed by the following command:

```
DISPLAY MAT(1,ALL) XVAL MAX(1...10)
```

(See also examples on page 6.)

SIMULATE [FROM] i [TO] j

causes the model to be iterated and produces a simulation run. The symbol i represents the first year that is to be simulated -- or the year of the first iteration. The symbol j represents the last year of the simulation. The portions in brackets are optional and may be omitted.

A convenient alternate form of this command is:

SIMULATE [TO] j

which assumes that the first year is 1 and the last year is j. Some examples:

```
SIMULATE FROM 1966 TO 2000
SIMULATE 20
```

SIMCON "remembers" all the intermediate states of a simulation. The following output commands will allow the user to see the values of the variables as they have changed over the simulated time.

PRINT variable(s)

outputs a table of the values of the named variables over the simulated time. Example:

```
PRINT POPULATION MORTALITY CATCH CATCH-PER-EFFORT
```

VIEW variable(s) [MAX=value]

produces a printer plot of the named variables over the simulated time. The optional MAX specification scales the plotting axis. If a maximum is not specified, a maximum is computed for each variable. Example:

```
VIEW FPOP YIELD MAX=1E6
```

The expression "1E6" is a short-hand notation meaning 1 times 10 to the 6'th power (1 million). An example of VIEW command output is shown in figure 1.1, page 14.

GRAPH variable(s) [MAX=value]

produces a graphics display of the named variables against simulated time for any device supported by the CYBER COMPLOT graphics package. These include all Tektronix graphic terminals and plotters and the centrally located GERBER flatbed plotter, as well as others. (Consult the CYBER COMPLOT User's Manual available at the O.S.U. computer center for more information about the graphics devices supported by COMPLOT.) See also Obtaining Output Hardcopy from the CDC CYBER, page 15. Example:

```
GRAPH OBSERVED PREDICTED MAX=10000
```

GRAPH variable1 VS variable2 [PEN=value]

This special form of the GRAPH command permits plotting one variable against the other rather than against time. Variable1 is plotted on the Y axis against variable2 on the X axis. The plotting scale is automatically determined for each variable. The option "PEN=0" displays the data as unconnected points. "PEN=1" connects the points by lines in the order they were generated by the simulation. Example:

```
GRAPH SPAWNERS VS RECRUITS PEN=1
```

TIME i

resets the model to a previous state. The symbol i may represent any time or year previously simulated. Examples:

```
TIME 1
TIME 1957
```

TIME 0

Very often, the initial model state before simulation is desired so that a model user may change parameters to prepare for a new simulation run. This command always refers to the initial (or "zero'th") state irrespective of the units of simulated time (whether they be years or simple iteration counts). For example, the following sequence produces a simulation run of 20 years between 1960 and 1980 then returns to the initial state:

```
SIMULATE 1960 1980
TIME 0
```

CONTINUE [TO] j

is an alternate form of the SIMULATE command which is particularly useful in respect to the TIME command. CONTINUE simulates from the current state (the ending state of the last SIMULATE or CONTINUE command or a state set by TIME) to the year j. If j is not specified, CONTINUE considers the current state the "zero'th" state of a new simulation and iterates to the ending year specified on the last SIMULATE or CONTINUE command.

GO i

A command similar to CONTINUE, GO iterates the model from its current state exactly i iterations. Note that i is not the value of a year but the number of years to iterate beyond the current state. If i is omitted, 1 is assumed.

AT time command

This command will interrupt the next simulation at "time", execute "command", then resume the simulation. The AT command is useful to alter a model parameter with a SET command in the midst of a simulation. The word "ALL" may be used to specify that the command is to be executed between every time step of a simulation. Up to 20 different AT commands may be in effect at one time. Examples:

```
AT 5 SET F=.7
AT ALL SET GRO=1
```

Most SIMCON commands may be used in AT commands. However, the commands SIMULATE, CONTINUE, GO, TIME, or another AT command should not be used since they would have little meaning in this context. The PRINT, VIEW, GRAPH, and FILE commands should be avoided as well. (The PLOT command is much better suited for these purposes. The PLOT and FILE commands are discussed in Advanced SIMCON Commands and Features, page 16.) Also, a command such as "AT ALL DISPLAY ..." is very inefficient and should be avoided in preference to the PLOT or PRINT commands.

AT LIST

lists all AT commands currently in effect.

AT CLEAR

deletes all current AT command.

QUIT

terminates SIMCON execution. To restart the program, type the command "SIMCON" in response to the system prompt "/". The model can be reset to the state where it was left by using the TIME command.

System Parameters.

SIMCON uses several internal system parameters as pointers, flags defaults, etc. The following is a partial list of SIMCON system parameters which may be relevant to a user's particular application. All SIMCON parameter names begin with the sharp sign (#), and they may be displayed or modified by DISPLAY and SET commands in the manner of any other program variable.

#DEFMAX
is the default plotting scale maximum used for variables in the plot queue. (The plot queue is discussed in Advanced SIMCON Commands and Features, page 16.) The default value is 100.

#IBAUD
This parameter has meaning only for graphics output to remote graphics terminals and represents the transmission rate to the terminal. It used by the graphics display routines to calculate delay times. The default value is 1200 baud.

#IDUMP
is a switch that when ON (set to 1), SIMCON stores all intermediate model states on a scratch file. This makes all the states of a simulation run available to output and other commands such as PRINT, GRAPH, and TIME. The default is ON.

#IGPLT
contains the output method code for the variables in the plot queue:

1 output a printer line plot (default) (VIEW)
2 output a table of values (PRINT)
3 output a graphics display (GRAPH)
4 not used
5 output a table to the file UFILE (FILE)

#IPLDEV
contains the code defining the graphics output device:

1 Tektronix terminal (default)
2 not used
3 GERBER flatbed
4 both Tektronix terminal and the GERBER flatbed

#IYBEG
is the base year for the simulation. The SIMCON command SIMULATE sets this parameter. The default is 0.

#IYEAR
is the current year of the simulation. This parameter is incremented during the simulation.

#IYEND
is the last year of the simulation. The SIMCON commands SIMULATE, CONTINUE, and GO set this parameter.

#LOGUNT
contains the logical output unit number for PRINT and VIEW command output. Valid unit numbers are 6, the terminal (file OUTPUT); 45, file ULOG; 46, file UFILE. The default is unit 6.

#MODTEK
is the Tektronix model number. This parameter has meaning only for graphical output to a Tektronix terminal. The default is the Tektronix model 4010.

#NLINE
is the number of characters per line used for the printer line plot display. The default is 60 characters.

#NPLOT
is the number of variables currently in the plot queue.

#NYSKIP
is the iteration interval at which model states are saved if and only if the system switch #IDUMP is ON. The default value of 1 causes model states to be saved for every time step of the simulation, #NYSKIP=2 means every other time step, etc.

#PLTMAX(20)
contains the plotting axis scaling maximum for each variable in the plot queue.

Getting Started.

To use SIMCON to run a model and obtain meaningful output, the user must have some knowlege of the model in general and its most important variables in particular. This section is intended to assist users new to the CYBER begin the log-in process (initiate communication with CYBER) and run the demostration model provided as an example (figure 1.1, next page). To run other models, new users should find that their instructors or the model programmer have provided the specific instructions necessary.

Figure 1.1 illustrates the log-in sequence and a sample run on a terminal that is connected to the computer via a direct line (a terminal which does not need to communicate over a telephone). Most users will probably have access to such terminals so only this one example will be given.

The first step is to make sure that the terminal has the power turned on and the switch marked LOCAL/LINE set to LINE. On some terminals, this may be marked TERM, TERM READY, DATA, or AUTO ANSW. To begin the log-in sequence, push the return key. The system should respond with a minus sign (-). Type NOS and again push the return key. The system will then ask for four indentifiers. These identifiers, which include a user account number, password, charge number, and project name are required by the system in order to log-on successfully. The identifiers are assigned to every CYBER user and are designed to facilitate accounting and preserve user privacy. Enter the appropriate number in the blacked-out area when requested and end each line with the return key. When the computer is ready to receive your input, it will prompt you with the symbol "/". Now, you may repeat the demonstration shown in figure 1.1 or you may experiment to familiarize yourself with the SIMCON system. To end a terminal session, type BYE.

