RM-76-24

WISSIM: AN INTERACTIVE SIMULATION CONTROL LANGUAGE

J.S. Buehring

September 1976

Preface

This report is one of a series describing a multidisciplinary multinational IIASA research study on the Management of Energy/Environment Systems. The primary objective of the research is the development of quantitative tools for regional energy and environment policy design and analysis--or, in a broader sense, the development of a coherent, realistic approach to energy/environment management. Particular attention is being devoted to the design and use of these tools at the regional level. The outputs of this research program include concepts, applied methodologies, and case studies. During 1975, case studies were emphasized; they focused on three greatly differing regions, namely, the German Democratic Republic, the Rhone-Alpes region in southern France, and the state of Wisconsin in the U.S.A. The IIASA research was conducted within a network of collaborating institutions composed of the Institut fuer Energetik, Leipzig; the Institut Economique et Juridique de l'Energie, Grenoble; and the University of Wisconsin-Madison.

This memorandum describes a simulation control system that has been used with several of the models used in this study.

Other publications on the management of energy/environment systems are listed in Appendix E at the end of this report.

W.K. Foell
September 1976

Abstract

WISSIM is a computer simulation control language designed to provide a convenient interface between simulation models and their users, and to facilitate the writing of such models. It has been implemented on the UNIVAC 1110 at the University of Wisconsin-Madison, and on the PDP-11 at the International Institute for Applied Systems Analysis. WISSIM can be linked with a user-written FORTRAN program that contains the code for the actual simulation. The control language has commands to interactively modify and display parameters, to control the simulation program, and to produce graphs and reports of results from one or more simulation runs. This report provides a general introduction and describes the commands.

Acknowledgements

Special acknowledgements go to two members of the Energy Systems and Policy Research Group (ESPRG) of the University of Wisconsin: Patricia Kishline who worked on the first version of WISSIM, and Lee Kirchhoff, now with Bell Telephone Laboratories, who was instrumental in the design and programming of the present version. I would also like to thank the other members of ESPRG for struggling through the early phases of WISSIM with us and providing many helpful suggestions. Bernhard Schweeger of IIASA Computer Services provided the graphics interface for the IIASA implementation and much valuable assistance with the IIASA PDP-11 computer.

TABLE OF CONTENTS

WISSIM: An Interactive Simulation Control Language

I. Introduction

A. Design Constraints

1. Purpose

WISSIM (WISconsin SIMulation), an interactive computer simulation control system, was originally developed to run the Wisconsin Regional Energy Model (WISE), a large FORTRAN simulation model composed of many distinct submodels*(Foell, 1975). Although designed specifically for the requirements of WISE and the uses envisioned for that model, considerable care has been taken to keep WISSIM, the command language, distinct from WISE, the set of models, so that WISSIM would have a wider application to other simulation models. The implementation and continued development of WISSIM at the International Institute for Applied Systems Analysis (IIASA) has demonstrated the value of this approach.

The study of energy use in Wisconsin by the Energy Systems and Policy Research Group, of which the WISE Model is an integral part, was begun in late 1972 within the Institute for Environmental Studies of the University of Wisconsin. The focus of the model was to describe the Wisconsin energy system including demand, supply, and environmental and social impacts in a manner that would shed light on policy questions of an individual, local, and regional nature with emphasis on policy issues of interest to regional decision-makers.

All of the eventual uses of the model, particularly its application to other regions of the world, were not envisioned at the beginning of the project. However, a number of important requirements were recognized and considered in the design of both WISE, the model, and WISSIM, the control system. Some of the more important are described below.

---

*The use of the words "model" and "submodel" to describe the various parts of the Wisconsin Regional Energy model should not prove confusing in the following discussions. The overall model (WISE) consists of a set of submodels (e.g., the Residential Model, the Generating Capacity Model, and the Environmental Impact Model) each of which is a model in its own right. Thus, the Residential Model is called a model when referred to as an independently running model but a submodel when discussed in relation to the Wisconsin Regional Energy Model.

- The first and initally most important use of any model is as a research tool by its developers. The development of a model is usually a learning process for the modeller that continues over a long period of time. During that period, a great deal is learned about the system being modelled and many improvements on the original concepts are made as reflected in continually upgraded versions of the model. When a satisfactory model of the system has been devised, it may be used as the basis for research analyzing the implications of alternative policies which import on the system. Examples of such applications of the WISE model include the work by the Energy Systems and Policy Research Group (ESPRG) at the University of Wisconsin on forecasting future electricity generation requirements for Wisconsin under a variety of assumed policy alternatives (Buehring, 1974) and a study of the effects of alternative commercial building codes on energy conservation (Mitchell and Jacobson, 1974).
- A second but equally important requirement imposed on both WISE and WISSIM was that the system must be useful and useable not only by the researcher but also by scholars working on similar energy policy analysis at other institutions and technical advisors and decision makers concerned with using the model for policy analysis. That is, the models had to be useable for demonstration purposes in workshop and conference settings with conference participants to include technical advisors and decision-makers as well as scientists from universities, research institutes, and government agencies. This character of the WISE and WISSIM system was demonstrated at the University of Wisconsin at two major conferences: A French American Conference on Energy Systems Forecasting, Planning and Pricing (Cicchetti and Foell, 1975) in which a demonstration of the model was given, and a conference for regional, state, and local energy/environment decision-makers from across the United States that included several demonstrations and a workshop session in which the participants

could actually specify inputs, modify parameters and examine the results. In addition, many informal smaller demonstrations have been given to interested people from both within and outside of the University. An interesting demonstration with a slightly different focus occured at the 1975 Engineering Exposition at the University of Wisconsin. The transportation model was simplified and used in an interactive mode with members of the public to project future demand for oil in Wisconsin under different policies and consumption trends. A modified version of the residential model was also used to analyze the dollar savings in a typical house by incorporating a set of measures specified by the audience.

- Eventually the models were to be turned over to Wisconsin state agencies to modify as necessary and use to provide technical data related to decisions made on the state level. The possibility of physical transfer of the models to state computers had to be considered even though their continued use on the University computer was a possibility.
- Methodology and perhaps relevant portions of the actual models could be transferred to other regions for analysis of their energy systems (the use of the models at IIASA to study the Rhone-Alpes region of France and a Bezirk in the German Democratic Republic (Foell, 1976) is perhaps the most notable example of such a transfer) further emphasizing the need for transferability.

2. Need for a Separate Control Language

The intended uses of the models listed above imposed some rather stringent constraints on the model builders. The model had to be divided into submodels that can run independently or interdependently. An individual researcher usually works on the detailed development of one or two submodels and must be able to write, test, and make meaningful simulations with those submodels without depending upon the rest of the model to be

fully developed and running concurrently . On the other hand, for the eventual study of feedback loops in the energy system such as the interrelationships between supply, demand, and price, the model must be constructed as a coherent whole. The necessity to run the models both independently and as a whole demanded a control system that was independent of the models - that could be used with any model or combination of models. For other reasons as well, it was desirable to separate the control system, that eventually became WISSIM, from WISE. The programming necessary to provide input and output capabilities including reinitialization of the model for repeated simulations, provisions for dynamic interaction during simulations, and producing reports and graphs on request is both time-consuming and complex. It was essential that this programming not be repeated for each model. For the users of the model, a wide variety of people with different degrees of familiarity with computers, programming, and modelling, a standard method of interaction with all of the models is desirable. A further advantage of the separation of the model and the control language is that the use of WISSIM is not limited to the WISE model.

3. Simulation Modelling

The simulation modelling technique was chosen for the study primarily for its flexibility. A variety of analytical techniques were likely to be used in the course of the development of the model and the simulation approach appeared to be a convenient method of integrating them. This has proved to be true. The simulation structure also allows selected components of the system to be modified without revising the entire system and allows for the submodel independence discussed above. Optimization techniques were eliminated from consideration for the basic structure of the model because of the difficulty of choosing an objective function and suitable constraints for a large system that includes many decision-makers (individual consumers, small businesses, corporations, and state, local, and national governments) with so many conflicting objectives. Optimization has however been considered for use within the infrastructure of

some of the submodels. The model was also intended to be a descriptive model rather than a prescriptive one. Rather than attempt to provide the decision-maker with one "best" set of policies based upon an objective function that could not possibly include all the values of the decision-maker, the purpose of the model was to be able to generate for a reasonable set of alternative policies, their effects on specific issues of major importance in the energy system. The simulation model has proved to be very useful in generating such "scenarios". The decision-maker can then incorporate the results of the model into the decision, perhaps using a technique such as multi-attribute decision analysis (Buehring, et al., 1976). A general flow diagram of WISE as it finally evolved is shown in Figure I-1.

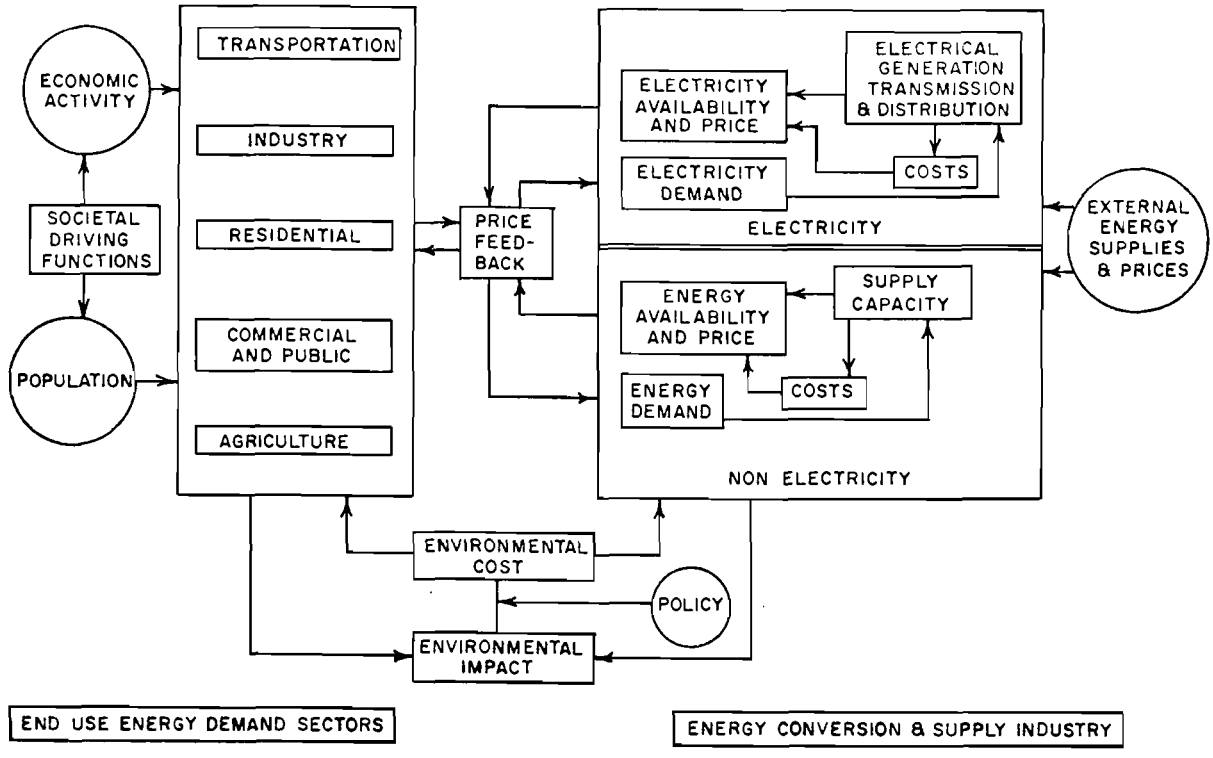


Figure I-1: The Wisconsin Regional Energy Model
Source: (Foell, et al., 1975)

4. Choice of a Programming Language

FORTRAN was chosen rather than DYNAMO (Forrester, 1968) and other simulation languages for a variety of very practical considerations:

- The FORTRAN compiler at the University of Wisconsin Computing Center was available, well-documented, and relatively error-free.
- It was anticipated that of the many researchers working on various sections of the mode, most of them would already be familiar with FORTRAN.
- Although the extent of the eventual distribution of the models was not known, it was assumed that they would also be used at other locations. A FORTRAN compiler is available at most computer installations. In addition to Wisconsin, the models have been applied at IIASA to the Rhone-Alpes region of France and to a hypothetical region of the German Democratic Republic. One or more of the WISE submodels have also been sent to several institutions within the United States.

5. Summary

The overall design of WISSIM was then defined within the framework of the considerations outlined above. That is, WISSIM was to be a control system for FORTRAN simulation models. Since models may change frequently, the linkage between WISSIM and the model must be simple and relatively inexpensive. The language is to be used by a large number of people of varying degrees of sophistication in modelling and programming. It must be easy to use and yet provide the degree of flexibility needed to produce reports and graphs for both researchers and decision makers. Its necessary control functions are to:

- Change the values of input parameters.
- Produce reports and graphs of results on request of the user.
- Control the prgress of the simulation including the ability to examine results and revise parameters during the course of a simulation
- Restore the model to a previous state (including the initial state) so that a series of scenarios or parameter studies can be done.

The flow of information to and from WISSIM is shown in Figure I-2.

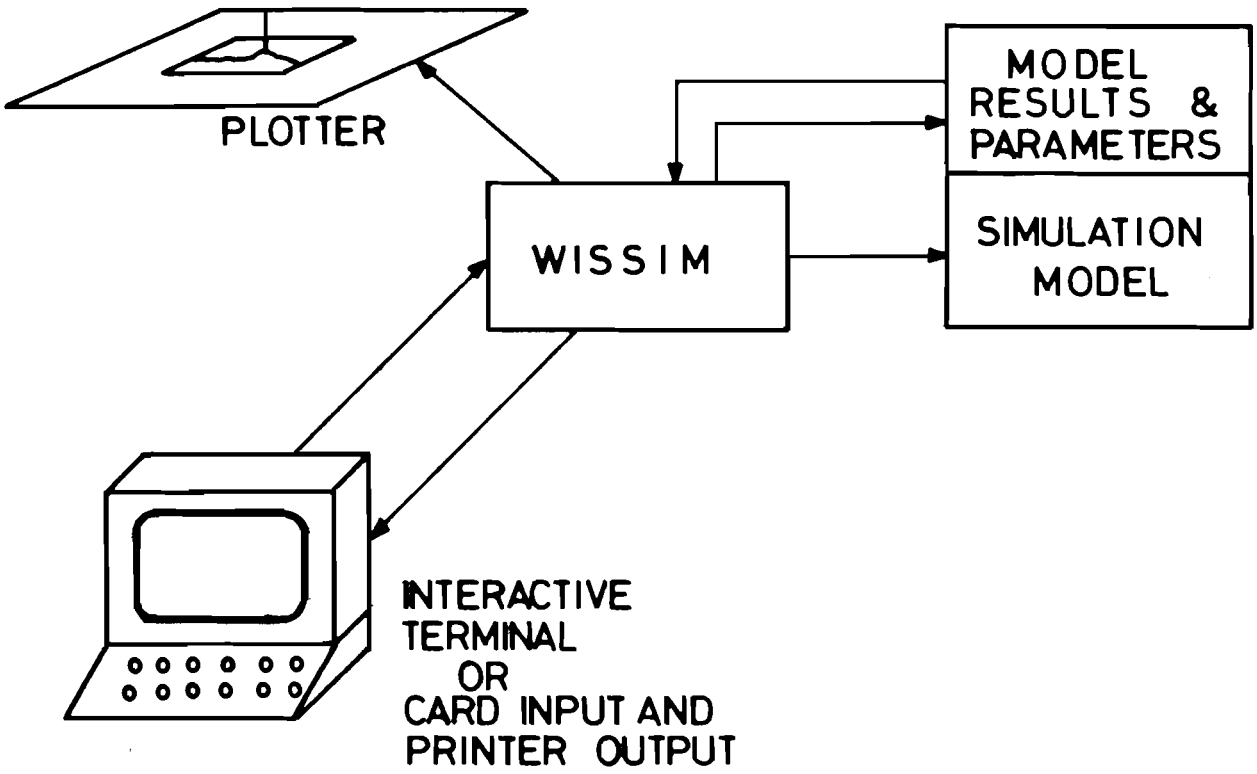


Figure I-2: Flow of Information to WISSIM

The primary purpse of WISSIM is to provide a convenient interface between a simulation model and its users.

