AN ABSTRACT OF THE THESIS OF

Christoph Wiese for the degree of Master of Science in Industrial Engineering presented on June 9, 1987.

Title: Simulation of Mechanized Log Harvesting Systems

Abstract approved: Redacted for Privacy

Eldon Olsen

The focus of this research is to develop a personal computer based simulation model of the mechanized logging process, from felling until the log arrives at the sawmill. The SLAM II simulation language is used for modeling, and the main emphasis is on the overall performance of this system, and the interaction between the individual components.

The main approach will be, to break the logging process down into its components. Each component can then be analyzed and modeled to form a modular system. By giving the components/modules of a specific operation, these modules will then be arranged in the desired order by the simulation processor. Thus, the variations of a system can be explored and evaluated in their performance for different machine configurations. The possible machine configurations for a particular environment can be tested and compared.

A front-end interface for simulation inputs and outputs is developed. The input interface facilitates the user in entering input parameters to the

model without the need to learn the simulation language. The output interface produces easy to understand and readable outputs.

Simulation of Mechanized Log Harvesting Systems

by

Christoph Wiese

A THESIS

submitted to

Oregon State University

in partial fulfillment of
the requirements for the
degree of

Master of Science

Completed June 9, 1987

Commencement June 1988

APPROVED:

Redacted for Privacy

Associate Professor of Industrial Engineering in charge of major

Redacted for Privacy

Head of Department of Industrial Engineering

Redacted for Privacy

Dean of Graduate School

Date thesis is presented June 9, 1987

Typed by Christoph Wiese for Christoph Wiese

ACKNOWLEDGEMENTS

During the course of this study many individuals helped me by the completion of this project. I would like to use this opportunity to express my thanks to all these people. Special thanks are due to:

Dr. Eldon Olsen, my major professor, for his counsel, guidance, and his patiences.

Don Schuh, Research assistant in the Department of Forest Engineering at Oregon State University for the countless hours of his time during the initialization phase of the project.

My family at home and Petra, who supported and encouraged me despite the distance between us and waited so patiently for my returning.

This work is dedicated to my father, Hermann F. Wiese

TABLE OF CONTENTS

TABLE OF CONTENTS
(continued)

LIST OF FIGURES

LIST OF TABLES

SIMULATION OF MECHANIZED LOG HARVESTING SYSTEMS

I. INTRODUCTION

In the Pacific-Northwest we can observe a significant change in the environment in which a logging operation typically takes place. The main reasons for these changes are the increased proportion of merchantable second growth forest and an increased public awareness concerning forest issues.

In the western region of the United States, we have typically had a low level of mechanization. Now, however, a wave of mechanization of the logging process is taking place because of the increased operation in second growth forest:

> In the past when stems were large, the logger had very little latitude on how the trees were handled and processed. Consequently, trees were felled and bucked at the stump, forwarded to the landing as log lengths, loaded on trailers and hauled to the mill.
>
> However, with smaller stems the opportunities for handling and processing are much greater. By combining material handling phases and/or processing material at the landing or stump, there are far more variables in the "Harvesting Cost Equation" (Carson, 1984).

For nearly every step of the logging process, material handling systems and automatic processors have been developed. A great variety of devices have been constructed and adapted to the logging environment. Thus, a great number of specialized machines exist. The development and investment costs for these machines have exploded with the increase of their complexity. On the other side, the recession in the past few years has hit the forest products industry, mostly because of the close relationship between the construction industry and the forest products industry. This had, of course, significant influence on the logging industry as the supplier of raw material to the forest products industry. Demand and profits went down considerably.

The social environment in which a logging operation takes place has changed:

> The public has become more concerned with conservation, wilderness, recreation areas and aesthetics. This concern has the effect of placing rigid restrictions on the timber cutting practices. Further restrictions are placed on the timber cutting practices by governmental practices (Anonymous, 1971).

These changes in the economical and social environment have created a number of problems for both the manufacturer of logging equipment and the logging companies. In the case of the machine manufacturer, some of the problems are:

- What systems should be developed to meet public concern and the needs of the loggers ?
- What is the economical performance of a proposed system?
- What is the utilization of the machines?
- How can a harvesting system be adapted to a specific environment?
- Where can I improve my harvesting system?

The logging operator is concerned about questions like:

- Which harvesting system should I use for which environment?
- What is the expected performance of the system in terms of economics, utilization, material flow, capacities?
- How can I improve my system?
- What machines should I buy for my logging environment?

- What machines should I buy for my logging environment?
- What is the best mix of equipment (type and number) and labor for a given logging site?

Thus, it can clearly be seen that there is need for a tool to analyze and optimize the timber harvesting process. Generally, this involves mathematical modeling of one sort or another. However, in complex systems, the mathematics can be extremely difficult to apply. Simulation is one tool to analyze more complex systems since this method is often easier to apply than pure analytical methods, and hence can be employed by many more individuals. Therefore, we will employ simulation to develop such an analytical tool.

II. OBJECTIVE

The main goals of this thesis will be:

1) Definition of a model of the harvesting process suitable for analyzing a broad range of harvesting configurations.

2) Development of a general simulation processor with the following abilities:

   - Capable of running in an IBM-PC or compatible computer hardware environment.
   - The user should be able to define, on-line in an interactive session, the specific machine configuration and environment of a timber harvesting process.
   - The program should verify the given system and parameters.
   - Output of the desired results in an easy to analyze and readable form.
   - Since the environment and machine configurations change for every application, the model should be built in a modular fashion, and be easy to use.