Figure 1.1. A log-in and sample SIMCON run. All user responses are underlined. DEMO is a procedure which links the demonstration model to SIMCON and is model specific. (The model and the DEMO procedure are shown in detail in Part II, page 34.) Generally, the instructor or model programmer will supply a similar procedure to run a particular model. New users are especially encouraged to repeat and experiment with this demonstration at their own terminals.

```
-NOS                                          Your account number.

 81/02/13. 12.23.41.
NOS 1.4
USER NUMBER: AAVI7N                           Type your password in the blacked out
PASSWORD                                      area provided by the computer.
********
TERMINAL:    107, TTY
RECOVER/ CHARGE: CHARGE                       Type the word "CHARGE".
  CHARGE NUMBER:
? **********                                  Your charge number goes here.
  PROJECT NUMBER:
? ********************                        And your project name.
 CUMULATIVE SRUS = ****    (* ******)
/GET,DEMO/UN=AAVI7N                           Type these commands exactly as shown.
/DEMO
REVERT. SIMCON DEMONSTRATION READY
/SIMCON                                       This command invokes the SIMCON system.
 SIMCON CDC VERSION 2.5.1

THIS IS A SIMPLE POPULATION GROWTH MODEL USING
THE CLASSICAL LOGISTIC FUNCTION,

    DN/DT = RN(K-N)/K

VARIABLES

POP      CURRENT NUMBER OF ANIMALS IN THE POPULATION (N)
R        CONSTANT OF PROPORTIONALITY (INTRINSIC RATE OF INCREASE)
CARRY    THE CARRYING CAPACITY (K)
YIELD    SURPLUS PRODUCTION, N(T+1) - N(T)

? DISPLAY POP R CARRY
 POP                  =    2.000000
 R                    =    .6000000
 CARRY                =    100.0000
? SIMULATE ! 15
? VIEW POP YIELD MAX=100
1
 VARIABLE  1 IS POP                 MAX =    100.
 VARIABLE  2 IS YIELD               MAX =    100.
      XXXXXXXXX+XXXXXXXXX+XXXXXXXXX+XXXXXXXXX+XXXXXXXXX+XXXXXXXXX+
    0I 21
    1I 21
    2I  2 1
    3I  2  1
    4I   2    1
    5I    2       1
    6I      2          1
    7I        2              1
    8I         2                    1
    9I         2                          1
   10I        2                                !
   11I      2                                    1
   12I    2                                        1
   13I  2                                           1
   14I 2                                             1
   15I 2                                             1
? QUIT
    1.289 CP SECONDS EXECUTION TIME.
/
```

Before typing, wait for this slash to appear. It signals that the computer is ready to accept your commands.

We need not stop here. We may change some model parameters or initial conditions with "SET" commands and continue simulations.

Obtaining Output Hardcopy from the CDC CYBER.

Output of printed tables or printer plots may be written to a file then routed to the central line printer. While in the SIMCON system, issue the the following commands:

```
? SET #LOGUNT=45          Define the output unit as the file ULOG,
? PRINT POP(1...10)       list the variables of interest, then
? QUIT                    exit SIMCON and issue the following CYBER
                          commands:

/TITLE,DUMMY./MYNAME      Label a dummy file with your name.
 TITLE COMPLETE.
/REWIND,ULOG              Rewind the output file, ULOG.
$REWIND,ULOG.
/COPY,ULOG,DUMMY          Copy the output to the dummy file.
 EOI ENCOUNTERED.
/ROUTE,DUMMY,DC=PR        Send the file to the line printer.
 ROUTE COMPLETE.          Your SIMCON output will be filed under your
```

name (the name given on the TITLE command) at the computer center's I/O counter. Just ask for it by name.

Sending graphical output to the GERBER plotter is done in much the same way as routing output to the printer:

```
? SET #IPLDEV=3           Select the GERBER plotter.
? GRAPH POP(5...10)
? QUIT

/TITLE,DUMMY,DC=GB,I      Provide the following label information:
? JOHN Q. PROGRAMMER      your name,
? 000000                  your charge number, and
? 1 PLOT 12 BY 18 IN.     number of plots desired and the dimensions
?                         End input with a return on an empty line
 TITLE COMPLETE.
/REWIND,TAPE10            SIMCON wrote the graph on the file TAPE10
$REWIND,TAPE10.           automatically for you.
/COPY,TAPE10,DUMMY
 EOI ENCOUNTERED.
/ROUTE,DUMMY,DC=GB        Route the file to the GERBER plotter.
 ROUTE COMPLETE.
```

Advanced SIMCON Commands and Features.

After some familiarity with SIMCON is gained, these commands and features should prove highly useful. Generally, the user should also be familiar with the CYBER file structure of local and permanent files.

STATS variable(s) [LO=value] [HI=value]

will produce a simple statistical summary of the named variables over the simulated time. The optional LO and HI values specify the lower and upper bounds of the simulated time for which the summary is to be made. The statistics computed are the maximum, minimum, mean, variance, standard deviation, and the standard error of the mean. Example:

```
STATS RAINFALL TEMPERATURE LO=1900 HI=1920
```

ONSTAT variable(s)

places the named variables into a queue. Statistics for these variables will be computed and output automatically after any subsequent simulation. If no variable list is given, the queue is cleared.

FILE variable(s) [FORMAT=(fmt)]

Similar to the PRINT command, FILE outputs a table of values without column headers onto the file UFILE. This command is useful where data generated by a simulation run are to be analyzed separately at a later time. The optional FORMAT specification allows the user to supply an alternate format other than the default (the default format is 10G12.5). The output format specification, "fmt" should be structured as in standard FORTRAN notation or the special mnemonic "U" may be used to specify unformatted binary output. The format conversions that may be used are E, F, G, and A. Integer and variable type conversions cannot be used. Consult the NOS FORTRAN IV Reference Manual for more details about format conversion types. The parentheses enclosing "fmt" are optional. Example:

```
FILE VAR1 VAR2 VAR3(1...3) FORMAT=(F6.2,E10.3,3F8.5)
```

GET variable(s)

This command reads the values of the named variables from the common block scratch file into a working area in central memory. The principle advantage of this command is that it allows for direct comparisons of variables from more than one simulation. Each GET command appends its variables to the working area up to a maximum of 10 variables. Then, a PRINT, VIEW, or GRAPH command which does not

have a variable list displays all the variables currently in the work area. Figure 1.2, next page, illustrates how variables from different simulation runs may be compared directly using the GET command.

Incidentally, all output commands except STATS and DISPLAY use this working area as a buffer space. A GRAPH command with a list of variables, for example, can be given, then the same display can be regenerated for another graphics device by repeating the GRAPH command without the variable list. This avoids costly accesses to the scratch file.

CLEAR

clears the working area used by output commands.

PLOT variable(s) [MAX=value]

The PLOT command places the named variables into a special plotting queue. During a simulation run, output to the terminal is automatically generated for all variables in the queue. The method of output, whether a printed table, printer line plot, or graphics display, can be selected by setting the system variable #IGPLT to the an appropriate code (see System Parameters, page 11). The optional MAX specification scales the plotting axis for printer plots and graphic displays. If a maximum is not specified, the system parameter #DEFMAX is used. Each PLOT command will add its variables to the queue up to a maximum of 20 variables. Example:

```
SET #IGPLT=1
PLOT POP(1...5) MAX=500
```

The PLOT command has the advantage of viewing simulation results in progress rather than accessing the common block scratch file after simulation is complete. For very large models, this can be a significant savings.

UNPLOT variable(s)

reverses the action of the PLOT command for the named variables.

INITIALIZE Command.

Frequently, it may be desirable to re-initialize a model with the user supplied subroutine UINIT. The INITIALIZE command has this function. If no subroutine UINIT has been supplied, INITIALIZE has no effect. The UINIT subroutine is discussed in Part II, section Optional User Routines, page 36.

Figure 1.2. Here, two simulation runs are made with one parameter changed and the results of an output variable are compared. This example was produced using the model PISCES (from User's manual for PISCES: a general fish population simulator and fisheries game program, Environment Canada Fisheries and Marine Service, Tech. Rept. 480 by A. V. Tyler).

```
? AT 10 SET BOATS=3000
? SIMULATE TO 30
? GET FPOP
? TIME 10
? SET BOATS=6000
? CONTINUE TO 30
? GET FPOP
? VIEW
1
 VARIABLE  1 IS FPOP                MAX =    .521E+08
 VARIABLE  2 IS FPOP                MAX =    .521E+08
      XXXXXXXXX+XXXXXXXXX+XXXXXXXXX+XXXXXXXXX+XXXXXXXXX+XXXXXXXXX+
   0I                                *
   1I                                 *
   2I                                  *
   3I                                   *
   4I                                    *
   5I                                     *
   6I                                      *
   7I                                       *
   8I                                        *
   9I                                        *
  10I                                       *
  11I                                    2   1
  12I                                  2   1
  13I                                2     1
  14I                              2       1
  15I                            2         1
  16I                          2           1
  17I                        2             1
  18I                      2               1
  19I                    2                 1
  20I                   2                  1
  21I                  2                   1
  22I                 2                    1
  23I                2                     1
  24I               2                      1
  25I              2                       1
  26I             2                        1
  27I            2                         1
  28I           2                          1
  29I          2                           1
  30I          2                           1
```

```
? AT 10 SET BOATS=3000
? SIMULATE TO 30
? GET FPOP
? TIME 10
? SET BOATS=6000
? CONTINUE TO 30
? GET FPOP
? VIEW
1
```

Generate a simulation run then get the variable FPOP (the total number of animals in the population). Then go back to an intermediate state and reset the model parameter BOATS, the quantity of fishing effort (measured in boat-days).
Get the variable FPOP from the second run, plot, and compare.

```
 VARIABLE  1 IS FPOP                MAX =    .521E+08
 VARIABLE  2 IS FPOP                MAX =    .521E+08
      XXXXXXXXX+XXXXXXXXX+XXXXXXXXX+XXXXXXXXX+XXXXXXXXX+XXXXXXXXX+
   0I                                *
   1I                                 *
   2I                                  *
   3I                                   *
   4I                                    *
   5I                                     *
   6I                                      *
   7I                                       *
   8I                                        *
   9I                                        *
  10I                                       *
  11I                                    2   1
  12I                                  2   1
  13I                                2     1
  14I                              2       1
  15I                            2         1
  16I                          2           1
  17I                        2             1
  18I                      2               1
  19I                    2                 1
  20I                   2                  1
  21I                  2                   1
  22I                 2                    1
  23I                2                     1
  24I               2                      1
  25I              2                       1
  26I             2                        1
  27I            2                         1
  28I           2                          1
  29I          2                           1
  30I          2                           1
```

Defining Variable Synonyms, the NAME command.