B. Applicability

The final version of WISSIM is applicable to a wide class of FORTRAN simulation models. The most important criteria is that the simulation occurs over discrete time periods with a general flow illustrated by Figure I-3. WISSIM is not well-suited to simulations that are driven by discrete random events

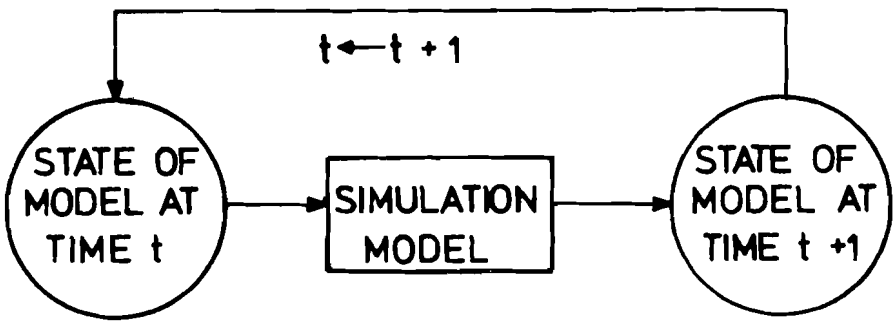


Figure I-3: Flow Diagram of a Generalized Simulation Model

such as simulation of an inventory control system or the operations of a group of check out counters in a grocery store. Simulation languages such as GPSS or SIMSCRIPT are better suited to that type of application (Meier, Newell, and Pazer, 1969).

To make full use of the capabilities of WISSIM, the model must satisfy the following conditions:

- It must be written in FORTRAN. FORTRAN suffers some limitations for model building but offers the important advantages that it is more widely known than any other programming language and relatively simple to program in.
- The state of the model must be completely determined by the values of variables that the designer places in certain common blocks. Minor restrictions on the format of the common block are imposed (e.g., no more than three levels of subscripting, no single subscript greater than 512).

WISSIM has been implemented and used both at IIASA and at the University of Wisconsin-Madison where its primary use was with the Energy Systems and Policy Research Group.

II. WISSIM Features - A General Description of WISSIM

A. Introduction

This Section discusses some of the WISSIM commands and features. Section IV contains specific formats and options for commands that are referred to in this section.

DEFAULT VALUE.	The value that a variable takes on if not explicitly assigned another value, i.e., the value it has "by default".
MODEL COMMON BLOCK.	A common block that the designer of the model has defined to WISSIM. Creation of model common blocks is discussed in Section III of this report.
MODEL VARIABLE.	A variable found in a model common block.
TEMPORARY VARIABLE.	A variable defined by WISSIM DEFINE command. Temporary variables are in the WISSIM common block TMPVAR. All temporary variables begin with a dollar sign ($).
SYSTEM VARIABLE.	One of a special group of variables found in the WISSIM common block SYSVAR beginning with the letters IZZ.

B. Control

The central function of WISSIM is control of the simulation model. The SIMULATE command executes the model for a specified time period and simulation increment. It also assigns a user-specified run identifier, called the runid, to the simulation run. Values from a particular simulation may be retrieved later by reference to runid. The simulation time period and increment may be either integer or real numbers. A sample SIMULATE command:

SIMULATE CASE1 FROM 1970 TO 2000 BY 5 is interpreted as follows. Values will be saved under the runid CASE1. Simulation will be in five-year increments from 1970 to 2000 and the model will be called seven times.

The course of the simulation may be altered with AT commands. AT commands precede the simulation(s) to which they apply and

allow a user to specify prior to simulation an action to be carried out by WISSIM during simulation. They may be used to 1) print the values of variables (DUMP), 2) change the values of variables (SET), 3) call special user-written routines (CALL or WRITE), or 4) temporarily suspend execution of program (PAUSE) at any time during the simulation. An AT command with no time specified is executed at every time interval.

A sample AT command:

AT 1975 SET MRATE = 0.005. The value of the user-defined parameter MRATE is changed to 0.005 at time 1975 during all subsequent simulations. Such a command or set of commands may be used to simulate the imposition of a new policy or constraint at some time after the start of the simulation.

There are eight policy variables (IZZPOL(1) - IZZPOL(8)) that may be used to further control the execution of AT commands. A policy may be specified on an AT command, in which case the AT command is executed during a given simulation only if that policy variable is currently the same as the policy on the AT command. The values of policy variables may be changed with SET command. The policy variables with the AT commands may be used to set up many different scenarios that may be run in different combinations without re-entering the commands.

A simple AT command with the policy option:

```
AT 1975 POLICY(2) LOW SET MRATE = 0.005
SET IZZPOL(2) = 'LOW'
SIMULATE CASE2 FROM 1970 TO 2000 .
```

Since IZZPOL(2) is set to 'LOW' prior to the simulation, the SET command will be executed at 1975. Then another simulation could be made without that command:

```
SET IZZPOL(2) = 'HIGH'
SIMULATE CASE3  FROM 1970 TO 2000  .
```

A delete option is also available to elimate unwanted AT commands. They may be deleted by time, by policy, or by AT command number which is assigned and printed when the AT command is first entered. Some examples:

```
AT DELETE          Deletes all commands.
AT 1975 DELETE     Deletes all AT commands with a time
of 1975 specified.
```

AT DELETE 5,10,12 Deletes AT commands 5, 10, 12. Other AT commands retain their original numbers.

The PAUSE command halts the simulation to allow the user to examine and/or change variables and parameters. A message PAUSE at time is printed and WISSIM waits for more input. At that point, values of variables may be displayed or changed. A CONTINUE command is given to continue the simulation.

C. Input

The values of parameters in a simulation model may be changed with SET commands before a simulation or during a PAUSE in the simulation. SET commands may also be used with an AT command to change a parameter at any or every simulation interval. A variable or array may be set to an integer, a real number, a string constant, or an arithmetic expression. There are also several WISSIM functions that may be included in the arithmetic expressions.

The DEFINE command allows a user to define additional variables at execution time. Defined variables follow the same rules as normal FORTRAN variables except that the first character must always be a dollar sign ($). They may be printed and graphed be WISSIM commands just like any other variable. For example, DEFINE $EPCAP = ENERGY/POP A new variable is defined called $EPCAP and at every time period during subsequent simulations, it will be given the value ENERGY/POP.

D. The SAVE File

Upon user request (via a SAVE command), WISSIM will save the values of variables on a temporary auxillary storage file. Saved values may be used to obtain reports or graphs from previous simulations, or to restore the model to a saved state with a RESET command (initial values are usually saved so that the system may be restored to its original state to begin another simulation).

There are two types of SAVE commands. The first type contains an "AS runid" clause and may be called an immediate save. The values of the specified common blocks are saved immediately and given the name in the "AS runid" clause for future reference. The second type contains no "AS runid" clause. No immediate

save is performed but during following simulations the values of the specified common blocks are saved with the run identification given on the SIMULATE command. The values are saved for every time increment by default but this may be altered by changing the value of the system variable IXXSFQ. If a SAVE command is not issued before a simulation, the values are not saved and are not available for future reporting or graphing.

Sample SAVE, SIMULATE, AND RESET commands:

SAVE AS INIT ALL All common blocks will be saved immediately under the name INIT.

This command is usually used in preparation for another simulation.

SIMULATE 1970,2000

RESET INIT The simulation has changed the values of the variables in the common blocks. The RESET command restores the variables to the values they had when the SAVE command was issued.

SAVE BLOCK1, BLOCK2 Assuming that BLOCK1 and BLOCK2 are the names of two common blocks that have been defined to WISSIM, this command will cause those two common blocks to be saved every time period in all subsequent simulations.

SIMULATE CASE1 FROM 1970 TO 2000 Because of the preceding SAVE command, the values from BLOCK1 and BLOCK2 will be saved under the name CASE1 at every time period during this simulation.

E. Output

Printed or graphed output is available from WISSIM. The current values of variables may be printed (DUMP command), information from previous simulations may be printed or graphed (NREPORT and GRAPH commands), or information may be printed during simulations (SREPORT command).

The DUMP command may be used to print the values of a list of variables, arrays, or entire common blocks. The WISSIM system variable IZZPRE which has a default value of 3 may be set to the number of digits to be printed. For example, suppose ARRAY1(10) is an array dimensioned as shown, and VAR1, VAR2, and VAR3, are unsubscripted variables. Then

DUMP VAR1,VAR2,VAR3,ARRAY1 dumps VAR1, VAR2, VAR3, and all of ARRAY1.

The SREPORT command contains a list of variables whose values are to be dumped during subsequent simulations, for example,

SREPORT VAR1,VAR2,ARRAY1(2) .

The NREPORT command is used to print information from a previous simulation. A SAVE command must have preceded the simulation from which data are to be retrieved. Examples:

```
SAVE ALL
SIMULATE CASE1 FROM 1970 TO 2000 BY 1
NREPORT FROM CASE1 VAR1, VAR2 .
```

Values are saved for every year from 1970 to 2000. The NREPORT command prints them.

Reports generated by both NREPORT and SREPORT may have titles and labels. The system variable IZZRT contains the title. The name of the variable is used as a column heading in the report unless a LABEL command has been entered for that variable. Labels may be general any time a variable is reported or specific to a particular run identification. The system variable IZZPRE described with DUMP above also determines how many digits will be printed on the reports. Some examples:

```
SET IZZRT = 'OH, WHAT A BEAUTIFUL MORNING $/$  OH,
                  WHAT A BEAUTIFUL DAY  $$'
```

All subsequent reports will have that title (split on two lines where the $/$ is) centered over the report.

```
LABEL FROM CASE1 VAR1 = 'HIGH' FROM CASE2 VAR1 = 'LOW'
```

The labels for VAR1 will be used as column headings on reports from the runs called CASE1 and CASE2.

The GRAPH command in its current implementations at IIASA and Wisconsin will produce graphs on the CALCOMP plotter of up to seven variables from one or more simulation runs. The Wisconsin version also has graphics for an interactive graphics terminal. The page width and height as well as scaling information may be specified by setting the values of certain system variables as described in Section IV.B.6. A legend is optional as are titles for the graph and for the x- and y-axes. The variables may be graphed with their true values or normalized to their initial values, (i.e. divided by the initial value).

Normally time is the x-axis variable. However, it is possible to use any variable from any simulation run on the x-axis by specifying the x-axis on an XAXIS command prior to the GRAPH command (again, see Section IV.B.6 for details).

III. Linkage with the Simulation Model

The model designer must write the code, a FORTRAN subroutine, to do the actual simulation from one time increment to the next. Routines may also be written for special input, output, or calculations that are callable by WISSIM command. An initialization routine used once at the beginning of the execution may also be written.

A. Variable Name Definition

Variable names, types, and dimensions are described to WISSIM by FORTRAN COMMON, REAL, and INTEGER statements. Those statements are placed on a special file whose exact format varies depending upon the computer. In general, the statements may also be included at compile time into the simulation model although the exact procedures for doing this depends on the computer and operating system.

WISSIM can manage variables in up to fourteen common blocks. Multiple common blocks are particularly useful for large models composed of many submodels. Then when a submodel is run independently, it is not necessary to include the common blocks for all parts of the large model. Each variable name must be unique. The following special restrictions in writing the statements for WISSIM should be noted.

1) Some words are reserved to have special meaning in the WISSIM commands. These reserved words, listed in Section IV.A. may not be used as variable names.
2) Only COMMON, REAL and INTEGER statements and comments are allowed. Note that this limits the types of the variables to real and integer.
3) Variables in REAL and INTEGER statements must not precede the COMMON statement that defines them. Any dimensioning must be done in the COMMON statement.
4) COMMON statements for different common blocks may not be intermixed, i.e., all COMMON statements for the first common block must precede all COMMON statements for the second common block and so on.

Variables in each common block can be defined in two classes:

1) those that are saved whenever the common block is saved; and
2) those that are saved only when ALL is included in the SAVE command as described in Section IV.B.

Important variables that may be required for graphs usually will be placed in Class 1. Other variables should be located in Class 2 to reduce computer costs, especially in a large model with many variables. In the common block, a comment card beginning with C$$$$$ separates Class 1 variables from Class 2. If the C$$$$$ card is omitted, as is likely in small models, all variables are Class 1. An example defining variable names for a hypothetical common block is shown in Figure III-1. The configuration presented would be useful if one subroutine in the model needed only one of the two common blocks. If this is not the case, all variables could be defined in a single file and/or a single common block.

B. The Simulation Model

The main routine of the simulation model is a FORTRAN subroutine UMODEL that must do the actual simulation from one time increment to the next. UMODEL may call other subroutines, however, so the entire simulation model may be as large and complex as necessary. Subroutine UMODEL has four real parameters in the following order:

BEGIN	is the beginning time simulation
END	is the ending time of simulation
INC	is the time increment
CUR	is the current time of simulation

BEGIN, END, and INC are all from the SIMULATE command. CUR has the value of the WISSIM system variable IZZCUR. When the model user enters a SIMULATE command, UMODEL will be called once for each time interval and CUR will be set to the correct time. For example, if the command were SIMULATE 1970,1980,2 UMODEL would be called six times with CUR=1970,1972,...1980.

It is possible to check for the first simulation period by comparing the values of CUR and BEGIN. They are equal for the first simulation interval.

Since the time increment is a parameter of the SIMULATE command and is passed as a parameter to UMODEL, it is often advantageous for all equations including rates in the model to explicitly include the time increment. For example, suppose A is the capacity in kilowatts of power plants in a model of an

File "ECMN"

```
C
C         ENERGY(F,S),F=1,4,S=1,5
C                   ENERGY USED BY FUEL AND BY SECTOR OF THE ECONOMY
C
C         F         FUEL
C         1         COAL
C         2         OIL
C         3         NATURAL GAS
C         4         WOOD
C
C         S         SECTOR OF THE ECONOMY
C         1         RESIDENTIAL
C         2         COMMERCIAL
C         3         INDUSTRIAL
C         4         TRANSPORTATION
C         5         ELECTRICITY GENERATION
C
      COMMON/ECMN/ENERGY(4,5)
C
```

File "XCMN"

```
C
C         EFAC(F,T),F=1,4,T=1,4
C                   EMISSION FACTORS
C         CFAC(F,S,T),F=1,4,S=1,5,T=1,4
C                   CONTROL FACTORS
C         EM(F,S,T),F=1,4,S=1,5,T=1,4
C                   EMISSIONS
C         SEM(S,T),S=1,5,T=1,4
C                   EMISSIONS BY ECONOMIC SECTOR
C         TOTEM(T),T=1,4
C                   TOTAL EMISSIONS
C
C         T         TYPE OF EMISSION
C         1         PARTICULATES
C         2         SULFUR OXIDES
C         3         NITROGEN OXIDES
C         4         HYDROCARBONS
C
      COMMON/XCMN/TOTEM(4),SEM(5,4),EM(4,5,4)
C
C$$$$$ CLASS 1 VARIABLES (IMPORTANT RESULTS TO BE GRAPHED) ABOVE AND
C         CLASS 2 VARIABLES (PARAMETERS, ETC., THAT WILL PROBABLY NOT
C         BE REPORTED OR GRAPHED) BELOW.
C
      COMMON/XCMN/EFAC(4,4),CFAC(4,5,4)
C
```

Figure III-1: Sample Statements to Define Model Variables

energy system and ADEL has been calculated as the additional capacity required in kilowatts per year in the system. Then if the model is always to be run with a time increment of one year, the equation A = A + ADEL will work very well. However, for the additional flexibility of running the model with varying time intervals, e.g. one-half year or five years, a preferred form of the equation is A = A + ADEL * INC.