III. PROBLEM ANALYSIS

A. LITERATURE REVIEW

There have been many attempts in simulating (modeling) the timber harvest process. In their state-of-the-art report Goulett, Iff and Sirois (Goulett, Iff, Sirois; 1979) classify these attempts into two general classes:

1) Tree-to-Mill models:

   The entire process from felling until the log arrives at the sawmill is modeled.

2) Phase models:

   A certain phase/part of the process is modeled. Most work so far has been done in this second class. Models for specific phases have been developed, like

   - Simulation of the operation of a log landing for a Heli-Stat airship in old growth timber stands (Gerstkemper, 1982).
   - Simulation of a helicopter yarding system (Ledoux, 1975).
   - Loading and hauling subsystems of a logging system (Johnson, 1970).
   - Simulation of a Rubber-Tired Feller-Buncher (Winsauer, Bradley, Dennis; 1982).
   - Harvesting machines for mechanized thinning (Newnham, Sjunnesson; 1969).
   - Simulation of a timberyard (Lohman, Lehnhausen; 1983).

- Mechanized felling in dense softwood plantations (Winsauer, 1984).

These are only a few models from the great existing number of simulation approaches for a single or a few phases of the logging process. However, in the first class Goulett, Iff and Sirois only identify eight models, which are concerned about the whole process. These approaches have different features and capabilities:

- The Auburn pulpwood harvesting system simulator (Bussel, Hool, Leppet, Harmon; 1969).

  Simulates eight shortwood and six tree length harvesting configurations.
- The forest harvesting simulation model (Killham, 1975).

  Capable of addressing variability in individual operations, but doesn't collect statistical data for these estimates.
- Full-Tree chipping and transportation simulator (Bare, Jayen, Anholt; 1976).

  Designed to simulate harvesting, in-woods full-tree chipping and transport to the mill.
- Georgia Tech model (Stark, 1975).

  Is one of the earliest attempts in simulating forest harvesting systems.
- Harvesting System Simulator (O'Hearn, Stuart, Walbridge; 1976).

  The most complex model found, capable of modeling many machine configurations in great detail.

- Residues for Power (Bradley, Biltonen, Winsauer; 1976).

  A more general material handling model that can be used to simulate timber harvesting and transportation systems.
- Simulation applied to logging systems (Johnson, 1976).

  General model that is adaptable to a variety of logging configurations.
- Timber harvesting and transportation simulator (Martin, 1976).

  Simulates the standard harvesting configurations.

All these models examine the timber harvest process, with great variety in both, function and detail. They are considered to be the first generation of timber harvesting simulators and are about ten years old. However, presently many of these models are obsolete. The assumptions and configurations used are no longer relevant, due to the mechanization wave. The environment has changed significantly, which isn't reflected in these models. Thus, today there exists a need for a new approach in simulating the timber harvesting process.

D. B. Webster (Webster, 1984) suggested the following approach for a simulation project. He divides the approach into three phases, with certain activities:

1) Initialization Phase:
   - Problem definition
   - Definition of objectives and criteria
   - System definition

     The most important and difficult step in a simulation problem is to define the simulation. The following questions should be asked:

a) What are the questions to be answered by the model?

b) What are the performance variables of interest?

c) What output is required?

d) Are only mean values required, or is a distribution format needed?

2) Modeling Phase:

Phase one has been successfully performed; the problem environment has been well defined. Tasks now required are:

- Model formulation
- Data preparation
- Selection of programming language
- Model development
- Coding of the model
- Model verification, and
- Model validation

3) Implementation Phase:

This last phase determines whether or not all previous effort has been in vain. Tasks which must be accomplished include:

- Strategic planning
- Tactical planning
- Experimentation
- Analysis of result
- Decision or indications obtained by the model
- Follow up studies

This thesis will deal only with phase one and two of this approach, because of the size and the complexity of the topic.

B. SIMULATION

Simulation is one of the most widely used techniques in operations research and management science. The recent advances in computer hardware and software makes this tool even more accessible for the decision-maker and researcher.

Simulation is an excellent analytical tool to view and examine complex systems. It makes it possible to explore systems in their totality as well as in great detail. However, simulation has one drawback, it is not an analytical optimizing technique. Still, this can be overcome by simulating the desired system several times, each time with a different set of parameter values. With the help of sensitivity analysis a projection can then be made to approximate the desired optimization or minimization objective. Simulation is quite often employed to analyze what-if scenarios of complex systems, where it is too costly or simply not possible to run a real-time experiment. As was demonstrated in the Literature Review, simulation techniques have been successfully employed to analyze the various aspects of the timber harvesting process.

In most simulations, time is the major independent variable. Other variables included in the simulation are functions of time and are the dependent variables (Pritsker, Pedgen; 1984).

Simulation models of systems can be classified generally into three classes: discrete, continuous or combined discrete-continuous models.

In a discrete simulation the dependent variables change discretely at specified points in simulated time. These points are referred to as event times. Depending on whether the discrete changes in the dependent variable can occur at any point in time or only at specified points, the independent time variable may be either continuous or discrete.

In a continuous simulation the dependent variables may change continuously over simulated time. The system may be either continuous or discrete in time. This is depending on whether the values of the dependent variables are described as differential equations for any given point in simulated time, or if with certain events the change in value is modeled.

In the combination of discrete and continuous models the dependent variables may change discretely, continuously, or continuously with discrete jumps super imposed. The time variable may be continuous or discrete.