The NAME command enables the user to define a synonym of up to 20 characters for a model variable. The variable may then be referenced by either name. Any character is legal in a synonym except imbedded blanks and the special characters [,;:#$]. The form of the NAME command is,

NAME oldname newname

For example, the command,

NAME POP POPULATION-SIZE

will allow the model variable POP to be referenced by the name POPULATION-SIZE. Descriptive names such as this may be helpful and pleasing to the eye when they appear on DISPLAY or PRINT tables. Arrays, parts of arrays, or single array elements may also be given synonyms.

This command requires that sufficient space be reserved in SIMCON's symbol table. The default table size allows for 166 model variables and synonyms which is probably more than adequate for most SIMCON applications. Since this is a programming consideration, this subject is discussed further in Part II, Optional User Routines, page 36.

User Defined Commands.

These commands perform user defined functions and are used in the same manner as the regular SIMCON commands. SIMCON provides for up to 7 user defined functions. The command functions are supplied by FORTRAN subroutines which are compiled with the model and linked with SIMCON at execution time. (See Part II, Optional User Routines, page 36.) The commands are: UCOMAN, UCMD2, UCMD3, UCMD4, UCMD5, UCMD6 and UCMD7.

Renaming SIMCON Commands.

SIMCON commands may also be given new names. These may be of up to 20 characters but must not include imbedded blanks or any of the special characters [,;:#$]. This feature should be especially useful with the user commands described above. Macros should not be given names that are defined as SIMCON commands or visa versa for when a macro call by name is issued, SIMCON searches its command list first. (Macros are discussed on page 21.) Using the CALL command avoids this difficulty since SIMCON presumes the name is a macro. The command has the form,

COMMAND oldname newname

HDUMP [n]

This command is primarily a debugging aid for model programs. It produces a catalog table of the variables in the model common block and the SIMCON utility common block. Included is information on the common block displacement of each variable, the variable length, maximum subscript sizes, and the variable type. The optional parameter n specifies the block to catalog: 0, both the model (user) and SIMCON common; 1, the model common; 2, the SIMCON common. The default is 1.

SAVE

stores the current SIMCON system state on the file SYDUMP. The SIMCON system state includes all system parameters (some of which are listed on page 11) and the symbol table (mentioned in conjunction with the NAME command, page 19 and discussed in more detail in Part II, Optional User Routines, page 36).

During SIMCON initialization, considerable execution time is devoted to the building of the symbol table and its associated parameters from the programmer supplied information contained in the file COMMON. (See Part II, Programming Requirements, page 32 for a discussion of the files associated with building the symbol table.) After this table is constructed, the SAVE command stores the table on a file so that during later SIMCON runs with the same model, the cost of construction can be avoided. This savings can be very substantial if the number of model variables is 100 or more.

RESET

restores the system state saved by the save command.

SIMCON Macros.

Definitions. A SIMCON macro is a user defined collection of SIMCON commands assigned an identifying name. The commands forming a macro are executed as a unit by simply entering the macro name as if it were a regular SIMCON command. Macros can provide a convenient means to quickly and accurately initialize a model for different types of simulation runs. They may also be used to execute often repeated command sequences or, in the case of a model used for instructional purposes, may contain explanatory comments to the user (see SIMCON Comments, page 26). A macro of this type might be named "HELP".

A macro is defined by the command,

MACRO name [P1 P2 ... P10]

where,

name is a character string of up to 20 characters identifying the macro. Letters, numbers, and symbols may be used in a name except blanks, commas, colons, or semi-colons. The characters # and $ may be used but must not be the first character of the name.

Pi is one of up to 10 parameters that may take on one of the following forms:

K a key-word name of up to 10 characters. A key word is a name used inside the text of a macro for which we may wish to substitute the name of a variable, symbol, or value when the macro is executed. The characters comprising a key-word name may be letters, numbers, or symbols except those mentioned above.

K=D a key-word K that is to be assigned the value or string D by default. Default assignment in this context means that the value or string D is substituted for the key-word name K wherever K appears inside the macro text if the user makes no other explicit assignment to K when the macro is executed. (Explicit key-word assignment is described in more detail below.)

K= a key-word assigned to "null" by default i.e. the key-word is to be deleted from the macro text before execution if no other explicit assignment is made to K.

A macro is executed by the command,

```
CALL  name [P1 P2 ... P10]
```

where,

CALL — is an optional item of the command,

name — is the name assigned to a macro, and

Pi — is an optional parameter of one of the following forms:

- V — a value or character string to be substituted for the key-word Ki in the order it appears on the macro definition statement,
- K=V — where V is a value or string to be substituted for the key-word K,
- K= — key-word K is to be deleted from the macro text.

When a macro is called by a command of the form,

```
CALL  name V1 V2 ... V10
```

V1 is substituted for key-word K1 on the macro definition statement, V2 for K2, etc. The parameters on this command are are order dependent. Example:

```
RATES .5 .1 .125 .05 .225
```

If a macro is called by,

```
CALL  name K1=V1 K2=V2 ... K10=V10
```

where Vi is either absent (null), a character string, or a value and Ki is the name of any key-word on the macro definition statement, V1 is substituted for K1 regardless of the order in which the key-words Ki appear. The parameters of this command are order independent. Example:

```
PLAN TYPE=1 VAR=FLEET
```

In general, order independent and order dependent forms ought not be mixed. However, order independent parameters may follow order dependent parameters on the CALL command but order dependent parameters should never follow order independent parameters.

Macro properties and limitations. Macros are easily defined from text files (see Command Files, page 25) or created interactively and are stored on a random access file which may be permanently maintained on the user's account. Any previously defined macro contained on this file, called the SIMCON macro library, is automatically available for direct use during any SIMCON run. A macro library created during one SIMCON run may be saved and used in any other SIMCON run at any time without special attention; the user simply gets the library as a local file either before SIMCON is executed (see SIMCON Loading and Execution, page 40) or during SIMCON execution with the SIMCON $GET command (see SIMCON System Commands, page 27). A new library may also be used by issuing another $GET command. Some simple macro examples are shown on figure 1.3, next page.

Macros may be nested (macros may reference other macros). While there is no arbitrary limit to the degree of macro nesting, each macro "call" increases the demand on the central memory allotted to macro execution (the actual amount depends on exactly where a macro call occurs inside another macro). If a memory overrun occurs, SIMCON issues an informative message and stops macro execution.

A macro library may contain up to 64 individual macros and each macro may consist of from approximately 60 to 100 SIMCON commands depending on the length of each command. By "chaining" macros (the last command of a macro being a call to another macro), the effective length of a macro can be increased. In this case, there would be almost no chance of a memory overrun since for each command executed, that memory space is freed.

There is no provision in SIMCON to edit a macro, however, a macro may be redefined by simply creating a new macro with the same name. Unfortunately, the macro library cannot be edited on any CYBER text editor (EDIT or XEDIT). Users may find it more convenient to define or redefine macros in command files which are easily edited. (See Command Files, page 25.)

Any SIMCON command may be contained in a macro except another macro definition statement.

Figure 1.3. Some macro examples.

```
? --DEFINING A SIMPLE MACRO
? MACRO RUN3
 ENTER MACRO TEXT
? GO 1
? DISPLAY PROGRAM COST YEARS TOTAL
? DISPLAY YIELD EFFORT CPUE SUM
? DISPLAY RECRUITS
? END
 ENTER COMMAND
?
```

The following commands form the macro RUN3.

END (or from an interactive terminal, a return on an empty line) terminates the macro definition.

```
? --CALLING A MACRO BY NAME
? RUN3
 PROGRAM          =    3.000000
 COST             =    .2400000E-01
 YEARS            =    3.000000
 TOTAL            =    .7200000E-01
 YIELD            =    475.8390
 EFFORT           =    3000.000
 CPUE             =    158.6130
 SUM              =    1996.490
 RECRUITS         =    8700827.
?
```

SIMCON looks up the macro RUN3 and executes it.

```
? --A MACRO WITH PARAMETERS
? MACRO SETVAL X Y
 ENTER MACRO TEXT
? SET XVAL=X
? SET YVAL=Y
? DISPLAY XVAL YVAL
? END
 ENTER COMMAND
?
```

Define the macro SETVAL with parameter key-words X and Y.

Execute the macro SETVAL.

```
? SETVAL 1.5 100
 XVAL             =    1.500000
 YVAL             =    100.0000
?
```

Command Files.

A command file is a user defined set of SIMCON commands stored as text on a file. A command file may be conveniently used to initialize a model before the first simulation run. In this case, the file need only be provided as a local file with the name BATCH and SIMCON will automatically read and execute the commands during the initialization process. (See SIMCON Loading and Execution, page 40.) Command files differ from macros since it is not possible to pass parameters or create them interactively while running SIMCON and since each command file must be maintained as a separate permanent file. They are, however, ideal places to perform model initializations, define macros, or entire macro libraries. (Macros and macro libraries are discussed on page 21.) Also, they may be of virtually unlimited length.