Because of the methods of implementing WISSIM on various computers, the values of variables defined in UMODEL cannot be relied upon from one simulation to the next or even from one time increment to the next unless

1) the variable is in common block; or
2) the variable was initialized to some value by a DATA statement and always has that same value upon exiting from UMODEL.

The same common blocks defined to WISSIM must of course be included in the model.

C. Initialization

Several options for initializing model data are available, and the model designer must choose one method, or perhaps some combination of methods, and inform the model users of initialization procedures. The options and a brief discussion of any particular merits or disadvantages of each follows:

1. UINIT: A special subroutine or entry point named UINIT is called by WISSIM immediately after execution begins. The call occurs before any commands are solicited from the user. This routine may be used to read a file of inital data or to perform some initialization calculations. Note, however, that UINIT will be called once at the start of the program. It will not be called at the beginning of every simulation (i.e., whenever the SIMULATE command is given).

   Since WISSIM calls UINIT, its entry point must always be provided. If no initialization is required, the entry point may be followed immediately with a return statement.
2. DATA statements: FORTRAN DATA statements may be used for initialization. Again the user must be aware that DATA statements will not automatically reset variables to their initial values at the beginning of every simulation run.

3. SET commands: A file of SET commands may be set up that are then executed at the beginning of every modelling session.

D. Special Routines and Reports

Two other subroutines or entry points may be included in the simulation model: UCALL (int) and UWRITE (int). UCALL and UWRITE are user programs that execute on request from the model user by typing a WISSIM CALL or WRITE command. They may be used to print special reports or graphs or to make special calculations. Both have a single parameter, int, which is an integer. The parameter receives its value from the parameter of the CALL or WRITE command. For example, if the user typed CALL 12, int would have the value 12.

IV. Wissim Commands

A. Introduction

This section describes each WISSIM command and its options. Numerous examples of each command are also included.

First, the general format of the commands is described. Commands are free format with each command beginning on a new line (or card) and continuing over as many lines as necessary. A space, dollar sign ($) at the end of a line indicates that the command is to be continued on the next line:

```
DUMP VAR1,VAR2 $
  VAR3,VAR4
```

Comments may be included anywhere in the command by enclosing them within double dollar signs ($$). Two examples:

```
SET DRATE = DRATE-.001 $$DECREASE RATE OF GROWTH$$
$$CASE1 HAS A LARGE PERCENTAGE OF ALL ELECTRIC HOMES$$
SIMULATE CASE1 1970,2000
```

The following items are the elements of the language. Each element is a separate entity, and is a character or a group of characters that must not be separated by spaces or commas.

1. Commands and reserved words appear in all upper case letters in the command formats of this section and must be spelled exactly as shown. WISSIM reserved words are:

ALL	CONTINUE	FROM
AND	COPYI	GRAPH
AS	COPYO	LABEL
AT	DEFINE	NORMALIZED
BEGIN	DELETE	NREPORT
BY	DUMP	PAUSE
CALCOMP	END	POLICY
CALL	FOR	RESET
SATH	SREPORT	VIEW
SATHR	STEP	WITH
SATHYR	STOP	WRITE
SAVE	TABLE	XAXIS
SET	TO	XTERP
SIMULATE		

Underlined lower case parameters are described below the command format. They are never entered exactly as shown, but replaced by a number or name.

2. FORTRAN variable names and common block names, the special WISSIM functions (e.g., SATHR, STEP), and user-assigned run identifiers follow the normal rules for FORTRAN identifiers; i.e. one letter followed by zero to five letters or numbers.
3. Temporary variables consist of a dollar sign ($) followed by one to five letters or numbers, e.g., $A.
4. Literals are enclosed in single quotation marks. If a quotation mark is necessary within the literal, two single quotation marks may be used for each quotation mark required, e.g., SET IZZGYT = 'BTU''S' would set the value of IZZGYT to BTU'S; SET IZZGYT = '''CATS''' would set the value of IZZGYT to 'CATS'.
5. Integers are formed as normal FORTRAN integers, i.e., an optional plus or minus sign followed by the required number of digits. The number may not be larger in absolute value than the largest allowed on the machine (2^{35}-1 on the UNIVAC 1110).
6. Real numbers are also formed as normal FORTRAN real constants and are restricted in magnitude again by the machine on which WISSIM is running (approximately 2.9×10^{-39} and 1.7×10^{38} except for 0 on the UNIVAC 1110). From the Madison Academic Computing Center FORTRAN V compiler manual real constants:
   (1) if positive, may be prefixed with a plus sign;
   (2) if negative, must be prefixed with a minus sign;
   (3) must contain not more than one decimal point; and
   (4) may be expressed by appending an E to the constant, followed by a signed (+ is optional) one- or two-digit integer exponent; real constants expressed in the E-form are not required to contain a decimal point.

   Some valid real constants are: 0.0, 0., 1., -20.05, 2.E2, .000088, 4.0E-02, 4.E02.
7. Comments are enclosed in double dollar signs as described above.

8. Special symbols known to WISSIM are listed below:

**	-
,	*
=	/
(	+
)	.
	$

Commas and spaces are treated identically by WISSIM except in lists of expressions in SET commands, subscripted variables, and function calls. A space or comma is required where a space is shown in the following formats, but the space may be replaced by any number of spaces and/or commas. The description of the SET command contains the special requirements and restrictions on the use of commas in that command.

The WISSIM commands contain many clauses or words that are optional or repeatable. The following conventions will be used to describe the format of each command.

1. Information in square brackets with a superscripted asterisk (*) is optional. If present, it may be repeated any number of times. For example, if the format is

```
AT DELETE [integer]*
```

then

```
AT DELETE
AT DELETE 1
AT DELETE 2 4 6 8
```

are all valid.

2. Information in square brackets with a plus sign (+) as a superscript must be present. However, one or more occurrences are possible.

```
DUMP [var]+
```

means that

```
DUMP
```

is not valid, but

```
DUMP A
DUMP A,B,C
```

are both valid.

3. Information in square brackets with an integer (say n) superscript must be present, but n is the maximum number of occurences that are allowed.
   For example,
   GRAPH [var]^2
   means that
   GRAPH A and
   GRAPH B,A
   are both valid, but
   GRAPH and
   GRAPH A,B,C
   are invalid.
4. Information in square brackets with no superscript is optional.
   SIMULATE [FROM] time
   means that either
   SIMULATE 1970 or
   SIMULATE FROM 1970
   is correct.

The commands are described in alphabetical order in the next section. Table IV-1 is a list of commands arranged by function, with a short description of each.

The descriptions that follow contain many sample commands. All of the commands will be based on variables for an imaginary model that is described in Appendix B. The variables are in two common blocks called ECMN and XCMN. In each common block, the variables are divided into two classes:

1) those that are saved whenever the common block is saved, and
2) those that are saved only if /ALL is added to the common block name.

The following variables are defined:

Common Block	ECMN	XCMN
Class 1 Variables	ENERGY(4,5)	TOTEM(4) SEM(5,4) EM(4,5,4)
Class 2 Variables		EFAC(4,4) CFAC(4,5,4)

A description of the model, definition of the variables, and a sample simulation may be found in Appendix B.

COMMAND	PURPOSE
Control Commands	
SIMULATE	Execute a simulation run
AT	Give a command to be executed at a specified time or times during simulation runs
PAUSE	Temporarily suspend simulation to allow intervention by the model user
CONTINUE	Resume execution after a PAUSE command
SAVE	Save values of variables for graphing or reporting purposes or to allow the system to be restored to a previous state.
RESET	Restore the system to a state that was previously saved
CALL,WRITE	Call a user-written routine
STOP	Halt execution of WISSIM
Input Commands	
SET	Change the value of a model variable, a WISSIM variable, or a temporary variable
DEFINE	Create a temporary variable
Output Commands	
DUMP	Print the current values of model variables, WISSIM system variables, or temporary variables
SREPORT	Print the values of selected variables during a simulation run
NREPORT	Print the values of selected variables from one or more simulation runs after the simulations are completed
GRAPH	Receive CALCOMP, printer, or graphics terminal graphs of variables from one or more runs
LABEL	Give a variable or a variable from a particular run a different name for labels on graphs and reports
XAXIS	Specify that some variable other than time is to be used as the X-axis on future graph commands

Table IV-1: Summary of WISSIM Commands

B. Command Descriptions

1. *AT*

```
1) AT [time] [POLICY(n) policy] [command]+
2) AT [time] [POLICY(n) policy] DELETE [command nbr]*
```

Form 1 of the AT command is used to enter the SET, DUMP, CALL, WRITE, and PAUSE commands that are to be executed. As each command is entered, a unique integer called the command nbr is assigned and printed by WISSIM. Form 2 of the AT command is used to delete AT commands.

Parameter	Description
time	A real number, integer, or rational number
n	An integer $1 \leq n \leq 8$. There are eight policy variables, IZZPOL(1) - IZZPOL(8). n indicates which one should be checked.
policy	A one to six character identifier, a letter followed by one to five letters or numbers
command	A valid SET, DUMP, CALL, WRITE, or PAUSE command
command nbr	The number of a previously entered AT command.

Form 1 of the AT command enters commands into a table that will be executed during following simulations. When each command is entered, a unique number is assigned and printed. That number may be used to delete the command.

At time IZZCUR in a simulation, the following algorithm is used to determine whether an AT command should be executed.

1) If time is omitted, go to 3.
2) If time is not equal to* IZZCUR, go to 6.
3) If policy is omitted, go to 5.
4) If IZZPOL(n) is not equal to policy, go to 6.
5) Execute the AT command.
6) Return.

*It should be noted that in the version of WISSIM on the UNIVAC 1110, two real numbers are considered equal if they agree to approximately seven significant digits.

AT commands that are no longer needed may be deleted with form 2 of the AT command. AT DELETE eliminates all AT commands and temporary variables. Commands may be deleted by time and/or policy or by command numbers. If by command number, the time and policy clauses should be omitted and only the commands actually listed will be deleted. Then the following algorithm is used to decide which commands to delete. Let ctime, cn, and cpolicy be time, n, and policy from a stored command. Then

1) If time is omitted, go to 3.
2) If time is not equal to ctime, go to 7.
3) If policy clause is omitted, go to 6.
4) If n is not equal to cn, go to 7.
5) If policy(n) is not equal to cpolicy(cn), go to 7.
6) Delete the command.
7) Return.

EXAMPLES

```
AT SET CFAC(2,2,2) = CFAC(2,2,2)*.01+CFAC(2,2,2)
```

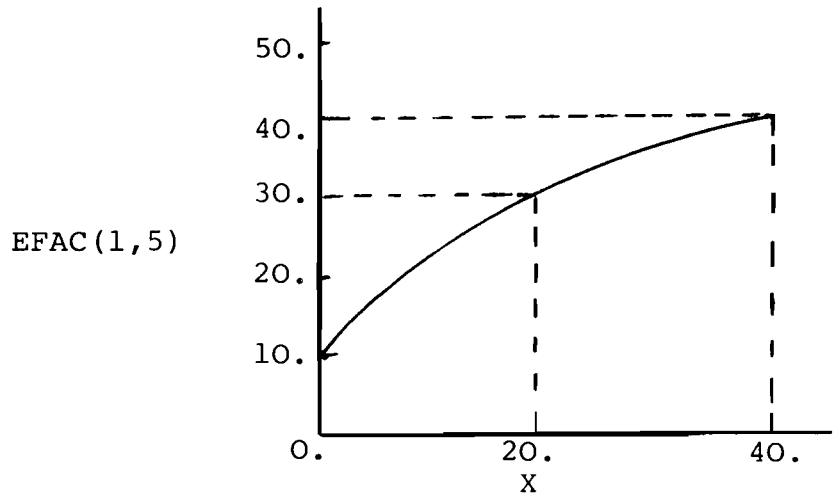
An AT command with no time is executed at every time period, so this command has the effect of increasing CFAC(2,2,2) by 1% per time period. The time period is specified in the SIMULATE command and is called IZZINC at execution time. Therefore the command may be written so that the increase is 1% per year no matter what time increment is specified for the specific simulation run e.g. 2 years, 5 years, etc..

```
AT SET CFAC(2,2,2) = CFAC(2,2,2)*1.01**IZZINC
AT 1980, POLICY(2) LOW SET CFAC(2,2,2)=.50
```

This example introduces the policy. If IZZPOL(2) is set to 'LOW' (SET IZZPOL(2) = 'LOW') prior to a simulation, then at time 1980, CFAF(2,2,2) will be set to 0.50.

```
AT SET EFAC(1,5) = SATHR(X,10.,50.,20.)
```

SATHR is a WISSIM function that defines a saturation curve. The base value is 10.; the saturation value, 50.; and the half time, 20. Thus a curve is defined:



where the value of the function SATHR may be found on the Y-axis given a value for X.

```
AT 1985 SET EFAC(1,1,1)=4 DUMP EFAC
```

More than one command may be entered without repeating the year. AT 1985 EFAC(1,1,1) will be set to 4 and the values of the entire EFAC array will be printed.

```
AT DUMP TOTEM(1)
```

DUMP TOTEM(1) will be executed at every time period.

```
AT 1985. SET CFAC(1,1,1) = .20, $
SET CFAC(1,1,2) = .15
```

At time 1985 (i.e., IZZCUR = 1985) both SET commands will be executed. Note, however, that a bad choice of a simulation interval will mean that IZZCUR is never 1985 and the commands will never be executed. For example, if the SIMULATE command is

```
SIMULATE 1970,1990,2
```

IZZCUR will have values 1970., 1972., 1974.,...,1990. and the AT command will not be executed.

AT POLICY(9) GREEN CALL 6 The command CALL 6 will execute at every time period if IZZPOL(8) has the value 'GREEN'.

AT DELETE All AT commands and any temporary variables are deleted.

AT DELETE 6,10,12 AT commands 6, 10, and 12 will be deleted.

AT 1975 POLICY(6) LOW DELETE Any AT commands with time = 1975, n = 6, and policy = LOW will be deleted.

AT 20. DELETE All AT commands with time = 20. will be deleted.

2a. *CALL*

2b. *WRITE*

CALL integer WRITE integer

CALL and WRITE are used to call special user-written subroutines.

Parameter	Description
integer	An integer that will be passed to the subroutine as a parameter.

CALL uses WISSIM to call a user-writen subroutine named UCALL; WRITE, a user-written subroutine named UWRITE.

EXAMPLES:

```
CALL 7
WRITE 10
```

3. *CONTINUE*

CONTINUE

The CONTINUE command is used to continue a simulation after a PAUSE command has been executed. The command is invalid at any other time.

The CONTINUE command has no parameters. Examples are under the PAUSE command.

4. *DEFINE*

DEFINE tvar = arithex

The DEFINE command is used to create a temporary variable.

Parameter	Description
tvar	A temporary variable name, a dollar sign ($) followed by one to five letters or numbers.
arithex	An arithmetic expression that follows the usual rules for FORTRAN expressions except that FORTRAN defined functions are not allowed. WISSIM functions described in Appendix C may be used.

DEFINE creates a temporary variable that exists during the current execution of WISSIM and may be used in subsequent SET, GRAPH, DUMP, SREPORT, and NREPORT commands. The value of DEFINE'd variables is calculated from arithex at every time period during subsequent simulations. The DEFINE'd variable will not have a value from simulations previous to the DEFINE command, even if temporary variables were saved.

EXAMPLES:

DEFINE $RATIO = SEM(3,2)/TOTEM(2) A temporary variable $RATIO whose value will be calculated at every period as SEM(3,2)/TOTEM(2). If $RATIO is to have an initial value, it must explicitly be given one with a SET command. From the variable descriptions in the appendix, SEM(3,2) is the oxides of sulfur emissions from the industrial sector and TOTEM(2) is the total oxides of sulfur emissions from all sectors. Therefore $RATIO is the fraction of the total SO_x emissions contributed by the industrial sector.