The timber harvesting process can be modeled with all three approaches depending on the simulation language used. However, by using a simulation language which is capable of employing all three techniques, the greatest possible flexibility can be maintained. Thus, it is the responsibility of the modeler to choose the appropriate technique and language.

C. PROCESS DESCRIPTION

The timber harvesting process can be broken down into seven major sub-processes (Kellogg, July 1986):

1) Felling
2) Bunching
3) Processing at stump
4) Primary Transport
5) Processing at landing or central site
6) Secondary Transport
7) Loading

Figure 1 (Kellogg, July 1986) shows a flow chart of the harvesting system. Figure 2 (Sessions, 1985) shows the same system, only this time in terms of the work elements. The main work elements of the timber harvesting process are:

- Felling
- Delimbing
- Measuring
- Bucking
- Topping
- Bunching
- Forwarding/Skidding/Yarding
- Loading
- Sorting
- Debarking
- Chipping
- Hauling

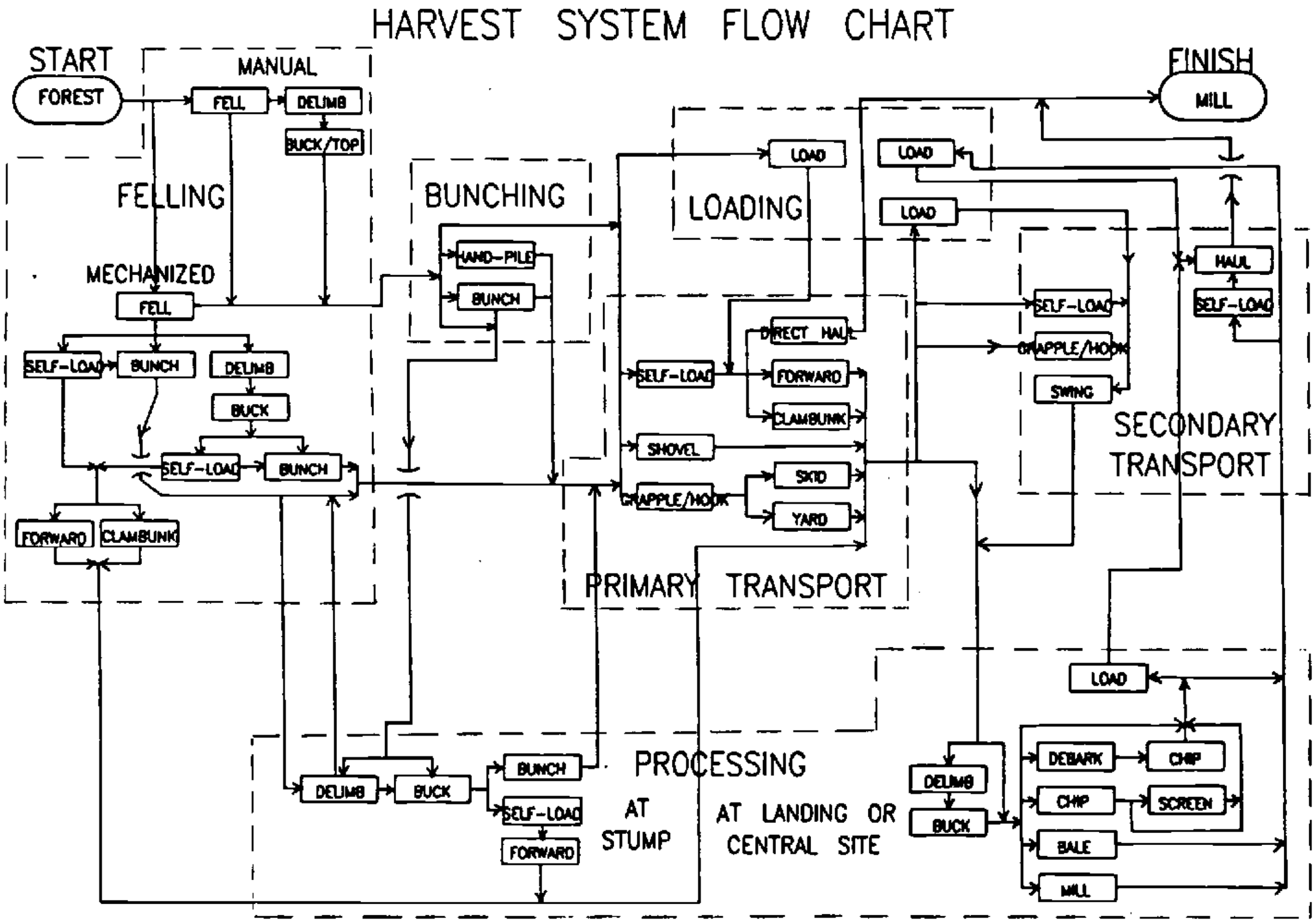


Figure 1: HARVEST SYSTEM FLOWCHART

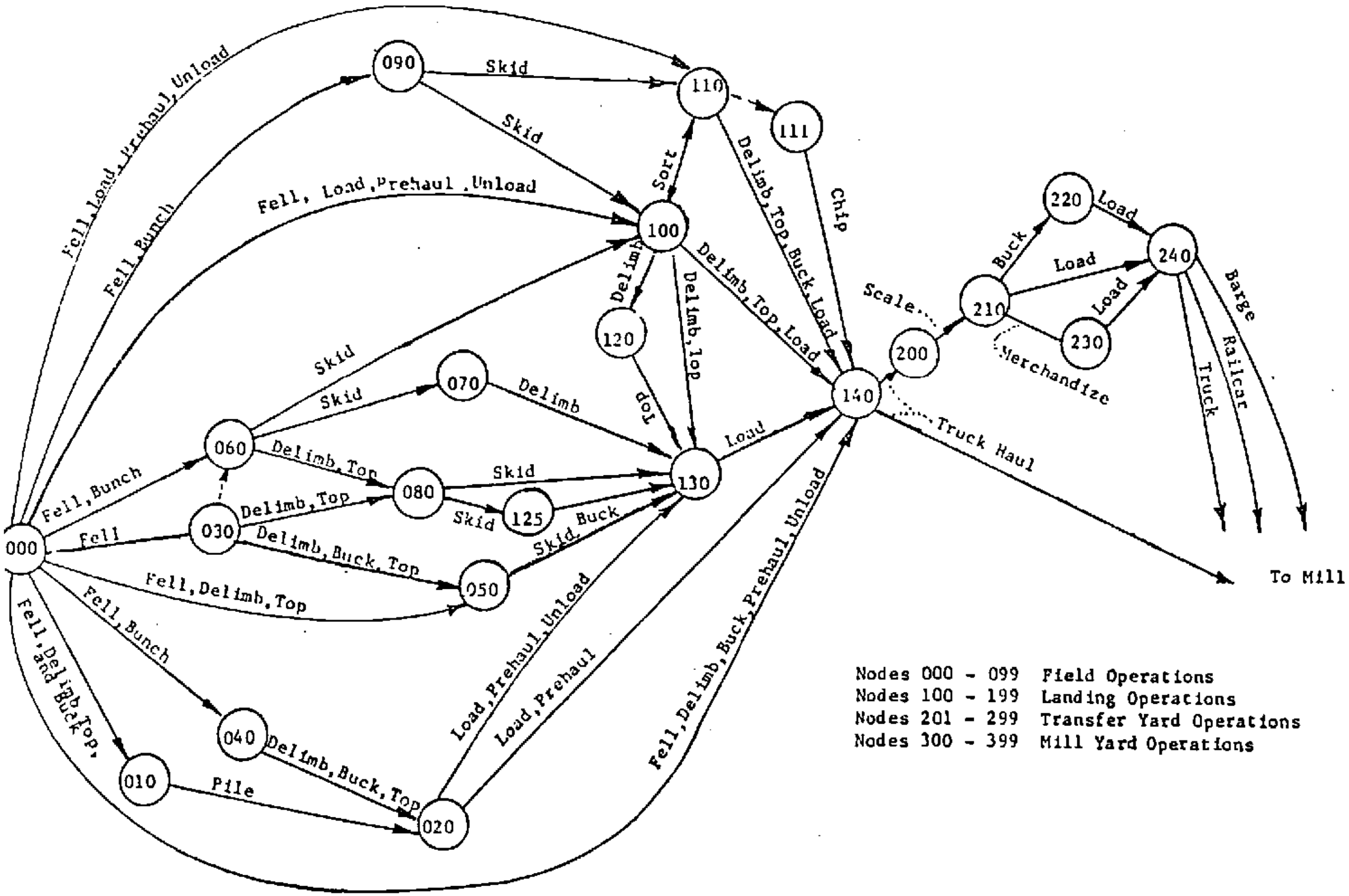
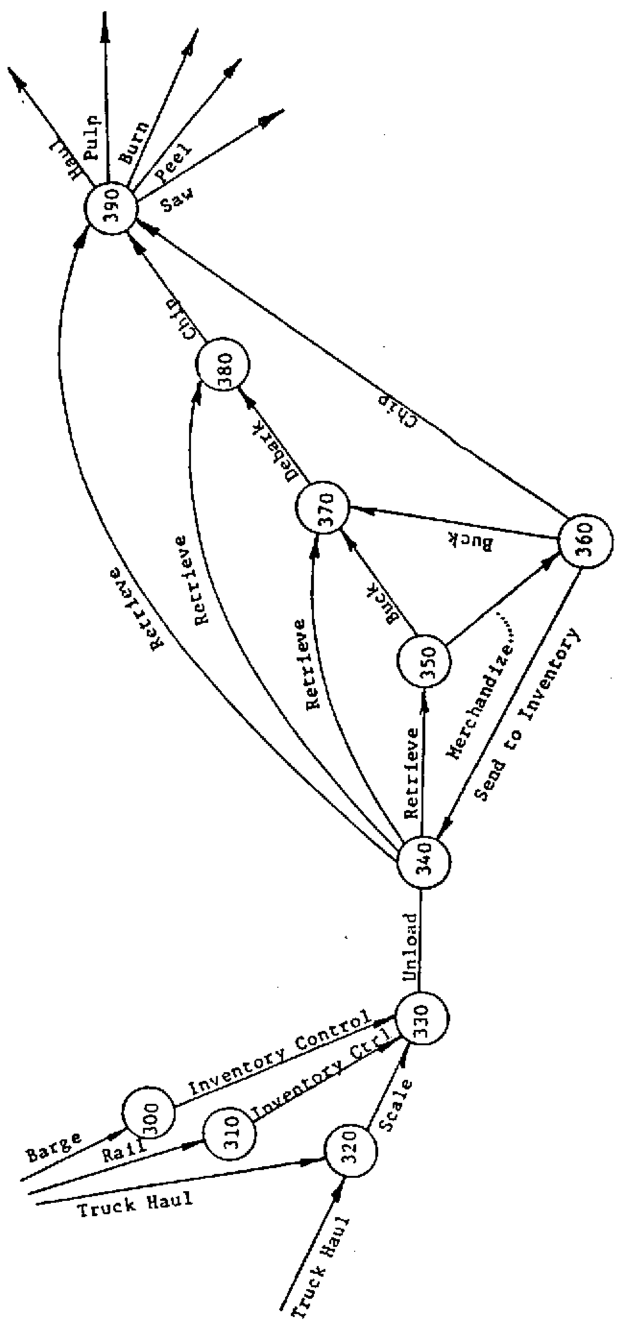


Figure 2a: HARVEST SYSTEM WORK ELEMENTS

Figure 2b: HARVEST SYSTEM WORK ELEMENTS



For a good discussion and explanation of these terms see Simmons, 1979.