Command files are explicitly executed by the READ command which has the general form,

READ pfn [/UN=user no]

where "pfn" is the name of an indirect access permanent file. The portion in brackets is optional and can be used to specify that the file is on an alternate user account (the file must be public). If "pfn" is not specified, the READ command will cause SIMCON to rewind then reread the last command file processed (permitting reinitializing via the same BATCH file). The action of this command is to first perform the equivalent of a system "GET" control statement (only when a "pfn" file name is given) which equates "pfn" with the file name BATCH. The effect is, "GET,BATCH=pfn". The SIMCON command "READ BATCH" is permitted and results in the equivalent of the system control statement "GET,BATCH=BATCH". (Note that this is not equivalent to the READ command without a "pfn" file specified.)

Caution: When a "pfn" file name is provided, the READ command will return or destroy the local BATCH file. If there is no permanent copy of the local BATCH file and the user wishes to preserve it, the user must use the SIMCON "$SAVE" command before issuing the READ command. (See SIMCON System Commands, page 27.)

A command file may contain any SIMCON command including calls to macros except that it may not contain, or in any way encompass, another READ command (as for an example, a command file calling a macro which contains a READ command). A READ command encountered while a command file is being processed may produce unexpected results.

SIMCON Comments.

Part of the usefulness of SIMCON macros and command files is the ability to provide explanatory messages to SIMCON users. This is particularly true when a simulation model is being used in a classroom or for demonstration purposes. There are two types of comments. One type is always echoed at the terminal when encountered in command files or macros. The other type is not echoed. The echoed comment type is a line of text prefixed by two periods (..) in columns 1 and 2. The non-echoed type is preceeded by two dashes (--) or a single asterisk (*). (This latter character is provided for compatability with certain CYBER system utility functions dealing with file management such as CATALOG and LIBEDIT.) The usefulness of the non-echoed comment type generally is limited to the annotation of command files. Example:

```
/SIMCON
 SIMCON CDC VERSION 2.5.1 EXTENDED
? MACRO SPEAKTOME
 ENTER MACRO TEXT
? --THIS STATEMENT WILL NOT BE SEEN OR HEARD.
? ..HI THERE! HOW ARE YOU?
? END
 ENTER COMMAND
? SPEAKTOME
 HI THERE! HOW ARE YOU?
?
```

SIMCON System Commands.

With these commands, the user may perform limited file manipulations with most SIMCON files. SIMCON system commands are prefixed by the dollar sign ($) and are similar in syntax to the system control statements which perform the same functions. The SIMCON system commands may manipulate all the following SIMCON files:

BATCH The latest command file referenced. This file is the last file referenced by a READ command or the original file provided during SIMCON initialization.

DUMP A SIMCON utility random access file containing the common block images from the last simulation run. This file is used as a scratch area to store successive model states.

MACLIB The current macro library file.

SYDUMP A scratch file used to store the symbol table and the state of the SIMCON system.

TAPE10 The output file for graphics information for the central plotters.

UDATA An optional user supplied data file that may be used as input to the user initialization subroutine, UINIT.

UFILE The output file for the FILE command.

ULOG An optional output file for the PRINT and VIEW commands.

The following SIMCON system commands are available. The "lfn" parameter must be the name of one of the above listed files, "pfn" may be any permanent file. All portions in brackets are optional.

$SAVE,lfn =pfn [/UN=userno]

This command enables the user to save as an indirect access permanent file the local file "lfn" under the name "pfn". If the "pfn" option is not specified, "pfn" defaults to the name supplied for "lfn". The "/UN=userno" option allows the file to be saved under an alternate account. (The alternate account must contain a public indirect access file of the name "pfn" with write permission.) The $SAVE command performs the equivalent function of the system control statement "REPLACE,lfn=pfn/UN=userno".

$GET,lfn[=pfn][/UN=userno]

This command "GET's" the indirect access permanent file "pfn" on the user's account or an alternate account specified by the "/UN=userno" option and equates it with the local file name "lfn". If "pfn" is not specified, "pfn" defaults to the name "lfn".

Caution: The $GET command will destroy the local copy of the "lfn" file. If the user wishes to preserve the original file, use the $SAVE command prior to the $GET command.

$REWIND,lfn

rewinds the named file. $REWIND has no effect on the macro library or the dump file.

$RETURN,lfn

returns the named file. The $RETURN command destroys the current local copy of the "lfn" file.

Miscellaneous Features.

Continuing commands on more than one line.

Long variable names and lists somtimes make desirable the ability to continue commands on more than one line. This can be particularly true on a narrow carriage machine such as the older model Teletype terminals. Any command (with the exception of comment lines) may be continued on the next line by simply terminating the line with a comma. There is no arbitrary limit to the number of continuation lines for a single command, however, SIMCON will issue an error message if the command contains more than 160 characters.

Multiple commands on one line.

More than one command may be entered on one line by separating the commands with semi-colons (;). Users may find that with the slow CYBER response time, multiple commands should be a speedy way to execute a sequence of short commands. Macros may also be defined on one line but with some restrictions. The macro must be completely specified in the multiple command sequence (continuation lines may be used). All the commands of the multiple command sequence following the MACRO command will become the definition of the macro. The END command is implied by the end of the command sequence and should not be explicitly included. Example:

```
MACRO STEP;GO 1;DISPLAY WEIGHT LENGTH NET-GROWTH
```

Attention Interrupts.

Any SIMCON function may be interrupted by pushing the BREAK key (on some terminals, this key is marked INTRPT). This feature is useful to prevent unwelcome or erroneous output to the terminal or halt a user's unintended mistake.

Part II. The FORTRAN Programmer's Guide to SIMCON

or

How to Create a FORTRAN Model for SIMCON

System Architecture.

The SIMCON structure has at least three fundamental properties. The first is that SIMCON is model independent. The model is simply constructed of the rules of change of the model system from one time step to the next. SIMCON provides for all input, output, and intervention needs. A call by SIMCON to the model simulates one time step.

The second property is the user's ability to reference all the model variables during program execution by FORTRAN name. The method is to locate all the variables the user may wish to view or change in the blank common block. The user then supplies SIMCON with a separate file containing all the declaration statements associated with the variables in the blank common block. From these statements, SIMCON calculates and stores variable attributes such as the declaration type, subscript ranges, and common block displacements for each variable.

The third property is that SIMCON performs three functions during model iteration:

- Variables that the user has specified are displayed as the simulation proceeds.
- Each state of the model is saved for later reference. Saving the state of the model means, in this case, that the contents of the model's blank common block is stored on a random access file.
- And, if a SIMCON command has been defined to intervene at a particular time step, the command is executed at that point.

Programming Requirements.

A user need only provide two files. One file contains the object code for the model subroutine(s) and is named BMODL. The second file contains FORTRAN declaration statements associated with the model's blank common block and is named COMMON.

BMODL must contain one subroutine named UMODEL. This subroutine functions as the main model routine. At the user's option, UMODEL may reference any number of satellite subroutines. The user may also reference the utility random number generators provided in the SIMCON library (see page 43). All user referenced subroutines except those contained in the SIMCON library must be compiled with the main routine UMODEL and contained within the object file, BMODL. The names of any user supplied subroutines other than subroutine UMODEL are arbitrary. Figure 2.1, page 34 demonstrates the correct form of a model and associated files prepared for a SIMCON run. SIMCON's calling sequence to subroutine UMODEL is,

```
CALL UMODEL (ITIME)
```

where ITIME is the current simulation time step.

Other considerations concern named common blocks and a blank common block more than 1000 words of memory in length. See the CCOM routine in the section Optional User Routines, page 36 for details.

The file BMODL may contain optional routines which are referenced by SIMCON rather than the user's model. These may include a one time initialization routine, one or more user command routines, or the specialized block length definition routines. These routines are discussed also in the section Optional User Routines.

The COMMON file contains only FORTRAN declaration statements associated with the model's blank common block. The declaration statements in the COMMON file must be identical to the blank common declaration statements as they appear in the user's model and must conform to the following rules:

1. The COMMON file must contain all INTEGER, REAL, LOGICAL, DIMENSION, and COMMON statements that apply to any variable in the blank common block.

2. The COMMON file may contain only INTEGER, REAL, LOGICAL, DIMENSION, COMMON, and comment statements.

3. A COMMON statement must be the last statement of the file. The dimensions of a variable may be given in any declaration statement, however.

4. Named common blocks are not permitted.

5. No more than nine continuation cards per statement are permitted.

6. Arrays of up to four dimensions are permitted although CYBER FORTRAN permits only three.

The COMMON file is not a program routine but a data input file to SIMCON used to construct the symbol table. For models with many variables, this is a costly operation. But, as long as the COMMON file and the model's blank common block have not been altered, it is not neccessary for SIMCON to read the COMMON file more than once. The SAVE command can be used to store the constructed symbol table on the scratch file SYDUMP. (See the SAVE and RESET commands, Part I, page 20.) Thereafter, during subsequent SIMCON runs with the same model, the file SYDUMP can be used in lieu of the file COMMON to initialize the system and symbol table. (See page 40, SIMCON Loading and Execution.)