DEFINE $ACCUM = TOTEM(2) + $ACCUM This command defines a temporary variable that accumulates the TOTEM(2)the total SO_x emissions from each simulation interval.

DEFINE $SOXKG = TOTEM(2) / 2.2 This command converts the pounds of SO_x emissions to kilograms.

5. *DUMP*

```
1)  DUMP FROM begin TO end
2)  DUMP [dvar]+
```

The DUMP command is used to print the current values of model variables, WISSIM system variables, or temporary variables.

Parameter	Description
begin	May be a model variable, a WISSIM system variable, a temporary variable, or the word BEGIN. BEGIN means to dump from the first word in the first model common block.
end	May be a model variable, a system variable, a temporary variable, or the word END. If end is a model or system variable, it must occur after begin in the definition of the common blocks. The system variable common block, SYSVAR, is the first common block; the temporary variable common block, TMPVAR, is the last. END means to dump the last variable in the last model common block.
dvar	May be a model variable, a system variable, a temporary variable, an unsubscripted model or system array name, a model common block, SYSVAR, TMPVAR, or ALL.

Form 1 of the DUMP command will dump the variables specified by begin, end, and all variables between.

Form 2 will dump all variables listed. If dvar is an unsubscripted array name, the entire array will be printed. If it is a model common block name, the entire common block will be printed. SYSVAR will dump the system variable common block; TMPVAR, the

temporary variable common block; and ALL, all model common blocks and, if there are any DEFINE'd variables, TMPVAR.

The system variable IZZPRE contains the number of digits that are to be printed for each variable. IZZPRE may be set to an integer value n, $3 \leq n \leq 7$. It has a default value of 3.

EXAMPLES:

```
DUMP FROM EM(1,1,1) TO EM(4,5,1)      The values of the
```

indicated variables will be printed.

Note that in both FORTRAN and WISSIM the leftmost subscript varies fastest, so this command prints the 20 values of EM(i,j,k) for values of i from 1 to 4 and j from 1 to 5 with k = 1.

```
DEFINE $TOTAL = ENERGY(1,1) + ENERGY(2,1) + ENERGY(3,1)
SET $TOTAL = ENERGY(1,1) + ENERGY(2,1) + ENERGY(3,1)
DUMP ALL      All model variables and temporary variables
```

($TOTAL) are printed.

```
DUMP CFAC, ENERGY(4,5), SYSVAR,$TOTAL     The values of
```

the entire array CFAC, of ENERGY(4,5), of the entire common block SYSVAR, and of the temporary variable $TOTAL are printed.

```
SET IZZPRE = 6
DUMP EFAC        The values of the array of EFAC are
```

printed to six significant digits.

6. *GRAPH*

GRAPH [[FROM runid][var [NORMALIZED][WITH 'c']]+]+

The GRAPH command is used to produce graphical output from previously run simulations.

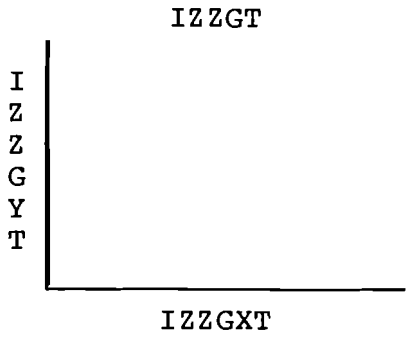
Parameter	Description
runid	A one to six character identifier (one letter followed by one to five letters or numbers). runid must have been used as the runid on a previous SIMULATE command.
var	A model variable or temporary variable. The variable must have been saved during the simulation run specified by runid.
c	Any printable character. It will be used to identify the lines on the graph.

Normally time is used as the x-axis variable and the variables listed in the GRAPH command are plotted against time on the y-axis. However any variable from any simulation run may be used as the x-axis by entering an XAXIS command prior to entering the GRAPH command.

No more than seven variables may be listed in any one GRAPH command. If the "FROM runid" clause is omitted, the most recent simulation with saved data is assumed. If more than one runid is specified, the begin time, end time, and time increment must agree for all runid's. If NORMALIZED is present, the values of the variable divided by its initial value is plotted. The word NORMALIZED is equivalent to and may be replaced by a slash (/) in the WISSIM commands.

Titles of up to 30 characters are entered by setting the literal arrays IZZGT, IZZGXT, and IZZGYT to the desired title. The title must end with a double dollar sign ($$) which is not printed as part of the title. The placement of the titles is

shown in the diagram below.



An automatic scaling routine is included that will scale the x- and y-axes to the maximum and minimum of the data. If the graph should have special scaling, the system variable IZZGSC may be set as follows:

IZZGSC	Scaling Option
0	No automatic scaling. Use values of IZZXMX, IZZYMX, IZZXMN, IZZYMN as the minima and maxima on the graph. One reason for specifying scaling option zero is to produce several graphs with the same scale for easier comparison.
1	(Default value). Zero is the y-axis minimum. Other scaling is automatic.
2	All automatic scaling.

If IZZGSC is set to zero, then the scaling must also be explicilty given by SET commands. IZZXMX and IZZXMN are the x-axis maximum and minimum respectively; IZZYMX and IZZYMN, the y-axis maximum and minimum. IZZXSC is an integer variable that contains the number of scaling intervals on the x-axis (default 5), and IZZYSC similarly for the y-axis (default 7). IZZSKP (default 5) specifies the frequency with which the symbols should be drawn on the lines for each variable. Page width and height for the graph in inches can also be specifically set. IZZPGW contains the page width in inches (default 6") and IZZPGH, the page height (default 9").

A legend, i.e. the name of each variable, its run identifier and the symbol used on the graph, is printed at the bottom of each graph. The legend may be suppressed by setting the system

variable IZZGLD (default value of 1) to zero.

The first example graphs TOTEM(2) from two simulation runs on the same graph. No titles are specified, scaling will be automatic, the lines for the two variables will be marked with "H" and "L", the plot will be 6" x 9", and a legend will be printed. This is the easiest way of getting a graph and requires only one statement after the simulations have been run.

```
SAVE AS INIT ALL/ALL
SAVE ECMN
SIMULATE H FROM 1970 to 1980
RESET INIT
SET CFAC (5,1,2) = .95
SIMULATE L FROM 1970 to 1980
GRAPH FROM H TOTEM(2) WITH 'H' FROM L TOTEM(2) WITH 'L'
```

The second example illustrates more of the special options associated with the graph command. A simulation is run, and the titles are specified.

```
SAVE XCMN/ALL
SIMULATE R1 FROM 1970 TO 1980
SET IZZGT(1) = 'SOX VS. SOX CONTROL FACTOR$$'
SET IZZGXT(1) = 'SOX CONTROL FACTOR$$'
SET IZZGYT(1) = 'SOX EMISSIONS (KG)$$'
```

The x-axis is to be CFAC(1,1,1).

```
XAXIS CFAC(1,1,1)
```

Scaling is to be automatic except that the y-axis minimum is zero. Page width is 4" by 5"; the y-axis scale is to have only five scaling intervals, and the legend is suppressed.

```
SET IZZGSC = 1
SET IZZPGW = 4.
SET IZZPGH = 5.
SET IZZYSC = 5
SET IZZGLD = 0
```

To graph it, R1 is assumed since it was the last simulation run and the command is:

```
GRAPH TOTEM(1)
```

7. *LABEL*

LABEL [[FROM runid][var = label]$^{+}$]$^{+}$

The LABEL command is used to rename a variable or a variable from a particular run for labels on graphs and reports.

Parameter	Description
runid	A one to six character identifier (one letter followed by one to five letters or numbers). runid must have been used as the runid on a previous SIMULATE command.
var	A model variable, temporary variable, or system variable. If runid was specified, the variable should have been saved during the simulation run.
label	A literal (enclosed in single quotation marks) whose length is 1 to 16.

If no label has been specified for a variable, the variable name is used as the column heading on reports (both SREPORT and NREPORT) and on the legend of graphs. If the runid is omitted, the label is used for all runid's. For reports, the labels are broken into two lines of eight characters each.

A LABEL command with no parameters erases all current labels.

EXAMPLES:

```
LABEL TOTEM(2) = 'SOX', TOTEM(3) = 'NOX'
```

All reports and graphs of TOTEM(2) and TOTEM(3) will be labelled SOX and NOX, respectively.

```
LABEL FROM RUN1 TOTEM(2) = 'SOX RUN1' TOTEM(3) = 'NOX RUN1' $
FROM RUN2 TOTEM(2) = 'SOX RUN2' TOTEM(3) = 'NOX RUN2'
```

TOTEM(2) and TOTEM(3) from RUN1 and RUN2 will be labelled as shown. Extra spaces can be inserted in the labels to make the variable name print on the first line of reports and the runid on the second.

8. *NREPORT*

NREPORT [[FROM runid][var [NORMALIZED]]+]+

The NREPORT command is used to print values of variables from previous simulation runs.

Parameter	Description
runid	A one to six character identifier (one letter followed by one to six letters or numbers). runid must have been used as the runid on a previous SIMULATE command.
var	A model variable, temporary variable, or system variable that was saved during the simulation run referred to by runid.

A maximum of seven variables may be listed for a demand run; twelve for a batch run. The LABEL command may be used to rename a variable for reporting purposes.

Several system variables affect the format of the report. IZZRFQ determines the frequency with which data is reported and must be an integer, n, greater than or equal to one. Then the values of the variable listed are printed every n periods. The default value is one. IZZPRE is an integer that specifies the number of digits to print in the report and has an allowable range of $3 \leq$ IZZPRE ≤ 7. The default is 3. IZZRT is a string array with a length of 72 characters containing the title of the report. The title must end with double dollar signs ($$), which will not be printed as part of the title. In addition the title may contain the characters dollar sign, slash, dollar sign ($/$), also not printed which will indicate that the title is to be divided into two lines at that point.

If the "FROM runid" clause is omitted, the most recent simulation with saved data is assumed. If more than one runid is specified, the begin time, end time, and time increment must agree for all runid's. If NORMALIZED is specified, the value

of that variable at every time period is divided by its beginning time value before being printed. For example, if ENERGY(1,1) = 10 at time 10 and is 20 at time 11, then the report would show 1 at time 10 and 2 at time 11. The word NORMALIZED is equivalent to and may be replaced by a slash (/).

The following example reports the values of TOTEM(1) and TOTEM(2) from every time period in RUN2. IZZSFQ is described with the SAVE command. It specifies the frequency with which values are to be saved in the same way that IZZRFQ determines report frequency. These commands are assumed to have been entered before the examples given below:

```
SAVE XCMN
SIMULATE RUN1 1,20,1
SIMULATE RUN2 1,30,1
SIMULATE RUN3 1,20,1
SET IZZSFQ = 2
SIMULATE RUN4 1,20,.5
```

Correct and incorrect uses of NREPORT from the simulations above are given below:

```
NREPORT FROM RUN2 TOTEM(1), TOTEM(2)
NREPORT FROM RUN1 TOTEM(1) TOTEM(2)FROM RUN3 TOTEM(1) $
TOTEM(2)
```

The values of TOTEM(1) and TOTEM(2) from every time period in both RUN1 and RUN3 are printed.

```
NREPORT FROM RUN1 TOTEM(1) FROM RUN4 TOTEM(2)
```

Since IZZSFQ = 2 and the time increment is .5 for RUN4, the values of IZZCUR on the save file are 1.,2.,...,20, which is identical to RUN1 and thus, this is a valid command.

```
NREPORT FROM RUN3 SEM(1,1) SEM(1,1) NORMALIZED
```

The value of SEM(1,1) and the value of SEM(1,1) divided by the value of SEM(1,1) at time 1 for each time period during RUN3 are printed.

```
NREPORT FROM RUN3 SEM(1,1) SEM(1,1)/
```

This statement is the same as the previous command.

```
NREPORT FROM RUN1 EFAC(1,1)
```

An invalid command because class 2 variables from common block XCMN were not saved during RUN1 (see page 24).

NREPORT FROM RUN4 ENERGY(1,1) Invalid because common block ECMN was not saved during RUN4.

9. *PAUSE*

PAUSE

PAUSE is used to temporarily halt a simulation run. It is only valid as a parameter of an AT command and is useful for interactive work at a computer terminal.

The PAUSE command has no parameters.

When a PAUSE command is executed a message is printed, PAUSE AT TIME n, where n is the current value of the system variable IZZCUR. At that time any of the following WISSIM commands may be entered:

AT	
CALL	SREPORT
DUMP	STOP
LABEL	WRITE
SET	XAXIS

When simulation should be started again, a CONTINUE command is entered.

EXAMPLE:

```
AT   1980 'PAUSE'
SIMULATE 1975,2000,5
```

The AT command is entered. WISSIM assigns and prints a command number for it. Simulation starts when the SIMULATE command is entered and pauses at time 1980. At that time any of the commands listed above may be entered, e.g.,

```
DUMP CFAC
SET CFAC(1,5,2) = .95
SREPORT SEM(1,1) SEM(1,2) SEM(1,3) SEM(1,4)
AT 1990 'PAUSE'
CONTINUE
```

The new commands are executed and simulation resumes with the entry of the CONTINUE command. Because of the SREPORT command, the values of SEM(1,1), SEM(1,2), SEM(1,3), and SEM(1,4) are printed at each time period. Execution is suspended again at time 1990 when the second SET command is executed. More commands can then be entered.

```
SET CFAC(1,5,2) = .99
CONTINUE
```

The execution of the CONTINUE command causes simulation to continue to time 2000, the endtime on the SIMULATE command.

10. *RESET*

RESET [[TO] runid] [[FROM] time]

The reset command is used to restore common blocks to a previously saved state.

Parameter	Description
runid	A one to six character identifier (one letter followed by zero to five letters or numbers). runid must have been used previously as the runid in a SAVE or SIMULATE command. If omitted, the runid from the first SAVE or SIMULATE command is used by default.
time	A real number, integer, or rational number to specify the time from runid that the common blocks are to be restored to. If omitted, the first time period saved is assumed.

Only the variables in the common blocks that were saved for a particular runid can be restored to their original values.

EXAMPLES:

```
SAVE AS INIT ALL/ALL
SAVE XCMN,ECMN,TMPVAR
SIMULATE RUN1,1970,2000,1      All class 1 variables
```
are saved at every time increment.
```
SET CFAC (ALL,ALL,ALL) = 80*0
SET CFAC(1,5,1-4) = 4*.5
SAVE AS LOWCON XCMN/ALL
SET IZZSFQ = 5
SIMULATE RUN2 1970,2000,1      All class 1 variables
```
are saved at every 5th time increment.
```
RESET LOWCON
```
Restores all values from common block XCMN to the values they

had when the "SAVE AS LOWCON" command was issued. Another example:

RESET TO RUN1 FROM 1980

Restores the system to time 1980 during RUN1. Class 1 variables from XCMN, ECMN, and TMPVAR are restored but class 2 variables are unaffected since the SAVE command issued just prior to the SIMULATE command for RUN1 did not specify "ALL" on any of the common blocks.

11. *SAVE*

```
1) SAVE [AS runid] ALL[/ALL]
2) SAVE [AS runid] [cb[/ALL]]+
```

SAVE is used to save the values of variables on a temporary SAVE file.

Parameter	Description
runid	A one to six character identifier (one letter followed by zero to five letters or numbers). runid must not have been used on a previous SAVE or SIMULATE command.
cb	The name of one of the user-defined model common blocks or the WISSIM temporary variable common block TMPVAR.

Form 1 is used to save all of the user-defined common blocks; form 2 saves only the ones that are explicitly named in the command.