Both figures demonstrate the complexity of the timber harvesting process. Due to the mechanization machines can combine several work-elements or even sub-processes into one machine. Logging is a serial operation; that is, certain steps must be performed in a given order so the objective may be achieved. However, the order of these steps varies from system to system (Conway, 1976). Moreover, work elements like delimbing and bucking can happen at nearly every stage of the harvesting process. Therefore, in different harvesting systems, different types of products can flow through at the same stage.

There can be restrictions on the inventory buffers before each process. Those inventory limits can be a minimum size of inventory required to operate a certain machine (ex. Chipper) or a maximum inventory allowable (ex. space constrains in the woods). There may be a startup level to each of those cases, after the minimum or a maximum inventory level has been reached. Also an initial startup-inventory level can apply for a work-element or stage.

Some work-elements require a preliminary loading action. This loading action can be provided by the machine itself (ex. self-loading truck) or by a loading device (ex. Log-Loader). It is possible for work elements on different stages to share the same loading device.

The logger or researcher has to match desired product mix with available technology and required performance criteria (Garland, 1986). Figure 3 (Garland, 1986) shows possible products emerging from the harvesting process. In the case of technology, we have to consider equipment, systems, techniques, and the labor force. Figure 4 (Garland, 1986) and 5 (Garland, 1986) give an overview for possible technology used for felling, yarding, loading, hauling and field processing. Some of the performance criteria are timber size capability, production potential, cost of production, topography limits, road access requirements, availability and environmental limitations.

A simulation model capable of accommodating all the constraints and desired properties has to be very general and broadly designed.