Figure 2.1. The simple logistic growth model demonstrated earlier in Part I, figure 1.1, page 14. The procedure DEMO performs the neccessary steps of preparation for SIMCON execution. Such a procedure is

The model ----

```
      SUBROUTINE UMODEL (IT)
C
C A SIMPLE LOGISTIC GROWTH MODEL
C
C VARIABLE DEFINITIONS
C   POP     CURRENT POPULATION SIZE
C   R       INTRINSIC RATE OF INCREASE
C   CARRY   CARRYING CAPACITY OF THE SYSTEM
C   YIELD   SURPLUS PRODUCTION (AMOUNT OF INCREASE FROM
C           THE LAST TIME STEP)
C
      COMMON POP,R,CARRY,YIELD
C
C THE CLASSICAL LOGISTIC EQUATION ...
      YIELD = R*POP*((CARRY-POP)/CARRY)
      POP = POP + YIELD
      RETURN
      END
C
      SUBROUTINE UINIT
C OPTIONAL USER INITIALIZATION ROUTINE
      COMMON POP,R,CARRY,YIELD
      POP = 2.
      R = .6
      CARRY = 100.
      RETURN
      END
```

optional. Each step may be given as individual commands from an interactive terminal. See page 40 for a general description of the steps that may be placed in a procedure.

The COMMON file (required) ---

```
C  THE USER'S COMMON BLOCK FILE.
C  THIS FILE IS NOT A SUB-PROGRAM BUT DATA INPUT TO SIMCON.
C  IT MUST CONTAIN ALL INTEGER, REAL, LOGICAL, DIMENSION,
C  AND COMMON STATEMENTS THAT PERTAIN TO ANY VARIABLE IN THE
C  BLANK COMMON BLOCK. (SIMCON IS SMART ENOUGH TO IGNORE THESE
C  COMMENTS.)
      COMMON POP,R,CARRY,YIELD
```

The BATCH file (optional) ---

```
--DEMOPRO BATCH FILE
--A FILE SIMILAR TO THIS FILE MAY BE EMPLOYED FOR MODEL
--INITIALIZATION INSTEAD OF OR IN CONJUNCTION WITH SUBROUTINE
--UINIT. THIS FILE CONTAINS ONLY COMMENTS, HOWEVER, ANY
--SIMCON COMMAND COULD BE INCLUDED.
..
..THIS IS A SIMPLE POPULATION GROWTH MODEL USING
..THE CLASSICAL LOGISTIC FUNCTION,
..
..    DN/DT = RN(K-N)/K
..
..VARIABLES
..
..POP      CURRENT NUMBER OF ANIMALS IN THE POPULATION (N)
..R        CONSTANT OF PROPORTIONALITY (INTRINSIC RATE OF INCREASE)
..CARRY    THE CARRYING CAPACITY (K)
..YIELD    SURPLUS PRODUCTION, N(T+1) - N(T)
..
```

The DEMO procedure (optional) ---

```
.PROC,DEMO.
GET,DEMOPRO,BATCH=DEMOBAT,COMMON=DEMOCOM/UN=AAVI7M.
RETURN,BMODL.
FTN,I=DEMOPRO,B=BMODL,L=0.
GET,SIMC/UN=AAVI7M.
SIMC.
REVERT. SIMCON DEMONSTRATION READY
```

Optional User Routines.

One or more optional subroutines may be used. An optional subroutine must be compiled with the model and contained in the object file, BMODL.

Model initialization, subroutine UINIT.

The user subroutine UINIT may be provided to initialize the model before simulation begins. The UINIT subroutine is generally intended to input initialization data from the file UDATA (assigned by SIMCON to unit 44) although it may be used to perform any specialized initialization function. There are no calling parameters to subroutine UINIT.

Model initialization may also be performed with a command file named BATCH. (Refer to Part I, Advanced SIMCON Commands, Command Files, page 25 and SIMCON Loading and Execution, page 40.) This BATCH file when provided before SIMCON initialization, will be automatically read and executed. Both the subroutine UINIT and a BATCH file may be used together. SIMCON's initialization sequence is first to zero the user's blank common block, call the UINIT subroutine if one is present, read and execute the BATCH file if one is present, then place the user in SIMCON command mode.

User command routines.

The user may create command routines to perform specialized functions. SIMCON provides seven general purpose user commands: UCOMAN, UCMD2, UCMD3, UCMD4, UCMD5, UCMD6, and UCMD7. Each of these commands calls a user supplied subroutine of the same name (a user supplied command subroutine must have one of these names). A user command subroutine should have the following structure:

```
SUBROUTINE UCOMAN (IBUFF,LEN)
DIMENSION IBUFF(LEN)
  .
  .
  .
RETURN
END
```

The integer array IBUFF contains a character image of an optional user defined parameter list that may be included on the command. Each array element contains one character, left justified, blank filled (the character occupies the left most position of the word followed by nine blanks, i.e. A1 format). LEN is the number of characters in IBUFF.

A SIMCON user command has the general form,

```
UCOMAN [user defined parameter list]
```

which results in a single call to the user subroutine UCOMAN and a character image of the parameter list being placed in the array IBUFF.

Clearly, this method of parameter passing can become quite complicated, however, the parameter list can be used as a "switching" device without too much difficulty. For instance, a user might write a command subroutine that will perform one function from a choice of functions, say, functions 1, 2, etc. If the user gives the command,

```
UCOMAN 1
```

the user's UCOMAN subroutine will identify the function type (in this case, using the fact that IBUFF(1) contains the single character "1") and branch to the appropriate section of code.

Common block redefinition routines.

Before program execution, the user may redefine the default sizes of two common blocks, the blank common and a SIMCON named common which contains SIMCON's all important symbol table.

The blank common. SIMCON predefines the blank common block size at 1000 words of memory. It is important to remember that the blank common as defined in the user's model does not define the size of the block during program loading. Therefore, if the user's blank common block is longer than 1000 words, the user must increase the blank common block size by supplying the subroutine CCOM within the BMODL object file. Do not confuse the CCOM subroutine with the COMMON file described in Programming Requirements, page 32; CCOM performs an entirely separate function. The CCOM routine should have the following form:

```
SUBROUTINE CCOM
COMMON DUMMY(n)
RETURN
END
```

where n is the new common block length in memory words. The COMMON statement can be the same common statement used in the model, i.e. all the COMMON statements pertaining to the blank common block (the other declaration statements are unnecessary). Otherwise, any dummy variable can be dimensioned to a suitable length. CCOM should contain no executable statements other than a RETURN statement.

If the user employs any named common blocks in the model, the definition of this block should also be placed in the CCOM routine.

CCOM is force loaded into the primary overlay of the SIMCON program to ensure that the contents of the common blocks defined in CCOM are preserved during all phases of SIMCON execution. If the user provides no CCOM subroutine, the default CCOM with 1000 words is loaded from the SIMCON library.

The symbol table. The symbol table contains all the attributes of the model and system variables necessary for the user to reference the variables by name. The default size of the symbol table allows for the 34 SIMCON system variables (some of which were described in Part I, System Parameters, page 11) and a maximum of 166 model variables. Each variable requires 11 words of memory for a total default block length of 2200 words. If the model has more than 166 variables in the blank common block or the user intends to define several variable synonyms during SIMCON execution (each synonym requires a table entry, see Part I, the NAME command, page 19), the length of the symbol table must be increased. This is accomplished by including within the BMODL object file a specialized block data subprogram named HCOM. HCOM must have exactly the form:

```
BLOCK DATA HCOM
COMMON /KCC/ MAX,ENTSIZ,NENTS,IKDAT(n)
DATA MAX/n/
END
```

where n is the length of the symbol table in words of memory. For example, to reserve space for 250 model variables, n should be computed as 11 X 250 + 11 X 34 = 3124 words. Recall that each variable entry requires 11 words and that SIMCON must have space for its 34 system variables. The value of n in the COMMON statement and in the DATA statement must agree.

SIMCON Files and Unit Assignments.

BATCH	unit 1	The last-accessed command file.
COMMON	unit 2	The definition file for the blank common block.
INPUT	unit 5	Input from an interactive terminal.
OUTPUT	unit 6	Output to an interactive terminal.
TAPE10	unit 10	The output file for graphics information to the central plotters.
DUMP	unit 13	The random access file used for storing model states.
MACLIB	unit 18	The random access file for the macro library.
SUPCOM	unit 19	The SIMCON utility common block definition file.
SYDUMP	unit 12	A scratch file used to store the SIMCON system state.
UDATA	unit 44	Initialization-data input file referenced by user supplied subroutines.
ULOG	unit 45	An alternate output file for PRINT and VIEW commands.
UFILE	unit 46	The output file for the FILE command.

SIMCON Loading and Execution.

The following are general descriptions of the necessary and optional steps to prepare a model for a simulation run. The steps are defined in the order they should be issued at the terminal.

FTN,I=MYMODEL,B=BMODL,L=0	Compile the model and optional user routines to create the object file, BMODL. If a program listing is desired, use L=filename. If there are errors, correct these before continuing.
GET,BATCH=filename	This will be an optional user created command file to initialize the model. If you are not using the BATCH file to initialize the model, skip this step.
GET,COMMON=filename	Create a local copy of the blank common block definition file. Either this file or the file SYDUMP is required.
GET,SYDUMP=filename	If the COMMON file is not to be used to initialize the symbol table, create a local copy of the file SYDUMP. (See Programming Requirements, page 32.)
GET,SIMC/UN=AAVI7M	Create a local copy of the procedure file to load SIMCON.
SIMC	Load SIMCON. If there are no errors, proceed.
GET,MACLIB=filename	This optional step provides a previously created macro library. (See SIMCON Macros, page 21.)
GET,DUMP=filename	This optional step permits you to examine the results of a simulation which had been previously saved.
SIMCON	Start SIMCON execution. When SIMCON prompts, enter a command.