WISSIM variables are divided into two classes: 1) those generally saved for reports and graphs, and 2) those that are not usually necessary to save. The costs incurred to save the values of all variables usually are large enough to make it desirable to save only class 1 variables. However, at times it may be convenient to save all of a model's variables, e.g., to save initial values. ALL or a common block name that is not followed by a /ALL indicates that only class 1 variables in that common block are saved; /ALL means both class 1 and class 2 variables are saved. The common blocks used as examples are divided into class 1 and 2 variables (page 24).

The presence of the AS runid clause indicates that the values will be saved immediately under the name given by runid. This type of SAVE command, called an immediate SAVE, does not affect the number or names of common blocks that are to be saved in future simulations.

A SAVE command with no AS runid clause sets a system variable to indicate that the common blocks specified should be saved during future simulations. Only one SAVE command with no "AS runid" clause is in effect at any one time. Each new command overrides the previous one. Once set, the save can be discontinued by setting the WISSIM system variable IZZSAV to 0. IZZSFQ is a positive integer that controls the save frequency. If IZZSFQ = n, the values are saved at every nth time period. The default value of IZZSFQ is 1,i.e., the values are saved every time period. IZZSAV and IZZSFQ must not be changed during a simulation.

EXAMPLES:

SAVE AS INIT ALL/ALL The current values of all variables in all common blocks are saved under the name INIT. This form of the command is very useful for saving the initial values of a model.

SAVE AS ACASE1 XCMN This command saves the current values of the class 1 variables (the TOTEM, SEM, and EM arrays) in common block XCMN under the identifier ACASE1.

SAVE ALL Class 1 variables in both common blocks, ECMN and XCMN, and the temporary variables common block TMPVAR will be saved in future simulations under the identifier assigned on the SIMULATE command. Nothing is saved immediately.

SAVE XCMN,ECMN If no temporary variables have been defined, this command is identical to SAVE ALL.

SAVE TMPVAR Temporary variables will be saved in future simulations.

SAVE XCMN/ALL

SET IZZSFQ = 5 All variables in common block XCMN will be saved every fifth time period in following simulations. The value of IZZSFQ at the time of the simulation determines the save frequency. Entering another SET IZZSFQ command before any simulation will change the save frequency.

SET IZZSAV = 0

SIMULATE 1970,2000,1 No values are saved because IZZSAV was set equal to 0.

12. *SET*

SET variable=explist

SET changes the values of variables.

Parameter	Description
variable	The variable may be either 1) a temporary variable created by a previously issued DEFINE command, or 2) a FORTRAN variable that has been defined in one of the COMMON blocks that are known to WISSIM or an unprotected system variable. In this case the variable may be a) an undimensioned variable, b) an array name, or c) a subscripted variable. Valid subscript entries are: n — n is an integer less than or equal to the dimension specified in the COMMON statement. n-m — n and m are integers less than or equal to the dimension specified in the COMMON statement, n < m. Array elements n through m are set to the values in the expression list. ALL — ALL is identical to 1-n where n is the dimension of the subscript in the COMMON statement.
explist	A list of arithmetic expressions, separated by commas. An expression may be duplicated an arbitrary number of times by preceding it with n @, where n is a positive integer. The arithmetic

ctd...

expressions follow the usual rules for FORTRAN expressions except that FORTRAN defined functions are not allowed. WISSIM functions described in Appendix C may be used.

The SET command assigns the value(s) calculated from the list of arithmetic expressions to the variable or array elements to the left of the equal sign. The number of array elements must be equal to the number of values calculated on the right. In the case of literals, the entire literal must fit into the array. Literals are not allowed when the command is a parameter of a AT command.

Mixed mode expressions are allowed in which case integer values are converted to real numbers before arithmetic operations are done. Commas are required with variables that have more than one subscript. They are also mandatory to separate parameters in a function call or to separate expressions in explist. No extraneous commas should appear in the SET command.

EXAMPLES:

```
SET ENERGY(4,5) = 0.
```

The value of the ENERGY(4,5) is set to 0.

```
SET CFAC(1,5,ALL) = 2@.5,1.5*CFAC(1,5,1),SATHR(CFAC $
(1,5,1),0.,1.,.25)
```

CFAC is dimensioned CFAC(4,5,4). After this command is executed, the values are:

I	CFAC(1,5,I)
1	.5
2	.5
3	.75
4	.75

Since the values are set from left to right, CFAC(1,5,1) has the value .5 when used in the calculations of CFAC(1,5,3) and CFAC(1,5,4). SATHR is a function that defines the saturation curve and is described in Appendix C.

SET EFAC(1,1-2) = 2@3.1 The values of EFAC(1,1) and EFAC(1,2) are set to 3.1.

AT SET CFAC(1,5,2) = SATHR(IZZCUR-1970,.5,.99,10,) Here is an example of an AT command with no time and no policy specified, i.e., the command will be executed at every time period during every simulation, that sets the value of CFAC(1,5,2) to the value of the function SATHR at every time increment. IZZCUR is a system variable containing the current simulation time; SATHR is an exponential function that can be used for saturation curves.

SET IZZGYT(1) = 'PARTICULATE,EMISSIONS $$' IZZGYT is a subscripted system variable used as a title. This set command gives it a value.

13. *SIMULATE*

SIMULATE [runid] [FROM] begtime [TO] endtime [[BY] timeinc]

Parameter	Description	
runid	An identifier consisting of a letter followed by 0 to 5 letters or numbers. The runid is the name under which values from the simulation will be saved. Runid must be unique and must not be used to identify a second SAVE or SIMULATE command. The runid field must be present if the system variable IZZSAV has a value other than zero, i.e., if a SAVE other than an immediate save has been issued. If IZZSAV has the value zero (no SAVE issued or a SET command of zero), then the runid is ignored.	
begtime	The beginning time of simulation.	May be real numbers, integers, or rational numbers of the form integer/integer. If timeinc > 0 then endtime > begtime. If timeinc < 0, then endtime < begtime.
endtime	The ending time of simulation.	
timeinc	The time increment - the default value is 1 and zero is invalid.	

The real system variables IZZBEG, IZZEND, and IZZINC have the values of begtime, endtime, and timeinc, respectively. During the simulation IZZCUR, also a real variable, is the current time of simulation.

During the execution of a simulate command, WISSIM performs the following:

1. Sets IZZCUR to begtime.
2. Executes any applicable AT commands. The values of any temporary variables are also calculated at this point.
3. Calls the model.
4. If a SAVE command was issued, saves the data.
5. Writes a line of the report, if requested.
6. Calculates a new current time (IZZCUR) and checks for end. If end is not found, WISSIM goes to 2. If end is found, the simulation is finished.

EXAMPLES:

SIMULATE FROM 1970 to 2000 by 5 Since no runid is present, no SAVE command should be in effect. The model will be called seven times with IZZCUR = 1970., 1975.,1980.,...,2000.

SIMULATE 10,20,1/4 Again no runid is present. Note that commas and/or spaces can replace the single space shown in the format. The model will be called 41 times with IZZCUR = 10., 10.25, 10.5, 10.75,...,20.

SIMULATE RUN 1 FROM 1970. to 1980. The runid is RUN1. A SAVE command should be in effect before this SIMULATE is entered; otherwise, specifying RUN1 has no useful function. RUN1 will be used as the runid in any future RESET, GRAPH, or NREPORT commands that will use values from this simulation run. Since the timeinc is omitted, it defaults to 1.; the simulation model is called eleven times with IZZCUR = 1970., 1971.,...,1980.

SIMULATE HIGH FROM 10 to 33/2 BY .5 This command illustrates that integer, real, and rational numbers may be used, but WISSIM converts them all to real numbers. The runid is HIGH. The model is called fourteen times with IZZCUR = 10., 10.5, 11., ...,16.5

14. *SREPORT*

1) SREPORT [var][13]
2) SREPORT

SREPORT is used to print the values of selected variables during future simulations.

Parameter	Description
var	A model variable, temporary variable, or WISSIM system variable.

The variables are printed in columns. The variable name is used as the column heading, unless changed with a LABEL command.

Only one SREPORT command is in effect at any one time. Each new SREPORT command overrides the previous one. The second form of SREPORT, i.e., with no parameters, indicates that no report is to be printed in future simulations.

EXAMPLES:

```
SREPORT ENERGY(4,1),ENERGY(4,2)
```

The values of ENERGY(4,1) and ENERGY(4,2) are reported at every time period during future simulations.

```
SET IZZRFQ = 5
SET IZZPRE = 4
SET IZZRT(1) = 'ENERGY FROM WOOD$$'      Now ENERGY(4,1)
```

and ENERGY(4,2) will be reported at every fifth time period with four digit precision during future simulations. The title 'ENERGY FROM WOOD' will be printed at the beginning of each simulation.

15. *XAXIS*

```
1) XAXIS [FROM runid] var [NORMALIZED]
2) XAXIS
```

The XAXIS command is used to specify the x-axis variable on subsequent GRAPH commands. Form 2) of the command resets x-axis to time.

Parameter	Description
runid	A one to six character identifier (one letter followed by one to five letters or numbers). runid must have been used as the runid on a previous SIMULATE command.
var	A model variable or temporary variable. The variable must have been saved during the simulation run specified by runid.

The x-axis command precedes the GRAPH commands to which it applies. If the "FROM runid" clause is omitted, the runid of the first variable in the GRAPH command will be assumed. NORMALIZED, as in the NREPORT and GRAPH commands, specifies that the variable should be divided by its initial value before it is graphed. The word NORMALIZED is equivalent to and may be replaced by a slash(/).

V. SUMMARY AND CONCLUSIONS

WISSIM has proved to be a convenient tool in the development and use of the models that compose the Wisconsin Regional Energy Model. At the development stage, it is a simple process to modify the model and relink it with WISSIM. Since the FORTRAN program and WISSIM use the same description of the model variables, i.e., the FORTRAN COMMON statements, it is not necessary to redefine the variables to WISSIM. The DUMP, AT DUMP, and SREPORT commands are also very useful during the model debugging and check-out stages. For demonstrations and workshops, plots on the graphics terminal are most useful and the relatively free format of the commands is very convenient. For researchers reporting on the results of their research, the CALCOMP graphs with scaling options and titles and the capability of looking at more than one simulation at a time proved to be very helpful. The use of multiple common blocks allows models such as the WISE submodels to run either independently or linked without recompiling the model.

There are many possibilities for extensions to WISSIM. The graphs and reports of course can be further refined; FORTRAN functions can be included in the SET command with little additional work. Bar graphs, graphs of initial values, and three dimensional graphs could also be added since the mechanism for retrieving data from previous simulations already exists. It may also be desirable for the DEFINE command to define an array of variables rather than just an unsubscripted variable. A simplified miniature version of WISSIM would also be desirable for mini-computers such as IIASA's PDP-11.

The general approach of WISSIM has been very practical rather than theoretical. The fewest constraints possible are placed upon the model, even to the extent of not really distinguishing between state variables, rates, parameters, and other variables.

APPENDIX A

SYSTEM VARIABLES

System variables are defined in common block SYSVAR and are of two types: protected and unprotected. Both types may be printed with DUMP or REPORT commands, but the value of a protected variable may not be changed with a SET command.

Variable	Type	Default Value	Possible Values	Meaning
			P R O T E C T E D	
IZZSIM	Integer	-		Simulation status:
			0	Simulation is in progress.
			1	Simulation is not in progress.
IZZBEG	Real		Any real number	The beginning time of simulation.
IZZEND	Real	-	Any real number	The ending time of simulation.
IZZINC	Real	1.	Any real number	The time increment for simulation.
IZZCUR	Real	-	Any real number	The current time of simulation.
IZZNVR	Integer	-	Any positive integer	The number of variables (including system variables) stored in the variable name table.
IZZRTP	Integer	-		Run Type:
			0	Demand
			1	Batch
			U N P R O T E C T E D	
IZZIUN	Integer	5	Any valid FORTRAN unit number	The logical unit number for system input.
IZZOUN	Integer	6	Any valid FORTRAN unit number	The logical unit number for system output.
IZZSAV	Integer	0		The SAVE status:
			0	Do not save values during simulation.
			1	Save values during simulation.

APPENDIX A (CONTINUED)

Variable	Type	Default Value	Possible Values	Meaning
U N P R O T E C T E D (Cont'd)				
IZZSFQ	Integer	1	Any positive integer	Save frequency during simulations. If IZZSFQ = n, then the save is performed every nth time period.
IZZNRP	Integer	0	0≤IZZNRP≤13	The number of variables to be reported during simulations. (The value of this variable is changed by the SREPORT command. It may be decreased with a SET command but should never be increased since WISSIM will not know what variables to report.)
IZZRFQ	Integer	1	Any positive integer	Frequency that report is to be printed during simulations. If IZZRFQ = n, then the values are printed every nth period.
IZZPRE	Integer	3	3≤IZZPRE≤7	The number of digits printed for real numbers in REPORT's and DUMP's.
IZZRT(12)	String	'$$'	•	The report title. The maximum length is 72 characters at Wisconsin; 48 at IIASA. The title must end with a double dollar sign ($$). It may contain a dollar sign, slash, dollar sign ($/$) to indicate the point at which the title is to be split into two lines.
IZZPOL(8)	String	Blank	-	Policies that are used in the AT command. Each policy is an identifier that must be one letter followed by one to five letters or numbers.
IZZECP	Integer	0	0 1	Echo print of input command indicator. Do not echo print. Echo print.
Graphics related - U N P R O T E C T E D				
IZZGRP	Integer	2	1 2 3	Type of graphics. Only type 2 is currently implemented at IIASA. Printer plot. Calcomp. Graphics terminal.
IZZGT(8), IZZGXT(8), IZZGYT(8)	String	'$$'	-	The main, the x-axis, and the y-axis titles respectively for graphs. The maximum length is 48 characters at Wisconsin, 32 at IIASA.

APPENDIX A (CONTINUED)

Variable	Type	Default Value	Possible Values	Meaning
Graphics related - U N P R O T E C T E D (Cont'd)				
IZZLGD	Integer	1	0 1	Legend status on graphs. Do not print legend. Print legend
IZZGSC	Integer	1	0 1 2	Scaling option for graphics. No automatic scaling. Use IZZXMN, etc. as the actual maxima and minima on the the graphs. Zero is the y-axis minimum. Other scaling is automatic. All scaling is automatic.
IZZXMN, IZZXMX, IZZYMN, IZZYMX	Real	-	Any real number	The x- and y-axis minima and maxima for the scale on graphs if IZZGSC option 0 is selected.
IZZXSC, IZZYSC	Integer	5, 7	Any positive integer (not 0!)	The number of intervals on the x- and y-axis scales respectively.
IZZPGW, IZZPGH	Real	6., 9.	Any positive real number	The page width and height in inches respectively for graphs. The numbers specified are the actual lengths of the axes. Additional space is required for titles, scales, and the legend. Care must be taken that the graph fits on the physical medium on which it is to be drawn (i.e., the paper for the CALCOMP).
IZZSKP	Integer	5	Any positive integer.	The frequency with which the sysmbols on "WITH 'c'" clause are to be printed on the graphs. (5 is every fifth value that is provided.)

APPENDIX B

SAMPLE MODEL

To illustrate the use of WISSIM, we will use a very simple hypothetical emissions model for air pollution from the use of fossil fuels. We will assume that inputs include the primary energy use per year (ENERGY) in million BTUs by sector of the economy modelled, (S), and fuel type, (F), where S and F are defined as shown in Tables B-1 and B-2.

Table B-1

SECTOR OF THE ECONOMY	S
Residential	1
Commercial	2
Industrial	3
Transportation	4
Electricity Generation	5

Table B-2

Fuel	F
Coal	1
Oil	2
Natural Gas	3
Wood	4

The energy use may be either a data list or the output of some demand or supply model(s). With each fuel, we will associate emission factors (EFAC) in pounds per million BTU for each of four types of emissions (T), described in Table B-3.