Figure 3: TIMBER PRODUCTS

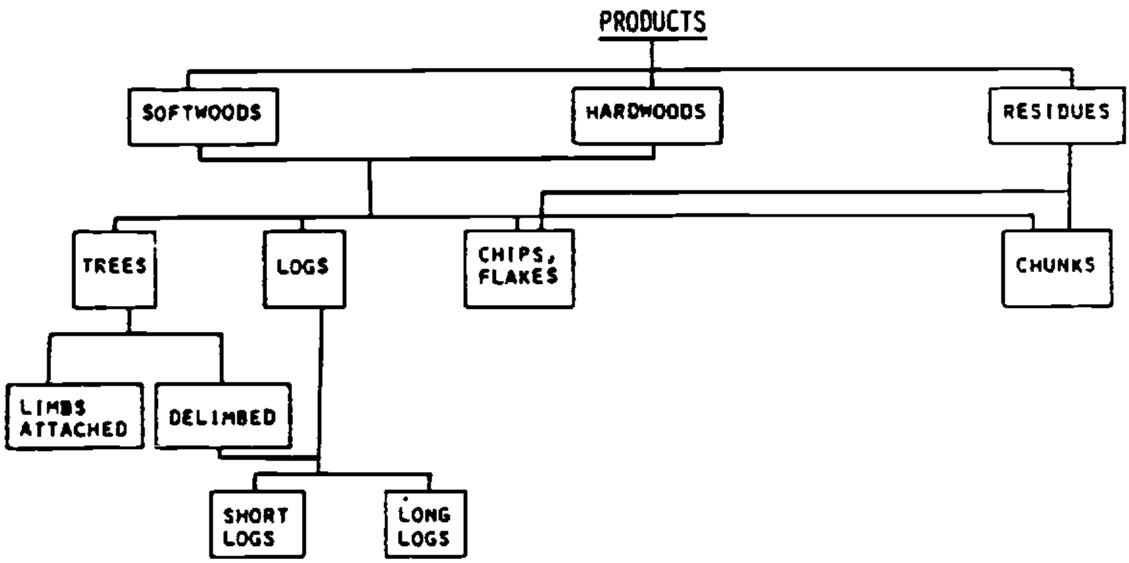


Figure 4: HARVEST SYSTEMS TECHNOLOGIES

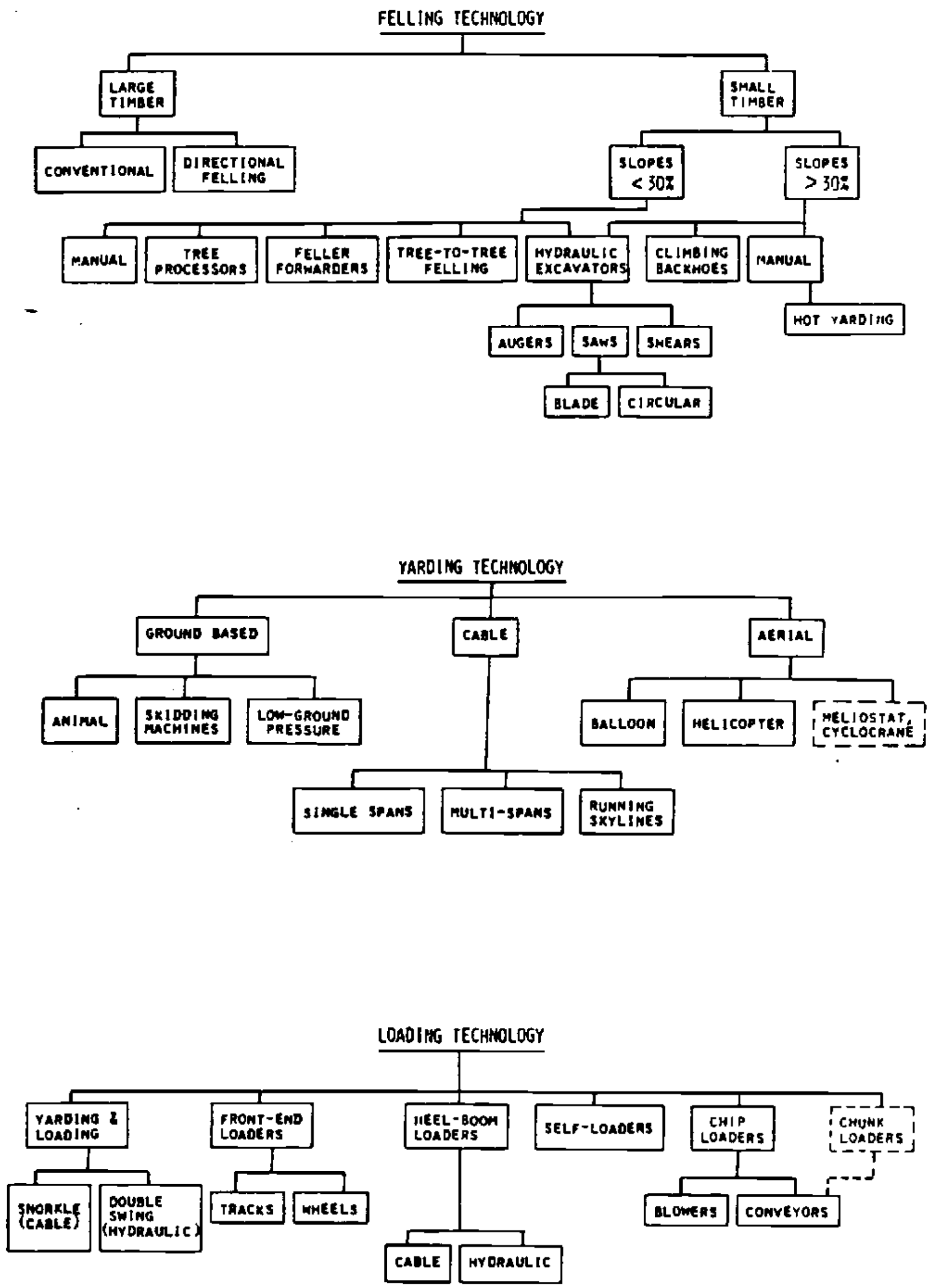
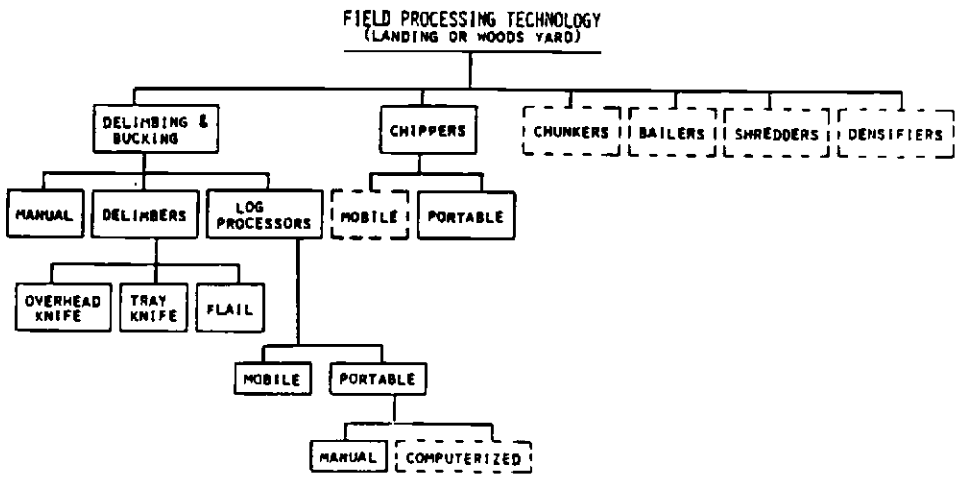
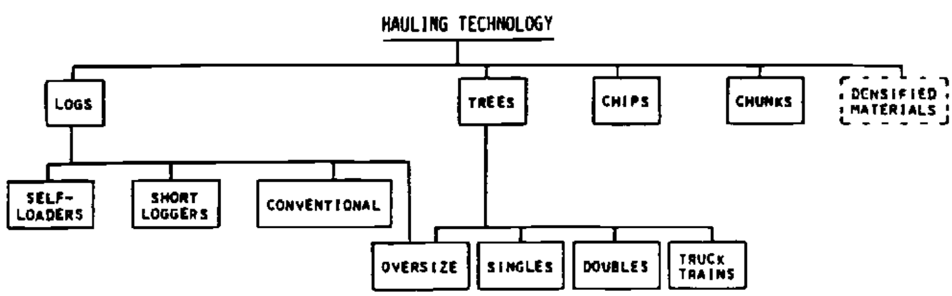


Figure 5: HARVEST SYSTEM TECHNOLOGIES





IV. MODELING APPROACH

A. PROGRAMMING LANGUAGE

The simulation model for the timber harvesting process was developed using SLAM (Simulation Language for Alternative Modeling) network modeling. SLAM is an event-oriented or a process-oriented simulation language and was developed by Pedgen and Pritsker (see Pritsker, Pedgen; 1984). In SLAM a discrete change system can be modeled within an event orientation, process orientation or both. Continuous change systems can be modeled using either differential or difference equations. Combined discrete-continuous change systems can be modeled by combining the event and/or process orientation with the continuous orientation. In addition, SLAM incorporates a number of features that correspond to the activity scanning orientation.