Once the model is running correctly, a procedure file can be created that performs all the steps above. Such a procedure is shown in figure 2.1, page 34 and demonstrated in Part I, figure 1.1, page 14.

Program Debugging.

Before SIMCON terminates due to a FORTRAN error, SIMCON closes all files thus preserving their integrity. The most important file in this respect is the DUMP random access file as it is through this file that the model can be recovered to the point at which the error occurred. (The DUMP file is created only if the system parameter #IDUMP is ON. Refer to Part I, System Parameters, page 11.) SIMCON also generates a catalog of its system variables on the file ULOG which may be helpful.

As a further aid in programming debugging, a model may be compiled with the CYBER FORTRAN Post Mortem Dump facility which provides additional detailed information when FORTRAN errors occur. Consult the CYBER NOS FORTRAN Extended Version 4 Reference Manual for instructions on the use of the Post Mortem Dump facility.

To restart after a FORTRAN error, begin SIMCON executing again by repeating the command "SIMCON" to the operating system. Now, at SIMCON's prompting, enter the command "TIME n" where n is the iteration year at which the error occurred (this number is provided somewhere within the error messages). Now, the model variables associated with the error can be examined, and presumably, the error located. If correction of the model program is indicated, quit SIMCON, correct and recompile the model source code, then repeat SIMCON loading by executing the SIMC procedure.

Conditional Attention Interrupts.

Conditional attention interrupts can be used as a program debugging aid or as an error trapping device. During model iteration, the model can generate an attention interrupt by calling the SIMCON subroutine ATTN. SIMCON detects the interrupt condition, immediately stops iteration, and returns the user to SIMCON command mode. A call to subroutine ATTN does not in itself cause an exit from the model program; you must provide the means of exit via a return statement within the UMODEL subroutine. A suggested method for employing the conditional interrupt is as follows:

```
      SUBROUTINE UMODEL (ITIME)
        .
        .
        .
      IF (X .LT. 0.0) GOTO 99      Presume X less than zero is an error.
        .
        .
        .
   99 PRINT*, "BAD BOO BOO"        Here is the error exit code of
      CALL ATTN                    UMODEL.
      RETURN
```

The SIMCON Utility Library.

The following are random number functions available from the SIMCON utility library (adapted from The GASP IV Simulation Language, Alan F. Pritsker). Users may also use the random number generators supplied by the FORTRAN library.

Function DRAND (ISTRM,ISED)

generates a uniform random variate in the range (0,1). All other random number functions call this routine.

ISTRM — a number from 1 to 6 specifying one of 6 random number sequences.

ISED — dummy argument.

Subroutine DRSET (ISTRM,ISED)

initializes the generator seed for DRAND. If DRSET is not called, a default seed is provided for each sequence.

ISTRM — a number from 1 to 6 specifying one of 6 random number sequences.

ISED — used as the generator seed for the random sequence specified by ISTRM.

Function UNFRM (RMIN,RMAX,ISTRM)

generates a uniform random variate in the range (RMIN,RMAX).

RMIN — lower bound on the range

RMAX — upper bound on the range

ISTRM — see DRAND

Function TRIAG (RMODE,RMIN,RMAX,ISTRM)

generates a random variate from a triangular distribution in the range (RMIN,RMAX).

RMODE — distribution mode (the peak of the triangle) necessarily a number within (RMIN,RMAX).

ISTRM — see DRAND

Function RNORM (RMEAN,RMIN,RMAX,SD,ISTRM)

generates a random variate from the normal distribution in the range (RMIN,RMAX).

RMEAN distribution mean

SD standard deviation

ISTRM see DRAND

Function RLOGN (RMEAN,RMIN,RMAX,SD,ISTRM)

generates a random variate from the log normal distribution in the range (EXP(RMIN),EXP(RMAX))

RMEAN mean of the normal distribution

RMIN lower bound of the range of the normal distribution

RMAX upper bound of the range of the normal distribution

SD standard deviation of the normal distribution

ISTRM see DRAND

Function ERLANG (BETA,IALPHA,RMIN,RMAX,ISTRM)

generates a random deviate from the Erlang distribution (a Gamma distribution with a positive integer parameter IALPHA) in the range (RMIN,RMAX).

BETA,IALPHA are the parameters of the distribution. The product IALPHA*BETA is the expectation, IALPHA*BETA**2 is the variance.

ISTRM see DRAND

Function EXPON (RMEAN,RMIN,RMAX,ISRTM)

generates a random variate from the exponential distribution in the range (RMIN,RMAX).

RMEAN the distribution mean

ISTRM see DRAND

Function NPSSN (RMEAN,RMIN,RMAX,ISRTM)

generates a random deviate from the Poisson distribution in the range (RMIN,RMAX).

RMEAN the distribution mean

ISTRM see DRAND

Function GAMA (BETA,ALPHA,RMIN,RMAX,ISTRM)

generates a random variate from the Gamma distribution in the range (RMIN,RMAX).

BETA,ALPHA are the distribution parameters, the product ALPHA*BETA is the expectation, ALPHA*BETA**2 is the variance.

ISTRM see DRAND

Function BETA (THETA,PHI,RMIN,RMAX,ISTRM)

generates a random variate from the Beta distribution in the range (RMIN,RAMX).

THETA,PHI where the expectation is THETA/(THETA+PHI) and the variance is THETA*PHI/ (THETA+PHI)**2*(THETA+PHI+1)

ISTRM see DRAND

Part III. Error Messages

SIMCON ERROR MESSAGES

This is an attempt to document all of the messages that SIMCON version 2.7.1 can generate when it is unhappy. Most often the result of these messages is that the command just issued by the user has caused a problem, the message is issued and the command is ignored. A couple of errors are fatal, meaning execution ceases after they are issued. Other errors are slightly worse in that they require the repair of something and the re-running of a simulation - an unpleasant thought if there is a lot of CPU time involved.

The messages are arranged alphabetically as much as possible with messages not fitting that scheme located at the end of the list. The narrative about each message contains a description about how it came to be and some suggestions on how possibly to remedy the situation. Some of these remedies (such as modification of a SIMCON subroutine) may be considered drastic for the casual user. These modifications are mentioned as possible solutions. However, they should only be undertaken with extreme caution and in close consultation with the SIMCON Implementer's Guide. For users with the determination to find out more about these errors, the subroutine in which the offensive WRITE statement is located is also given. There are a few occasions where the code was found to be in conflict with the Users Guide and these points are duly noted.

Hopefully, this addition to the growing body of SIMCON documentation will make use of the package a little less forbidding.

"AT" LIST FULL - COMMAND NOT PROCESSED

It is only possible to have 20 AT commands in effect at any one time. If this number is exceeded, this message is issued. By the same token, there is only a finite amount of space (ca 500 packed characters) allocated for storage of AT commands for use in subroutine ATS. If the AT commands the user has defined are very complex, this space resource may be exhausted before the 20 command limit is reached. User action: If either of the two constraints mentioned above are too confining, SIMCON may be modified and recompiled. The operative lines are statement number 1 in subroutine ATS for the 20 command limit and common block /ATCMDS/ which is initialized in the BLOCK DATA subroutine for the size constraint. A less drastic way around the size problem may be to define a macro and use it in conjunction with an AT command.

BAD SYNTAX NEAR _____

The user has entered the SET command but there is either no equal sign on the line or it appears at the end of the line. In either case, there is little information content in the line and subroutine LOOPER complains. Re-enter the line.

BAD SYNTAX IN MAX= NEAR _____

The MAX= option in the PLOT command was being exercised but no maximum value was specified or the value was non-numeric. The line is reprinted at the end of the message from subroutine LOOPER. The user should re-enter the line.

BAD SYNTAX IN NAME= NEAR _____

Subroutine LOOPER identified the string "NAME=" in the PLOT command being processed but there was apparently nothing else on the line. The line is repeated at the end of the message for the user's edification. The suggested user action is to omit the "NAME=" string from the PLOT command but do include a variable list on the line. The "NAME=" convention is not implemented in the SIMCON version running at OSU, so this message should not appear.

BAD SYNTAX IN RHS OF SET COMMAND _____

When using the SET command, various variables are set equal to either numerical or logical values. These numerical or logical values appear on the right-hand side of an equals sign. This error

message occurs because SIMCON (subroutine VGTS) cannot interpret the right hand side of a SET command. This side must contain numbers, the strings "ON" or "OFF" or combinations of numbers and asterisks like FORTRAN DATA statements. Any other characters found on the right side of SET commands [such as trying to set the contents of one variable equal to the contents of another variable, SET POPMAX = POP(10)] will generate this message.

BAD SYNTAX IN TIME COMMAND

The user has requested a reset of the model to a time that is illegible to SIMCON (subroutine TIMEF) is something other than digits. User action is to re-enter the command.

COMMAND STACK ERROR

Three conditions emanating from subroutine MASTER, can generate this error message

1) Attempt was made to take something off stack and bottom got above top. Then this message would immediately follow message IRET BASE TOP NW LEN
2) A set of multiple commands was encountered, the first to be processed, the rest to be added to stack. At this point, the stack either became full and was cleared.
3) An intervention command (AT) was taken from the stack and either caused the stack to overflow (not likely) or the bottom to exceed the top. Either way stack is cleared and command echoed.