Table B-3

Emission Type	T
Particulates	1
Oxides of Sulfur	2
Oxides of Nitrogen	3
Hydrocarbons	4

Policy input will be by means of parameters called control factors (CFAC) by sector of the economy, fuel type, and emission type that will give the fraction of the emissions that will be caught by pollution control devices. The results to be calculated are the total emissions per year (TOTEM) of each type and the emissions per year broken down by sector of the economy SEM. The equations are very straightforward:

$$EM(F,S,T) = (ENERGY(F,S,T)*EFAC(F,T))*(1. - CFAC(F,S,T))$$

$$SEM(S,T) = \sum_{F} EM(F,S,T)$$

$$TOTEM(T) = \sum_{F}\sum_{S} EM(F,S,T)$$

The model is written so that it may be linked with another model which will provide the energy demand data. In that case, we will want the energy variables to be in a separate common block so that other models will not need to include variables that are only used by this model. The energy common block:

```
C
C       ENERGY(F,S),F=1,4,S=1,5
C                   ENERGY USED BY FUEL AND BY SECTOR OF THE ECONOMY
C
C       F           FUEL
C       1           COAL
C       2           OIL
C       3           NATURAL GAS
C       4           WOOD
C
C       S           SECTOR OF THE ECONOMY
C       1           RESIDENTIAL
C       2           COMMERCIAL
C       3           INDUSTRIAL
C       4           TRANSPORTATION
C       5           ELECTRICITY GENERATION
C
      COMMON/ECMN/ENERGY(4,5)
C
```

Then the emission block can be:

```
C
C         EFAC(F,T),F=1,4,T=1,4
C                   EMISSION FACTORS
C         CFAC(F,S,T),F=1,4,S=1,5,T=1,4
C                   CONTROL FACTORS
C         EM(F,S,T),F=1,4,S=1,5,T=1,4
C                   EMISSIONS
C         SEM(S,T),S=1,5,T=1,4
C                   EMISSIONS BY ECONOMIC SECTOR
C         TOTEM(T),T=1,4
C                   TOTAL EMISSIONS
C
C         T         TYPE OF EMISSION
C         1         PARTICULATES
C         2         SULFUR OXIDES
C         3         NITROGEN OXIDES
C         4         HYDROCARBONS
C
      COMMON/XCMN/TOTEM(4),SEM(5,4),EM(4,5,4)
C
C$$$$$ CLASS 1 VARIABLES (IMPORTANT RESULTS TO BE GRAPHED) ABOVE AND
C         CLASS 2 VARIABLES (PARAMETERS, ETC., THAT WILL PROBABLY NOT
C         BE REPORTED OR GRAPHED) BELOW.
C
      COMMON/XCMN/EFAC(4,4),CFAC(4,5,4)
C
```

We must also provide initialization procedures for the model. Since we are assuming the model is part of a larger model, we need not worry about the energy use figures. They must either be calculated by another model or entered by SET commands. The emission factors will usually remain the same, so we initialize them with DATA statements with the knowledge that they can be overriden by SET commands. The control factors will be initialized to zero, i.e., no control. The emissions will all be calculated and therefore need no initialization. The model:

```
      SUBROUTINE UMODEL(BEG,END,INC,CUR)
C
C
C        DEMONSTRATION MODEL FOR WISSIM — A PEDAGOGICAL AIR
C                  POLLUTION EMISSIONS MODEL.
C
      REAL INC
      INTEGER F,S,T
C
```

```
C
C         ENERGY(F,S),F=1,4,S=1,5
C                   ENERGY USED BY FUEL AND BY SECTOR OF THE ECONOMY
C
C         F         FUEL
C         1         COAL
C         2         OIL
C         3         NATURAL GAS
C         4         WOOD
C
C         S         SECTOR OF THE ECONOMY
C         1         RESIDENTIAL
C         2         COMMERCIAL
C         3         INDUSTRIAL
C         4         TRANSPORTATION
C         5         ELECTRICITY GENERATION
C
      COMMON/ECMN/ENERGY(4,5)
C
C
C         INITIALIZE EMISSION FACTORS AND CONTROL FACTORS
C
C
C         EFAC(F,T),F=1,4,T=1,4
C                   EMISSION FACTORS
C         CFAC(F,S,T),F=1,4,S=1,5,T=1,4
C                   CONTROL FACTORS
C         EM(F,S,T),F=1,4,S=1,5,T=1,4
C                   EMISSIONS
C         SEM(S,T),S=1,5,T=1,4
C                   EMISSIONS BY ECONOMIC SECTOR
C         TOTEM(T),T=1,4
C                   TOTAL EMISSIONS
C
C         T         TYPE OF EMISSION
C         1         PARTICULATES
C         2         SULFUR OXIDES
C         3         NITROGEN OXIDES
C         4         HYDROCARBONS
C
      COMMON/XCMN/TOTEM(4),SEM(5,4),EM(4,5,4)
C
C$$$$$ CLASS 1 VARIABLES (IMPORTANT RESULTS TO BE GRAPHED) ABOVE AND
C         CLASS 2 VARIABLES (PARAMETERS, ETC., THAT WILL PROBABLY NOT
C         BE REPORTED OR GRAPHED) BELOW.
C
      COMMON/XCMN/EFAC(4,4),CFAC(4,5,4)
C
      DATA EFAC/3.90,4.19,1.01,0.10,3.66,1.43,0.52,0.03,
     1    1.17,0.04,0.17,0.35,4.56,2.39,0.48,0.09/
      DATA CFAC/80*0./
C
C         CALCULATE EMISSIONS AND TOTALS
C
```

```
      DO 100 T=1,4
      TOTEM(T)=0.
      DO 75 S=1,5
      SEM(S,T)=0.
      DO 50 F=1,4
      EM(F,S,T)=ENERGY(F,S)*EFAC(F,T)*(1.-CFAC(F,S,T))
 50   SEM(S,T)=SEM(S,T)+EM(F,S,T)
 75   TOTEM(T)=TOTEM(T)+SEM(S,T)
 100  CONTINUE
      RETURN
C
C        PROVIDE ENTRY POINT FOR UINIT AND UCALL
C
      ENTRY UINIT
      ENTRY UCALL(N)
      RETURN
      END
```

The following output routine prints the emission or control factors on request.

```
      SUBROUTINE UWRITE(NO)
C
C
C
C
C        EFAC(F,T),F=1,4,T=1,4
C                  EMISSION FACTORS
C        CFAC(F,S,T),F=1,4,S=1,5,T=1,4
C                  CONTROL FACTORS
C        EM(F,S,T),F=1,4,S=1,5,T=1,4
C                  EMISSIONS
C        SEM(S,T),S=1,5,T=1,4
C                  EMISSIONS BY ECONOMIC SECTOR
C        TOTEM(T),T=1,4
C                  TOTAL EMISSIONS
C
C        T         TYPE OF EMISSION
C        1         PARTICULATES
C        2         SULFUR OXIDES
C        3         NITROGEN OXIDES
C        4         HYDROCARBONS
C
      COMMON/XCMN/TOTEM(4),SEM(5,4),EM(4,5,4)
C
C$$$$$ CLASS 1 VARIABLES (IMPORTANT RESULTS TO BE GRAPHED) ABOVE AND
C        CLASS 2 VARIABLES (PARAMETERS, ETC., THAT WILL PROBABLY NOT
C        BE REPORTED OR GRAPHED) BELOW.
C
      COMMON/XCMN/EFAC(4,4),CFAC(4,5,4)
C
```

```
      DIMENSION FUEL(4),ETYPE(4),SECTOR(2,5)
      DATA FUEL/6H COAL ,6H OIL  ,6HN GAS ,6H WOOD /
      DATA ETYPE/6H PART ,6H SOX  ,6H NOX  ,6H HYDC /
      DATA SECTOR/12HRESIDENTIAL ,12H COMMERCIAL ,12H INDUSTRIAL ,
     1    12H TRANSPORT  ,12H  ELEC GEN  /
C
C        MAKE THE WISSIM COMMAND  WRITE 1  A REQUEST FOR EMISSION
C        FACTORS AND  WRITE 2  A REQUEST FOR CONTROL FACTORS
C
C        FIRST CHECK NO
C
      IF(NO.EQ.1) GO TO 50
      IF(NO.EQ.2) GO TO 100
      WRITE(6,10) NO
 10   FORMAT(1HO,I6,29H WRITE ROUTINE IS NOT DEFINED)
      RETURN
C
C        EMISSION FACTORS
C
 50   WRITE(6,60)
 60   FORMAT(1HO,36H EMISSION FACTORS - (LBS/10**6 BTU) )
      WRITE(6,65)(FUEL(I),I=1,4)
 65   FORMAT(1HO,8X,4(A6,2X)/)
      WRITE(6,70)(ETYPE(I),(EFAC(J,I),J=1,4),I=1,4)

 70   FORMAT(4(1X,A6,4(2X,F6.3)/))
      RETURN
C
C        CONTROL FACTORS
C
 100  WRITE(6,110)
 110  FORMAT(1HO,10X,15HCONTROL FACTORS)
      DO 200 K=1,5
      WRITE(6,120)(SECTOR(I,K),I=1,2)
 120  FORMAT(1HO,12X,2A6)
      WRITE(6,65) (FUEL(I),I=1,4)
      WRITE(6,70) (ETYPE(I),(CFAC(J,K,I),J=1,4),I=1,4)
 200  CONTINUE
      RETURN
      END
```

Finally, we create a file of commands to input the energy use data (ENERGY (F,S); F = 1,4 and S = 1,5, where F and S are described in Tables B-1 and B-2). The initial data we will use are shown in Table B-4. Then, to change the energy use, we'll use AT commands to decrease the amount of coal used for electricity generation.

Oil and natural gas for electriciy generation will remain constant; oil for transportation will grow at 2 percent per year; natural gas for industrial purposes will gradually decrease,

Table B-4

Sector	Fuel (10^6BTU)			
	Coal	Oil	Natural Gas	Wood
Residential	20,000.	110,000.	125,000.	0
Commercial	0	50,000.	55,000.	0
Industrial	80,000.	15,000.	185,000.	0
Transport-ation	0	300,000.	0	0
Electricity Generation	240,000.	5,000.	30,000.	0

while coal for industrial uses grows. Other energy uses will remain constant. The commands are shown below.

```
SET ENERGY(ALL,1)=20000.,110000.,125000.,0.  $$ RESIDENTIAL $$
SET ENERGY(ALL,2)=0.,50000.,55000.,0.        $$ COMMERCIAL $$
SET ENERGY(ALL,3)=80000.,15000.,185000.,0.   $$ INDUSTRIAL $$
SET ENERGY(ALL,4)=0.,300000.,200.            $$ TRANSPORTATION $$
SET ENERGY(ALL,5)=240000.,5000.,30000.,0.    $$ ELECTRICITY GENERATION $$
AT 1978 SET ENERGY(1,5)=210000.              $$ COAL FOR ELEC GEN $$
AT 1980 SET ENERGY(1,5)=190000.              $$ COAL FOR ELEC GEN $$
AT SET ENERGY(2,4)=ENERGY(2,4)-              $$ DECREASE TRANSP BY 2% PER YR $$ $
     (.02*ENERGY(2,4))*IZZINC
AT SET ENERGY(3,3)=SATHR(IZZCUR-IZZBEG,      $$ DECREASE GAS USE IN IND$$ $
     185000.,50000.,10.)
AT SET ENERGY(1,3)=SATHR(IZZCUR-IZZBEG,      $$ INCREASE COAL USE IN INDUSTRY$$ $
     80000.,250000.,8.)
```

APPENDIX C

WISSIM FUNCTIONS

Several functions for curves that occur often in simulation modeling are included in WISSIM both as a convenience for the programmer and because such functions make the model more comprehensible. The functions may also be used in WISSIM SET commands. The functions are:

1) STEP - a step function
2) TABLE - linear interpolation with any number of points, but with fixed increments on the x-axis
3) XTERP - linear interpolation with any number of points and no fixed increment on the x-axis
4) SATHYR - a special saturation curve
5) SATH - a saturation curve with integer x values
6) SATHR - a saturation curve with real x values

The calling sequences for the functions are described and brief examples of each are given below.

A. STEP

1. Purpose

Given n sets of points $(x_1,y_1)...(x_n,y_n)$ where $x_1 < x_2 < ... < x_n$ and an x value, STEP returns the corresponding y value assuming that the original set of points define a step function.

2. Calling Sequence

STEP (X,XARRAY, YARRAY, N)

where

X is a real number giving the current x value,
XARRAY is the real array of known x values in ascending order, i.e., XARRAY(I) $\leq$ XARRAY (I + 1) for all I,
YARRAY is the real array of corresponding y values, and
N is an integer that contains the number of known points.

3. Function Value

Case 1. If X < XARRAY (1), then STEP = YARRAY(1)
Case 2. If X $\geq$ XARRAY(N), then STEP = YARRAY(N)

Case 3. If XARRAY(I) $\leq$ X < XARRAY(I + 1), then STEP = YARRAY(I).

4. EXAMPLE

Suppose

N = 3,
XARRAY(1) = 1970, YARRAY(1) = 5,
XARRAY(2) = 1975, YARRAY(2) = 20,
XARRAY(3) = 1990, YARRAY(3) = 10,

Then the graph is shown in Figure C-1.

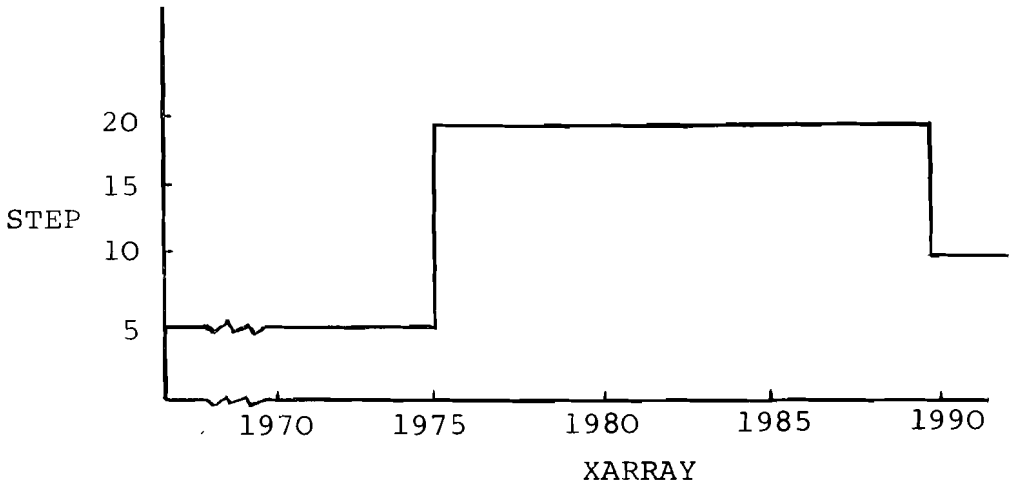


Figure C-1: A Graph of an Example STEP Function

If

X = 1960, STEP = 5
X = 1980, STEP = 20
X = 1990, STEP = 10
X = 2000, STEP = 10

B. TABLE

1. Purpose

The TABLE function duplicates the DYNAMO (Forrester, 1968) table look up and interpolation function. Given $x_1 \ldots x_n$, and z such that $x_i = x_{i-1} + z$ for all i such that $n \geq i > 1$, and $y_1, y_2, \ldots y_n$ and x, TABLE returns the y-value corresponding to x assuming that the given set of points define a function where the value of the function between the given points is found by linear interpolation.

2. Calling Sequence

TABLE(X,ARRAY,X1,XN,XI)

where

X is a real variable containing the current x value,

YARRAY is the array of known y values

X1 is a real variable containing the smallest value of the independent variable, i.e., TABLE(X1) = YARRAY(1),

XN is a real variable containing the largest value of the independent variable--it corresponds to the last element in YARRAY, and

XI is a real variable containing the increment for the independent variable.

Then the number of data points, N, is the integer portion of

$$\left(\frac{XN - X1}{XI} + .5\right) + 1.$$

X1, XN, and XI define a set of points (RX(1), RX(2),...RX(N)), where RX(1) = X1 and RX(N) = XN.