SLAM as the simulation language was chosen because of a number of reasons:

1) SLAM is available for an IBM-PC environment.
2) SLAM unifies all three modeling approaches within one simulation language; moreover, all three approaches can be combined within the same simulation model.
3) SLAM is based on FORTRAN, thus, easy to learn and usable on a great variety of computers. It is possible to port the developed programs and simulation models with a minimum effort

to other computing hardware, even to mainframes if the need arises (Lilegdon, O'Reilly; 1986).

4) A SLAM language package was available through the Department of Forest Engineering at Oregon State University.

In the process orientation in SLAM, a modeler combines a set of standard symbols, called nodes and branches, into an interconnected network structure which represents the system of interest pictorially. After the network model of the system has been developed, it is translated into an equivalent of SLAM program statements for execution on the computer (Law, Kelton; 1982). In the event orientation and continuous orientation of SLAM, the modeler defines as FORTRAN subroutines the events, potential changes to the system when an event occurs, and the differential or difference equations which describe the dynamic behavior of the system. These subroutines are then linked to the SLAM Execution Processor, which performs the actual simulation. SLAM also provides the modeler with a set of subroutines, which can be employed in the custom FORTRAN coding. These pre-written subroutines provide an interface to the SLAM network, and ease the task of statistical data collection. It is not the scope of this Thesis to provide the reader with a working knowledge of the SLAM simulation language (for a complete and thorough discussion of SLAM as a simulation language see Pritsker and Pedgen, 1984; Lilegdon and O'Reilly, 1986; and O'Reilly, 1984). To understand completely the programming done in this research, a good knowledge of FORTRAN is also mandatory (for references of the FORTRAN computer language see Etter, 1984; and the Microsoft FORTRAN Compiler Manuals, Microsoft 1985).

The simulation model was programmed in SLAM II for the IBM-PC, available from Pritsker & Associates Inc., West Lafayette,Indiana and the Microsoft FORTRAN Compiler Version 3.31 from Microsoft Corporation, Redmond, Washington. The programs created will run under MS-DOS (Microsoft Disk Operating System) Version 2.11 and higher.

B. THE SIMULATION MODEL

1. GENERAL REMARKS

The simulation model consists of three major parts. First the SLAM Network model, second the FORTRAN written subroutines, which accompanies the network and contains routines to initiate the simulation process, statistical data sampling functions, routines to assign values to variables and the output user-interface. The third part is the user interface for inputting, editing and printing the simulation parameters of the harvesting system. This user interface is also written in FORTRAN, which produces files that can be read by the SLAM simulation language. Therefore, it is possible to specify harvesting systems in advance, store them on mass storage media like floppy disks or hard disks and retrieve them whenever desired. The user-interfaces are described in more detail in their respective section.

The LOGGING SIMULATOR (LOGSIM) can be viewed as a general framework model of the log harvesting process. The user defines the desired harvesting system as a set of parameters. This set of parameters can be viewed as a customized model by itself and will be referred to as the harvesting model throughout this text. After the harvesting system has been specified with the input program, a customized SLAM Execution Processor has to be executed. This program will perform the actual simulation and gives a detailed simulation results output in readable form at the end of the run.

LOGSIM is capable of simulating a harvesting system from the felling operation until the log arrives at the sawmill or timberyard. The harvesting of one timber stand at a time can be modeled. The model consists of thirteen processes, which can be arranged in nearly any desired order. However, the model always starts out from process #1 and must end with either Process #12, #13, or both. Process #11 has the ability to divert the material flow into two branches, therefore, making it possible to simulate sorting operations or systems with two primary products like pulpwood and sawlogs. Each of the processes can employ different types of machines that have to be specified. In total, the simulation model can handle up to 90 machines within the system. The properties and capabilities of those thirteen processes are discussed in more detail in the following sections.

In order to simulate the flow of different materials through the harvesting system, the user can specify up to four material frequency distributions with up to ten frequency classes. An example is that it is possible to distinguish between whole trees, bucked logs, pulpwood pieces and sawlogs within the model and investigate the influence of piece sizes. By specifying different material frequency distributions for each simulation run, the performance of a given harvesting system configuration can be evaluated.

The model is capable of investigating the properties and influence of inventory buffers throughout the harvesting system. Optionally, the user can also model machine breakdown through the use of cumulative frequency distributions which describe time between machine breakdowns and repair times.

Times are generally described in decimal format throughout the simulation. An example is that 1.0 hrs is equal to one hour, 0.5 hrs are 30 minutes and 0.1 hrs are 6 minutes.

In the following sections we will describe the workings of the simulation network and FORTRAN programs.