COMMAND STACK OVERFLOW

This can also occur in one of two ways:

1) A pop of the stack resulting in a stack full status. This is logically impossible in STRMGR.
2) A push of the stack resulting in a stack empty status. This is also logically impossible.

Therefore, a user should not be able to generate this message via MASTER.

COMMAND STACKING ERROR, MACRO EXECUTION ABORTED

Here everything with the macro is fine but when subroutine MACROS tried to put the macro onto the command stack, the stack was already full and there was no room for it. Subsequently, the macro was not performed and the command stack itself was cleared.

COMMAND TOO LONG

A command contained on more than one line extended beyond 149 characters. The entire command is reprinted and must be reentered in a shorter version, possibly using macros. Subroutine MASTER made this complaint.

COMMON STATEMENT >660 CHARACTERS, EXCESS IGNORED

COMMON statements are being read (either from COMMON or SUPCOM by subroutine RDSTMT) for the purpose of constructing the symbol table. Any statement in these files that is composed of too many (>9) continuation lines, is truncated when the 660 character limit is exceeded. This does not mean that multiple COMMON, REAL or INTEGER statements cannot be used. There is no limit to the number of COMMON statements in the user's common subroutine. So the solution to this error is multiple statements rather than long, continued statements.

ERROR IN COMMON BLOCK DUMP FILE I/O AT _____

In processing some sort of a VIEW command, subroutine VWGET attempted to get information about a certain variable in a certain year for which no information was present in the auxillary dump file. Specifically, there was no record of the year that appears at the end of the message. This is a sinister error and hopefully should not appear.

ERROR MAXIMUM SUBSCRIPT EXCEEDED FOR _____

This error message comes from subroutine STGT which is called when SET, DISPLAY, PLOT, UNPLOT, ONSTAT, STATS, VIEW, or PRINT commands are executed. For this message to be issued, STGT has looked in the symbol table for the variable in question and found that the original dimension of the variable in user's blank common is less than the subscript specified in the command being processed. User should verify that the blank common definition is in agreement with the user's intentions.

ERROR NEGATIVE SUBSCRIPT FOR _____

A negative subscript (which is not allowed in FORTRAN IV) was deleted by subroutine STGT in the parameter but for any of the following commands: SET, DISPLAY, PLOT, UNPLOT, ONSTAT, STATS, VIEW or PRINT. The command will probably be ignored.

GENERALLY BAD SYNTAX NEAR _____

A generally vague message from subroutine STGT because a myriad of things may have gone wrong to cause its appearance. Probably coming from a SET, DISPLAY, PLOT, UNPLOT, ONSTAT, STATS, VIEW or PRINT command. SIMCON managed to find the variable name in the parameter list but the subscript(s) associated with it were generally garbled. They were non-numeric or something equally obtuse. User should re-enter command.

EXCEEDED MAXIMUM SAVED STATED, 100 MODEL STATES RETAINED

This is an insidious problem. Only 100 model states can be saved because of size limitations. However, the user is not made aware of this until the last iteration of the model (when #IYEAR = #IYEND by subroutine RTSTUF) by which time numerous iterations of the model may have occurred. These results are inaccessible to VIEW, GRAPH, PRINT, etc. commands. From a cost standpoint the user should evaluate

(#IYEND - #IYBEG) / #NYSKIP

before using the SIMULATE command so that CPU time is not wasted on inaccessible results. If more than 100 years are to be simulated, do the simulation in sets of 100 years or modify #NYSKIP.

IDUMP MUST BE ON FOR TIME COMMAND

This message is issued by subroutine TIMEF and is issued when a TIME i command is encountered. This is an attempt by the user to reset the model to a previous state by resetting the content of users common block to some pre-existing state. These pre-existing states are stored at each model iteration on an auxiliary file (unit 13) but only if flag #IDUMP is on (set to 1). If #IDUMP is not on, the previous model states have not been saved and there is no point in returning to them. User action: DISPLAY #IDUMP to see that it is set to 1, if it isn't, SET #IDUMP=1.

IDUMP MUST BE ON FOR USE OF THE VIEW COMMAND

Any VIEW, PRINT, GRAPH, GET or FILE command processed by VIEWX use of the contents of the auxiliary file that maintains a history of the model states. If this file has not been filled (#IDUMP=0), these commands have no meaning. That is the information content of this message. In order to use these commands, the user must SET #DUMP=1 and repeat the SIMULATE command.

IDUMP NOT SET ON

The flag #IDUMP shows whether the variables in the user's blank common block are being stored at every time step. If they are being stored (#IDUMP=1), then statistical analysis can be performed by subroutine STATS. If they are not being stored in an auxiliary file, no analysis can be performed. User action is to DISPLAY #IDUMP and if it is not on, SET #IDUMP=1.

ILLEGAL SYNTAX

Several ways of getting this message from subroutine NAMVAR:

1) Three words must be entered when using the NAME command. If more or fewer words are used, this message results.
2) Arrays, or portions of arrays may also be renamed but this command must be able to find the parentheses and translate what is inside of them into standard FORTRAN or SIMCON jargon. If the subscripts cannot be interpreted correctly, then this message is issued. Negative subscripts are also flagged by this message.

INTERNAL PARAMETER ERROR

In working its way around the "FROM" and "TO" character strings in the SIMULATE command, subroutine SIMLT got lost. One remedy of this might be to enter the command again but leave out "FROM" and "TO". If this does not work, the user may try various combinations of the TIME i, SIMULATE i, or SET #IYEAR commands to achieve the desired results.

INTERVAL BOUNDS IMPROPER

If the optional LO and HI bounds were specified in the STATS command but LO HI, this message is issued by STATS. The number of variables for statistical analysis is set to zero and the OSTAT flag is turned off, effectively negating the command. User should reenter the comand making sure LO < HI.

IRET	BASE	TOP	NW	LEN
#####	#####	#####	#####	#####

Commands to be executed by SIMCON are loaded onto a stack, building up until they are executed. There are pointers that "manage" the stack by pointing to the next command to be performed (TOP) and the bottom of the stack (BASE). This message is issued when somehow the pointer for TOP is below the pointer BASE. This error is generated by subroutine STKMGR.

KEY-WORD TOO LONG, _____

Key words are only allowed to be 10 characters in length. Exceeding this stipulation results in this message by subroutine MACROS.

KEY-WORD UNKNOWN, _____

Use of the order independent form of a macro with a keyword that does not correspond to the keyword used when the macro was established diagnosed by subroutine MACROS. Recommended action - redefine macro with keywords user can remember.

MACRO BUFFER OVERFLOW, MACRO TOO LARGE

Two possible methods of generation (both from subroutine MACROS):

1) A macro can only contain so much information (either commands or parameters). If that amount (200 packed characters) is exceeded at the time of building, this message is issued and the macro is not constructed.
2) Macros can also contain key-words that are substituted when the macro is executed. The size limit still holds however. If a short keyword was used in the macro's definition, say X and was later substituted with the variable name BAROMETRIC-PRESSURE, these kinds of difficulty may arise.

MACRO LIBRARY ALREADY FULL

SIMCON is configured to allow for 64 unique macros that means re-defining an existing macro still counts as only one. If a macro is being defined and this number is exceeded, this message is generated by subroutine MACROS which maintains the dictionary. Solution: rename an existing macro or modify the entries in common block /MET/ defined in BLOCK DATA.

MACRO NAME TOO LONG

The macro encountered either had a blank as its first character or was longer than 21 characters when it is being defined. Subroutine MASTER will complain about either of these situations and the user is reminded of the maximum length of a macro name of 21 characters.

MACRO NOT FOUND

A CALL <macname> was executed and the macro had not yet been defined. Subroutine MASTER consulted the dictionary of macros and did not find the macro in question. The user must define the macro in the manner set forth on p. 21 of the Users Guide.

MAXIMUM SUBSCRIPT EXCEEDED FOR _____

Subroutine NAMVAR has checked the symbol table for the variable receiving the synonym and knows the number of its subscripts and the maximum size of those subscripts. If the user has attempted to access a non-existent subscript or a subscript value of the array that exceeds its dimensioned size, this message is printed along with the array name in question. User must refer to original array declaration in the common definition file (unit 2).

NEW COMMAND EXCEEDS MAXIMUM, COMMAND NOT ALLOWED

There is room provided for 45 commands in SIMCON. 34 of these commands are pre-defined i.e., GRAPH, VIEW, TIME, etc. This means that there is room for 11 substitutions via subroutine NAMCMD. Understand that using COMMAND to rename a SIMCON command does not make the original command obsolete; it can be referenced now in two ways. Therefore, this message will appear on the twelfth issuance of COMMAND. User action can only be to revert to macros at this stage for redefining commands.

NEW NAME ALREADY USED _____

This message can only be issued when a condition is met that should terminate SIMCON before the message is issued. If this message appears, the program does not function as the documenter assumes. This error should be brought to the attention of a systems programmer.

NO CLOSING PARENTHESIS IN COMMON STATEMENTS
RECORD CAUSING ERROR WAS

In attempting to read either the user's blank common from file COMMON or SIMCON's /SUPCOM/ common block from file SUPCOM, the subroutine CMREAD found an error. An array was defined in the common block without a closing parenthesis. These files should be reexamined to find the error.