3. Function Value

Case 1. If $X \leq X1$, then TABLE = YARRAY(1).

Case 2. If $X \geq XN$, then TABLE = YARRAY(N).

Case 3. If $X1 < X < XN$, then the value of TABLE is the linear interpolation between the values of the function at RX(i) and RX(i + 1) where $RX(i) \leq X < RX(i + 1)$.

4. Example

Suppose:

(YARRAY(i), i=1,N) = 10., 20., 25., 5.

X1 = 1970.

XN = 1985.

XI = 5.

Then if:

X = 1960., TABLE = 10.

X = 1970., TABLE = 10.

X = 1972., TABLE = 14.

X = 1981., TABLE = 22.

X = 2000., TABLE = 5.

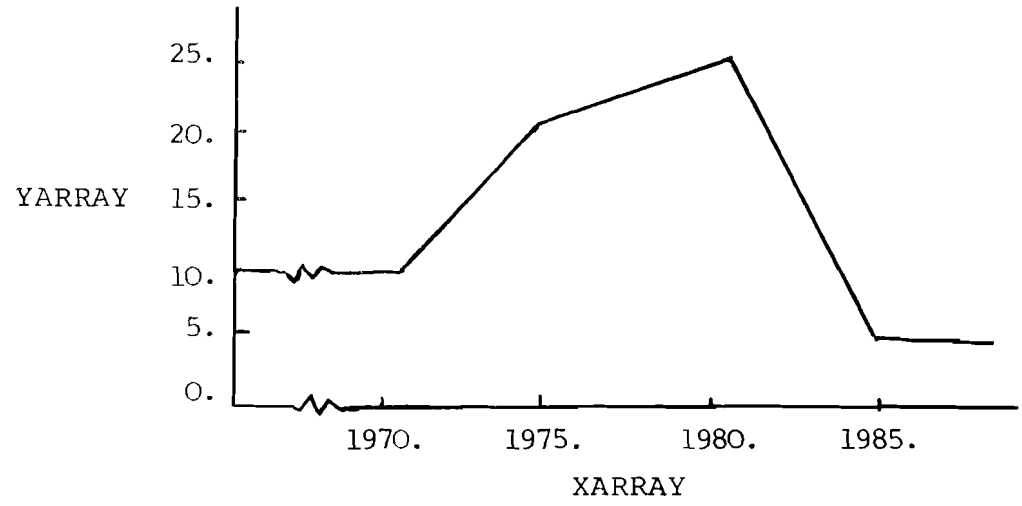


Figure C-2: A Graph of the Example TABLE Function

C. XTERP

1. Purpose

For any n set of points, (x_1,y_1), (x_2,y_2),...(x_n,y_n), where $x_1 < x_2 < \ldots < x_n$ the linear interpolation function, XTERP, may be used to calculate an unknown y for a specified x. The method of calculation, as the function title indicates, is a linear interpolation procedure. Any number of known points may be specified and for any x a corresponding y will be calculated.

2. Calling Sequence

XTERP (X, XARRAY, YARRAY, N)

where

X = a real variable containing the x-value currently being considered for which a y-value is desired,

XARRAY = a real array of initial x-coordinate values in ascending order, i.e., XARRAY(I) < XARRAY(I + 1),

YARRAY = a real array of initial y-coordinate values, and

N = an integer containing the number of points N $\geq$ 2.

3. Function Value

Case 1. If X $\leq$ XARRAY(1), then XTERP = YARRAY(1).

Case 2. If X $\geq$ XARRAY(N), then XTERP = YARRAY(N).

Case 3. If XARRAY(I) $\leq$ X $\leq$ XARRAY(I + 1), the value of XTERP is the linear interpolation between YARRAY(I), and YARRAY(I + 1).

4. Example

Data that is input: N = 3:

XARRAY(1) = 1970., YARRAY(1) = 50.
XARRAY(2) = 1980., YARRAY(2) = 90.
XARRAY(3) = 1990., YARRAY(3) = 110.

The graph is shown in Figure C-3. It is now desired to know the y-values for the years shown below. The values calculated would be:

1960., XTERP = 50.
1975., XTERP = 70.
1985., XTERP = 100.
2000., XTERP = 110.

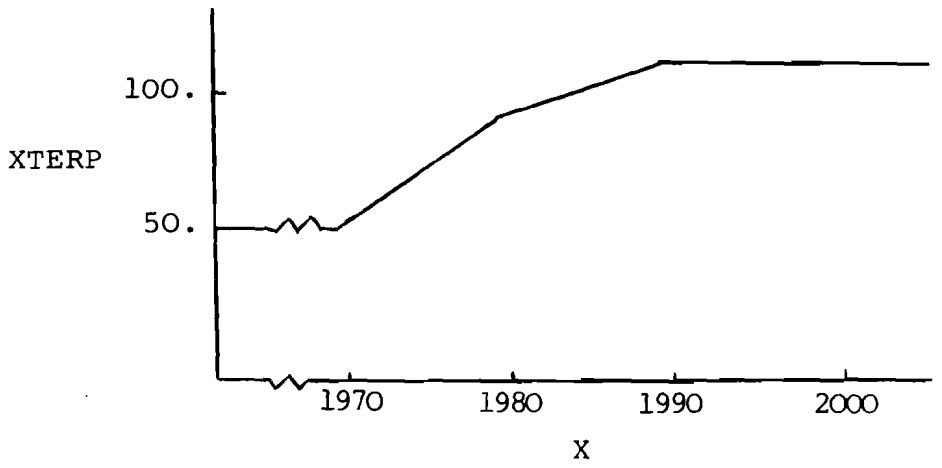


Figure C-3: A Graph of an Example XTERP Function

D. SATHYR

1. Purpose

Given that a parameter is expected to change as a function of time, SATHYR calculates changes that can be approximated by (1) exponential increase or decrease towards some saturation value, or (2) linear change. The function is flexible enough to allow the parameter to be constant for a number of years before exponential or linear changes start. The SATHYR function was written primarily for use with the Wisconson Regional Energy Model and must encompass a time period from 1970 to 2000 by one year increments.

2. Calling Sequence

SATHYR(K,BV,SV,I,J)

where

K is an integer variable containing the current year,

BV is a real variable containing the value of the parameter from last year. If SATHYR is to adjust the value in the first year, one must be careful to use an initial value of BV that gives the desired value for SATHYR after adjustment,

SV is a real variable containing the saturation value of the parameter when exponential behavior is desired or the desired value of the parameter in the year 2000 when the linear behavior is desired,

I is an integer variable containing the year in which the change from the initial value starts, and

J is an integer variable containing the half-time, or the number of years to achieve 50% of the difference between the initial value and the saturation. If linear behavior is desired, J should be 200 or greater.

3. Function Value

1. Exponential change (J < 200)

   i. K < I

      SATHYR = BV

   ii. K $\geq$ I

      SATHYR = BV + (SV - BV) * [1. - EXP(-.693/J)]

2. Linear change (J $\geq$ 200)

   i. K < I

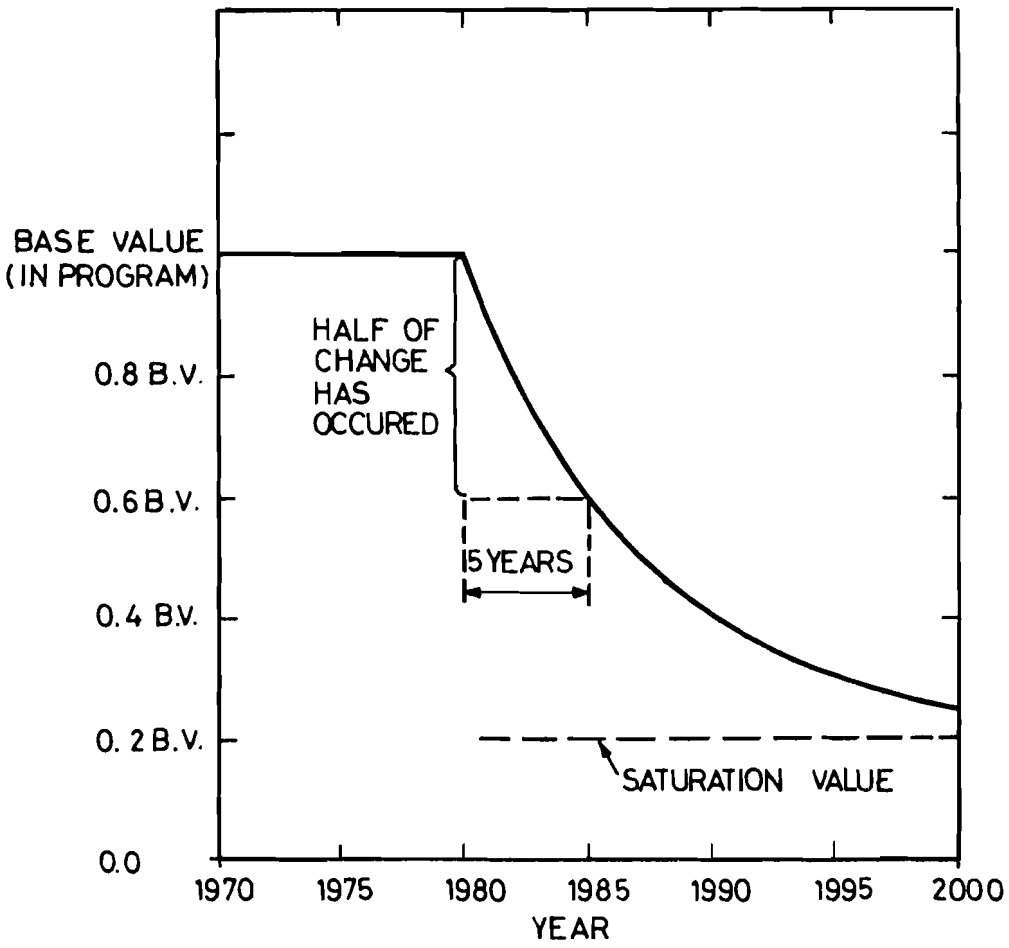
      SATHYR = BV

   ii. K $\geq$ I

      SATHYR = BV + (SV - BV)/(2001 - K)

4. Example

Suppose parameter FUNDER is expected to remain constant until 1981, when it begins to decrease: however, it is never expected to fall below 20% of its original value (ORIG, the 1970-80 value). By 1985, it should decrease to 60% of its original value. Then the following statements appear in the program: SAT = 0.2 * ORIG; FUNDER = SATHYR (K,FUNDER,SAT,1981,5). Figure C-4 shows the value



HERE IS HOW THE SATURATION WORKS, WITH THE THE BASE VALUE (B.V.) GIVEN IN THE PROGRAM. THE EXAMPLE SHOWN ABOVE ASSUMES

INPUT SATURATION OF 0.2 BASE VALUE
INPUT START OF EFFECT IN 1981
INPUT 5 YEARS AS THE TIME TO ACHIEVE 50 % OF THE CHANGE BETWEEN THE BASE VALUE AND THE SATURATION VALUE

Figure C-4: An Example of the SATHYR Function

for the parameter as a function of time. Note that by making the half-time short, a rapid convergence to the saturation value can be achieved, as illustrated in Table C-1.

TABLE C-1

Multiple of half-time	Fraction of difference between saturation and original value
1	$1 - 2^{-1} = 0.5$
2	$1 - 2^{-2} = 0.75$
3	$1 - 2^{-3} = 0.875$
5	$1 - 2^{-5} = 0.96875$
10	$1 - 2^{-10} = 0.9990235$

Therefore, after five multiples of the half-time, SATHYR has achieved 96.9% of the change between the original value and the saturation value.

E. SATH

1. Purpose

The purpose is the same as for SATHYR except that the year in which the change from the initial value starts must be 1970.

2. Calling Sequence

```
SATH (K,BV,SV,J)
```

where

K is an integer variable containing the current year,
BV is a real variable containing the value of the parameter from last year,
SV is a real variable containing the saturation value (exponential change) or the desired value in the year 2000 (linear behavior), and
J is an integer variable containing the half-time in years (exponential). J should be input as 200 or greater if linear behavior is desired. See SATHYR for more complete descriptions.

3. Function Value

1. Exponential change (J < 200)
   SATH = BV + (SV - BV) * [1. - EXP(-.693/J)]
2. Linear change (J ≥ 200)
   SATH = BV + (SV - BV)/(2001 - K)

4. Example

(See SATHYR description, SAT is the saturation value).

```
FUNDER = SATH(K,FUNDER,SAT,5)
```

F. SATHR

1. Purpose

SATHR is a general purpose satruation curve. Given a base value, a saturation value, and the x increment in which half of the change occurs, SATHR defines the curve and given any x value, returns the y value.

2. Calling Sequence

```
SATHR (X,YBV,YSV,HTIME) where
```

X is a real variable containing the current x value,

YBV is a real variable containing the base value, the value of SATHR when x is 0,

YSV is a real variable containing the saturation value, the value that SATHR approaches as x approaches infinity, and

HTIME is a real variable containing the half-time, the x increment in which half the change from the current value to the saturation value occurs.

3. Function Value

If x < 0, then SATHR = 0. Otherwise,

```
SATHR = YBV + (YSV - YBV) * [1 - EXP(-.693 * X/HTIME)]
```

4. Example

Let

```
YBV = 1.,  YSV = 10.,  HTIME = 4.
```

Then the graph of the function is as shown in Figure C-5.

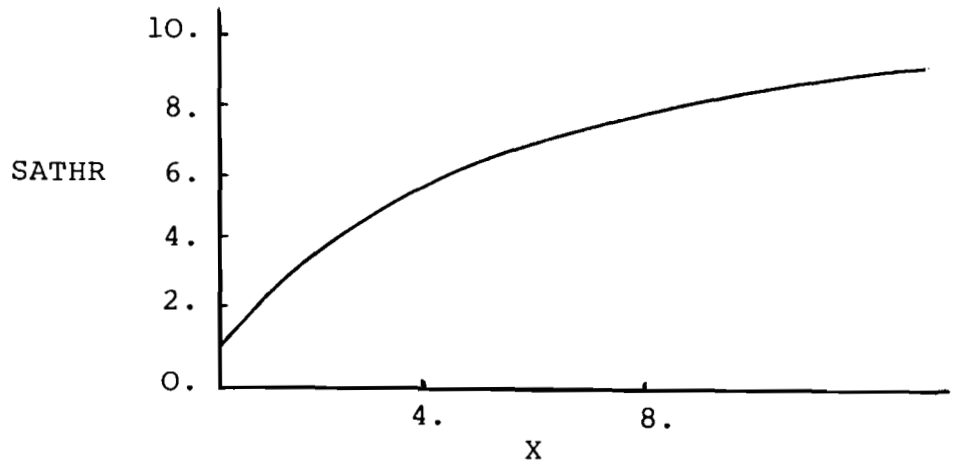


Figure C-5: A Graph from Example SATHR Function

When

x = 0., SATHR = 1.

x = 4., SATHR = 5.5

x = 8., SATHR = 7.75

APPENDIX D

INTERNAL DESIGN

1. Introduction

The external design of WISSIM should be clear from the general description given in Section II and the detailed description of the commands in Section IV. The linkage of WISSIM with the simulation model is by its nature very machine- and site-dependent; however, and overview is contained in Section III. This Appendix outlines very briefly the important features of the internal design of WISSIM. There is nothing in the rest of the report that depends upon the information contained in this appendix.

The discussion of the internal design begins in Section 1 with a description of the model variables and the variable name table or symbol table that WISSIM uses to store them. Section 2 is a general description of the scanner, that part of the system that interprets the command input stream. A short description of the SAVE file, the file that temporarily stores the state of the model at specified simulation intervals, is contained in Section 3. Then the operation of the SIMULATE command is discussed in Section 4. Detailed operations for the other commands are not described.