NO SIMULATION RECORD FOUND

The current year is less than or equal to the beginning year (implying no SIMULATE-type command has been issued) or the value of #NYSKIP is negative. Any of these situations make the use of VIEW, PRINT, GRAPH, GET or FILE commands processed by VIEWX nonsensical. Some simulation needs to have taken place prior to examination of results from that simulation.

NO SIMULATION RECORD FOUND FOR TIME _____

User specified a time that was not saved on unit 13 probably because it did not fall within the bounds of a previous SIMULATE command (or there was no previous SIMULATE command). Subroutine TIMEF diagnosed this. User can DISPLAY #IYBEG and #IYEND and adjust the value of the TIME command accordingly.

NO VARIABLE LIST

Subroutine STATS noticed that the STATS command was followed by nothing. The user must specify for what variables statistical analysis is desired. The ONSTAT command followed by nothing discontinues statistical analyses and should not issue this message.

OLD COMMAND NOT FOUND

The user is trying to rename a SIMCON command but the SIMCON command specified for renaming does not exist (or at least NAMCMD could not find it). User action: reenter the line with proper spelling.

ONLY 10 VARIABLES REPORTED UPON

The STATS/ONSTAT commands will handle statistical analyses for only 10 variables at a time through subroutine STATS. The definition of variable here is simple variables and array elements. Therefore if the user wanted statistics on POP (1...11), this error message would occur. There does not appear to be any stipulation against merely splitting more than 10 variables up among more than one STATS/ONSTAT command. In this case, the hypothetical user above could issue two STATS commands and achieve the desired results (if not quite so elegantly).

OLD NAME NOT FOUND

The variable name to which the user is trying to attach a synonym in subroutine NAMVAR, is not in the symbol table.

PARAMETER ERROR

Issued because of a bad COMMAND statement by subroutine NAMCMD. There were fewer than three words found in the command. Recheck the statement entered and re-enter paying special attention to spacing (one space separating COMMAND from <oldname> from <newname>).

PLOT Q FULL - CAN'T ENTER _____

Contrary to the SIMCON Users Guide, the plot queue can only contain 10 variables, if this limit is exceeded, this message is issued along with the name of the variable attempting to enter the queue. The variable of interest can be added to the plot queue but only at the expense of some other variable which must be UNPLOTted first. If the limit of 10 variables is too confining, subroutine RPUTS2 (which issued the message) can be modified to allow more variables as well as a host of array redimensionings in common block /SUPCOM/ (see page B6 of Implementer's Guide).

READ COMMAND UNAVAILABLE

A READ command was processed resulting in a call to subroutine CMDFIL which, in the general release of SIMCON, is a dummy subroutine. The subroutine's purpose in life is to issue this message to the user and return. To allow execution of command files that are <u>not</u> local files titled BATCH, this subroutine must be modified (see p. 12 of Implementation Guide). This routine is complete in the OSU version of SIMCON so this message should not appear.

SIMULATION "FROM" TIME INVALID

The string following FROM in the SIMULATE command was non-numeric according to subroutine SIMLT. Re-enter line. Note as long as it is a number, even a very small one, this message will not be delivered.

SIMULATION "TO" TIME INVALID

Multiple ways to get this message all in subroutine SIMLT:

1) The value appearing in conjunction with the CONTINUE command was unintelligible to SIMCON. Re-enter line.
2) Using the SIMULATE TO --- version of the command, the value was either non-numeric or negative. Simulating backward in time is not a valid operation in SIMCON. Re-enter line.
3) Using the SIMULATE FROM -- TO -- version of the command, the value for TO could not be distinguished by SIMCON. Re-enter line.

4) When this message is issued after the GO command, the value of the increment was either non-numeric or was negative. That trick won't work with this command either. Re-enter line.

STACK MANAGER ERROR

When attempting to execute an AT command at the time specified by the AT definition, subroutine ATS invoked the stack manager. The command string following the AT was loaded onto the command stack but the command stack was filled in the process and consequently the command stack was cleared.

SYNTAX ERROR IN "AT" COMMAND

This error is generated when an AT command is processed by subroutine ATS. The complaint here is that the second word in the line is not one of the following: a number (representing time), ALL, LIST, or CLEAR. Those are the only entities that are allowed to follow AT in a command line.

TIME SET TO____SINCE MODEL STATES WERE SAVED EVERY____TIME STEPS

This is not really an error message. TIMEF is informing the user that SIMCON could not comply with their wishes precisely. This may come about perhaps by requesting TIME be set to an odd numbered year when the simulation was begun on an even numbered year with #NYSKIP=2. In this case TIME would be set to the next lower year. User action: if user is desperate to set time to that specified, #NYSKIP may be reset or the beginning year of the simulation may be reset. In either case, the simulation must be redone.

"TO" TIME LESS THAN "FROM" TIME IN SIMULATE COMMAND

SIMLT has diagnosed a problem. When using SIMULATE, the time to simulate to, must be greater than the time at which the simulation was begun (FROM). When using CONTINUE, the value used must be greater than the current year. Check #IYEAR and reset it if necessary to get to the year stipulated in the CONTINUE command, but don't try to go backwards in time.

TOO MANY DUMP PERIODS (_____), ONLY 100 USED

SIMCON will only store results of a simulation for 100 times intervals. When an attempt is made to VIEW, PRINT, GRAPH, GET or FILE a variable list through subroutine VIEWX and there are more than 100 intervals, this message appears. The number of intervals is in integer arithmetic. If a simulation is to run for less than 100 years, this should not be a problem. If a simulation is to run longer than 100 years, some adjustment will have to be made to #NYSKIP with its inherent loss of resolution to circumvent this error or the simulation could take place in sets of 100 years.

UNABLE TO INTITALIZE BECAUSE THE COMMON FILE IS ABSENT
SYSTEM DUMP FILE EMPTY OR BAD, CANNOT RESET

This is an initialization failure message issued by subroutine INIT. One of the first things SIMCON needs to do is build the symbol table which is referred to often. The conventional way of building this is through access to the file COMMON (p. 32 of Users Guide). If the file is not present, SIMCON can reconstruct the symbol table from the file SYDUMP which may have been created in a previous SIMCON run using the SAVE command (p. 20, Users Guide). If neither of these files are present, this error message is generated. The remedy may be as simple as making one of these two files local (CYBER's /GET command) or as difficult as building the COMMON file of the user's variables to be included in the symbol table.

Use of the RESET command may also generate the second line of this message. This is an attempt to restore the system state to that saved by the SAVE command. SIMCON's complaint here is that no SAVE was done, consequently SYDUMP doesn't exist and a RESET cannot be accomplished.

UNDETERMINED TYPE IN COMMON FILE.
RECORD CAUSING ERROR WAS

This error is caused by subroutine CMREAD of the first character in a line is not an I (for INTEGER), L (for LOGICAL), R (for REAL), C (for COMMON), or D (for DIMENSION). A command line that contains no blanks also generates this error. The offensive line will be echoed so the error can be identified.

UNKNOWN COMMAND

Two sources for this message from subroutine MASTER:

1) The command being processed was not an ordinary SIMCON command. It was checked against the macro dictionary and found lacking there too, or macro library empty.
2) The command has been identified as being a system command (having a $ in the first position). This gives user access to installation commands outside SIMCON. Issuance of this error message implies that provision was not made for the particular system command entered in SUBROUTINE SYSCMD. This subroutine is user-defined and appears as a dummy in the general release version of SIMCON.

WRONG NUMBER OF ARGUMENTS FOR SIMULATE COMMAND

Subroutine SIMLT is unhappy.
There are several forms of the SIMULATE command. It can be as simple as SIMULATE which sets #IYEAR=#IYBEG, to as complex as SIMULATE FROM 1980 to 2000 with intermediate forms leaving out the words FROM and TO or having only one year value. But that's it. Adding more to the command than its most complex form above will generate an error.

One word of caution. The command SIMULATE FROM 10 is interpreted as taking the model from the current value of #IYEAR to 10, if that is a positive direction. The word FROM is not expressly noticed by SIMCON so it won't straighten out any confusion. It assumes the users know what they are doing.

*** COMMON TABLE SIZE EXCEEDED

The symbol table size has been exceeded. SIMCON cannot keep track of as many variables as the user wishes. Either reduce number of variables or write a BLOCK DATA subroutine titled HCOM as described on p. 38 of User's Guide to allow for storage of more variables. This maximum limit is, of course, machine dependent, but on the CYBER this number is 200 variables (see p. B3 of Implimenter's Guide for more detail). Generated by subroutine ENTR.

$ COMMANDS NOT AVAILABLE

A call issued to subroutine SYSCMD whenever a command is processed that contains a $ as the first character. In the general release of SIMCON, SYSCMD is a dummy subroutine containing only the statement generating in this message. This subroutine is functional in the OSU installation so this error should not appear.

_____ WAS NOT FOUND IN COMMON

A variable in the parameter list of SET, DISPLAY, PLOT, UNPLOT, ONSTAT, STATS, VIEW, or PRINT was not found in the symbol table by subroutine STGT. User should check the blank common block to see that all pertinent varibles have been included there.

_____ NOT IN QUEUE

An attempt was made to UNPLOT a variable that was not in the plot queue. Clever, but subroutine LOOPER was not fooled. DISPLAYing #NPLOT will show the user how many variables are in the queue but the user must keep track of what they are.