Figure D-1 is a diagram of the general organization of WISSIM. The initialization routines read the FORTRAN description of the model common blocks and build the Common Block Table and the symbol table, an internal WISSIM hash table that contains the name, type, dimensions, and location of each model variable. Then the scanner reads the commands from the user of the model. These may come from an interactive teletype or graphics terminal, from punched cards, of from a previously prepared input file. The scanned commands are sent to the appropriate command routine for execution. Execution of a SIMULATE, CALL, or WRITE command causes the simulation model or part of it to be executed.

2. The Scanner

The scanner is the part of the WISSIM that reads the input commands, determines that the syntax is correct, and passes a simplified version of the command to the appropriate command routine. The scanner in WISSIM is of a fairly common type that

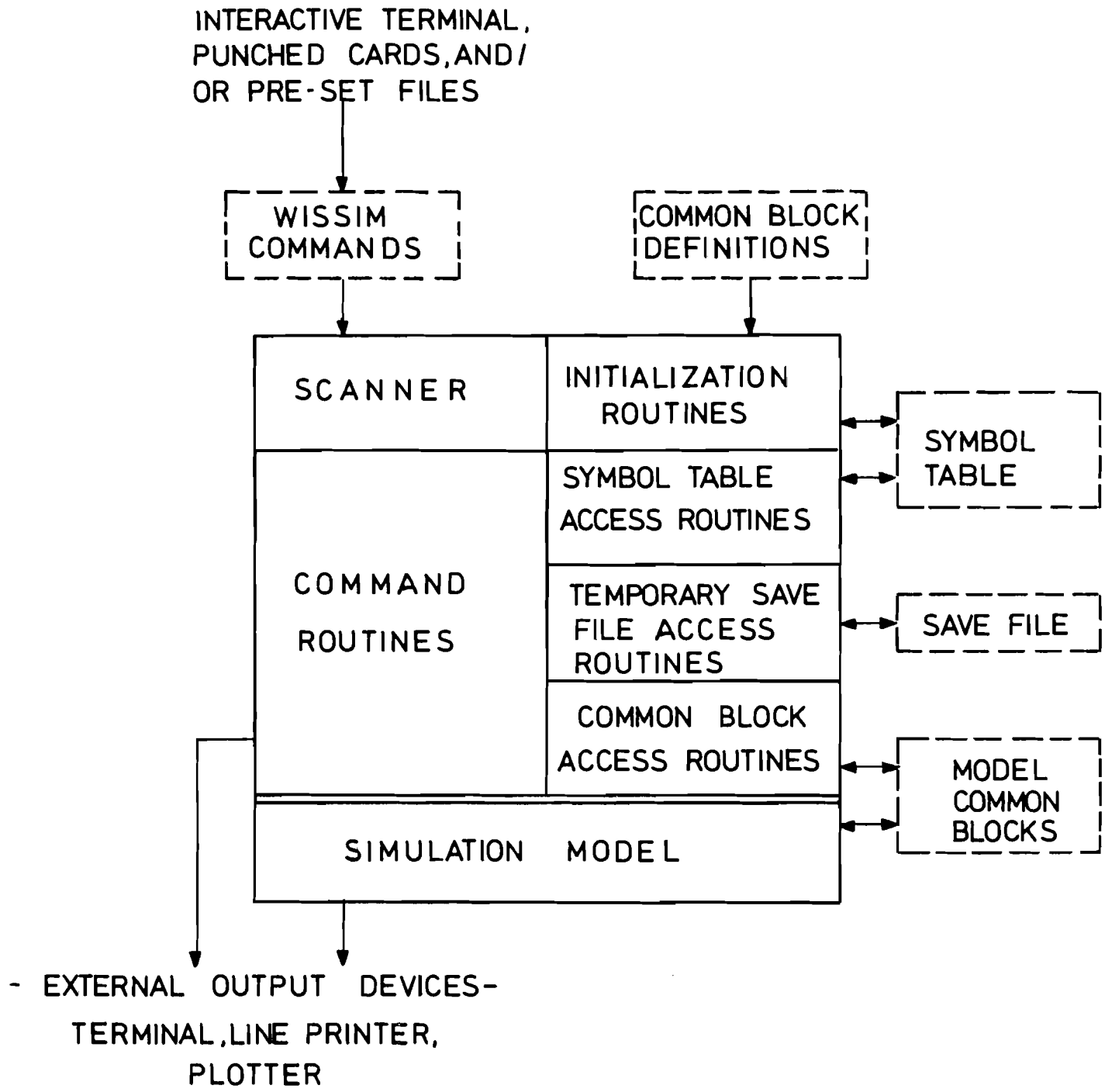


Figure D-1: Internal Organization of WISSIM

operates as a modified finite state automaton. The input string tokens are all possible character codes on the computer on which the scanner is run. The following input tokens are valid; all others are invalid and cause an error message to be printed.

SCANNER INPUT TOKENS

The letters A-Z,
The numbers 0-9, and
, = ( ) ' - * / + . $ @ space

Then for each valid input token i and state of the scan s, three pieces of information are stored:

$s_{i,j}$ is the new state of the scan
$a_{i,j}$ determines whether the input token should be added to the output token that is being formed, and
$p_{i,j}$ determines whether the current input token or the next character in the input string should be used as the new input token.

The states and the transition and output functions of the automaton will not be described here, but their form should be fairly obvious from the input and and output sets described here and the descriptions of the command formats in Section IV. The output tokens of the scanner are listed below and are integer coded for use by the command routines.

SCANNER OUTPUT TOKENS

Identifiers (Model and system variables, run identifiers, temporary variables)
WISSIM commands and reserved words
literals (enclosed in single quotes)
integers
real numbers
** , = ( ) - * / + $
a special end-of-command symbol

3. The SAVE file

Since it is possible to restore the simulation model to a previous state and to graph or report variables from a simulation

that has already been done, it is necessary to temporarily store the state of the system at every time period of interest. In a model with a large number of variables, this could be the most costly part of the simulation so an efficient method of data storage and retrieval from the SAVE file is an important consideration in WISSIM.

There are two different purposes for the SAVE file. The first is to save the state of the system for use as the initial conditions of another simulation. In that case the values of all variables must ususally be saved. The second is for graphs and reports. In that case it is not usually necessary to save all variables, but only a subset that contains the important results. In addition, in a large model that is composed of distinct submodels, such as the WISE model, it may often be necessary to look at only those variables associated with a particular submodel. Therefore to reduce computer costs, the design of WISSIM allows the user some flexibility in specifying the data to be saved, while providing default values that will produce the most commonly used options.

A state or a set of states associated with a particular simulation run is identified by a four (or six depending upon the computer) character run identifier assigned by the user. Information is both stored and retrieved under that name. The saved states are then divided into two sections: 1) an index of the run identifiers that may of may not be kept in core, and 2) the data itself stored on a temporary disk file. For efficiency of I/O, especially on the University of Wisconsin UNIVAC 1110 with its relatively large cost per I/O operation, the data is buffered so that an I/O operation is not required for each state saved or retrieved unless the number of variables is very large.

A simplified diagram of the save file is shown in Figure D-2. Common blocks are both stored and retrieved in the order they appear in the Common Block Table to make more efficient use of the data file buffer. The pointer labelled "common block inf." in the Common Block List is actually a pointer into the Common Block Table where the name, length, and address of each of the common blocks are kept.

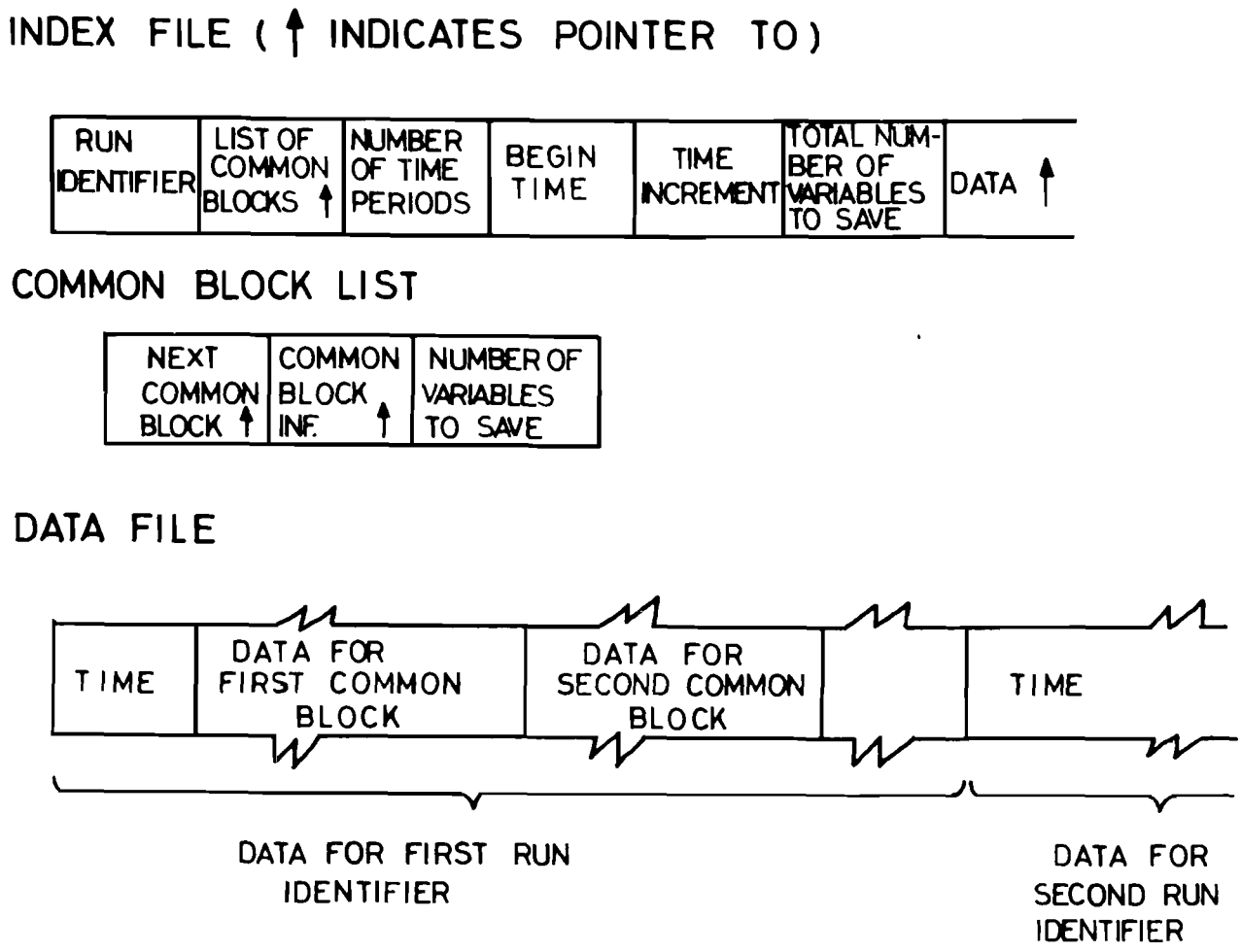


Figure D-2: The SAVE File Organization

Random I/O is not machine independent, but the SAVE file routines have been written in standard FORTRAN except for the subroutines that do the actual I/O. Those subroutines, one for input and one for output, have three parameters: 1) the buffer area, 2) the number of words to read or write, and 3) the address relative to the beginning of the file where the read or write is to start. Therefore the SAVE file procedures can be changed to any machine with standard FORTRAN by providing those two I/O routines.

4. The SIMULATE Command

The operation of the SIMULATE command is very straightforward. The only purpose in describing it here is to discuss the relationship with the AT commands. At every time interval implied by the parameters of the SIMULATE command: 1) any applicable AT commands are executed, 2) the simulation model is called, 3) data are saved on the SAVE file if requested by a previously entered SAVE command, and 4) a line of the report is printed if a during simulation report has been requested.

APPENDIX E

PAPERS IN THE IIASA PUBLICATION SERIES ON MANAGEMENT OF ENERGY/ ENVIRONMENT SYSTEMS

Keeney, R.L., Energy Policy and Value Tradeoffs, RM-75-76, International Institute for Applied Systems Analysis, Laxenburg, Austria, 1975.

Foell, W.K., Scenario Writing: One Component of a Systems Approach to Energy/Environment Management, RM-76-20, International Institute for Applied Systems Analysis, Laxenburg, Austria,1976.

Born, S., C. Cicchetti, R. Cudahy, J. Pappas, P. Hedrich, K. Lindner, D. Ufer, J.-M. Martin, D. Finon, Energy/ Environment Models and their Relationships to Planning in Wisconsin, the German Democratic Republic, and Rhone-Alpes, RM-76-21, International Institute for Applied Systems Analysis, Laxenburg, Austria, 1976.

Foell, W.K., The IIASA Research Program on Management of Regional Energy/Environment Systems, RM-76-40, International Institute for Applied Systems Analysis, Laxenburg, Austria, 1976.

Buehring, W.A., W.K. Foell, Environmental Impact of Electrical Generation: A Systemwide Approach, RR-76-13, International Institute for Applied Systems Analysis, Laxenburg, Austria, 1976.

Buehring, W.A., W.K. Foell, R.L. Keeney, Energy/Environment Management: Application of Decision Analysis, RR-76-14, International Institute for Applied Systems Analysis, Laxenburg, Austria, 1976.

Stehfest, H., A Methodology for Regional Energy Supply Optimization, RM-76-57, International Institute for Applied Systems Analysis, Laxenburg, Austria, 1976.

Buehring, W.A., R.L. Dennis, A. Hölzl, Evaluation of Health Effects from Sulfur Dioxide Emissions for a Reference Coal-Fired Power Plant, RM-76-23, International Institute for Applied Systems Analysis, Laxenburg, Austria, 1976.

Buehring, J.S., WISSIM: An Interactive Computer Simulation Control Language, RM-76-24, International Institute for Applied Systems Analysis, Laxenburg, Austria, 1976.

Forthcoming

Dennis, R.L., Regional Air Pollution Impact: A Dispersion Methodology Developed and Applied to Energy Systems, RM-76-22, International Institute for Applied Systems Analysis, Laxenburg, Austria, 1976.

REFERENCES

Buehring, W.A., W.K. Foell, and R.L. Keeney (1976), Energy and Environmental Management: Applications of Decision Analysis, RR-76-14, International Institute for Applied Systems Analysis, Laxenburg, Austria.

Buehring, W.A., W.K. Foell, P.H.Hunter, D.A. Jacobson, P.D. Kishline, J.L. Pappas, and D.B. Shaver (1974), Alternative Future Electricity Generation Requirements for the State of Wisconsin, Institute for Environmental Studies, Report No. 26, University of Wisconsin-Madison.

Cicchetti, C.J., and W.K. Foell, ed. (1975), Energy Systems Forecasting, Planning and Pricing, Proceedings for a French-American Conference, Institute for Environmental Studies, University of Wisconsin-Madison.

Foell, ed. (1976), Proceedings of Conference on Integrated Management of Regional Energy/Environment Systems, (In press).

Foell, W.K., J.W. Mitchell, and J.L. Pappas (1975), The Wisconsin Regional Energy Model: A Systems Approach to Regional Energy Analysis, Institute for Environemtnal Studies Report No. 56, University of Wisconsin-Madison.

Forrester, J.W. (1968), Principles of Systems, Wright-Allen Press Inc., Cambridge, Massachusetts.

Hilborn, R. (1973), A Control System for FORTRAN Simulation Programming, Simulation, 20, 1972-175.

Madison Academic Computing Center (1972), FORTRAN V Compiler Reference Manual for the 1108, The University of Wisconsin-Madison.

Meier, R.C., T. Newell, and H.L. Pazer (1969), Simulation in Business and Economics, Prentice Hall Inc., Englewood Cliffs, New Jersey.

Mitchell, J.W., and D. A. Jacobson (1974), Implications of Commercial Building Codes for Energy Conservation, Institute for Environmental Studies Report No. 42, University of Wisconsin-Madison.