2. THE SLAM NETWORK

The SLAM network consists of three subnetworks. The first is used to initialize the simulation, the second to model machine breakdowns and the third to model the actual harvesting process. The third network is the main logic to simulate the harvesting process. It consists of several subroutines, one for altering resource capacities, one for modeling time delays, one for each process and several help routines for controlling the simulation.

The model uses approximately 13800 words out of 16000 available reserved by the SLAM processor to define a network. Thus, 2200 Words still can be used for further modeling or increasing the number of machines the model is capable to handle if necessary. The relatively small size of the model was achieved by indexing processes and variables and using FORTRAN written user functions, thus, replacing SLAM code with FORTRAN statements which are not accounted for in the space reserved by the SLAM processor for the network description. Therefore, it is possible to build models which use up to 640 K on the IBM-PC even if the SLAM processor only occupies 320 K of memory. The current version of the LOGSIM requires approximately 420 K available memory to run, thus, leaving an additional 220 K free for further programing of FORTRAN user functions.

In most simulation models an entity that "flows" through the network represents a material unit, a customer, or a work piece. In the SLAM Network of the LOGSIM, however, an entity within the network represents a *machine* rather than a tree or cubic foot of wood. Each machine entity has seven ATTRIBUTES attached to it describing various parameters that are used to control the flow of the entity through the model (see Appendix A. 2).

The network model employs the concept of resources to indicate if a given machine is active and available and to control the simulation. In Appendix A. 1, an output of the SLAM Network is given along with several tables to describe which variables carry which values.

The ARRAY function available in SLAM II was heavily employed in the modeling process. These ARRAY functions and the XX(i) variables contain the parameters that describe the harvesting model and are also used to control the simulation. For a description of what the XX(i) variables represent, see Appendix A. 3 . For the description of the ARRAY variables, see Appendix A. 4.

For a description of the initialization of the simulation model and the use of FORTRAN user functions see Appendix A. 6.

When starting a processing cycle the entity seizes a machine (see also section IV.*B*.4), checks if a loader is required for this process and if so also seizes a loader (see likewise section IV.*B*.10). Then, the entity is routed to the main processing subroutine. There the actual load size of this run is determined according to the specified distribution for this process and the load size capacity of the machine. This is done by a

FORTRAN written user function (for a listing of the FORTRAN user functions see Appendix B). It should be noted that each time a FORTRAN user function is called the variable XX(5) should be indexed with the number describing which user function ought to be performed. This is done to overcome a flaw in the SLAM Processor, that does not transfer this number correctly when using the USERF(IFN) function. The model then performs several inventory calculations and checks, tests if the process is ended, e.g. if this is the last load, and then models the loading function if one is required. After this the actual machine processing, e.g. the time delays because of machine action, are accomplished (see also section IV.*B*.4). The model proceeds then with calculating the new inventories, checking inventory levels, adding machine hours and collecting statistics on the inventories. The resource machine is freed and the gates of the current and following process are pushed open to enable continuation of the simulation. Finally, the entity is routed back to the process where it originated and the whole cycle begins again.

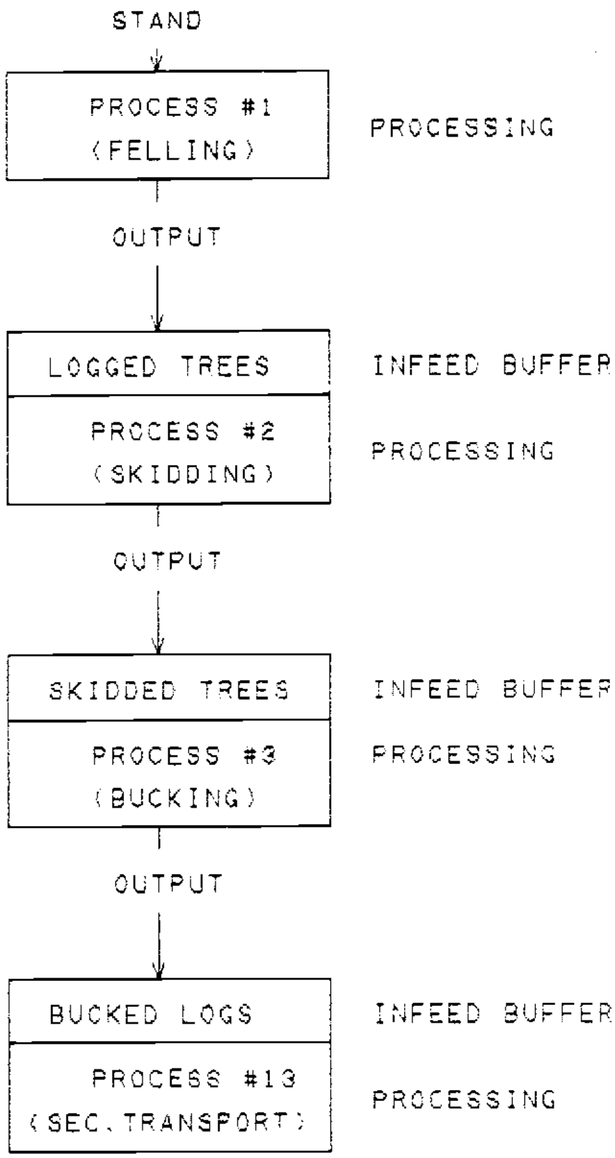
When the initial startup level for a process is reached, the assigned machines are activated by altering their respective resource capacities. The statistics for the process are initialized and the model then begins processing available inventory in the above described manner. Also entities are placed into the machine breakdown network if the modeling of machine breakdowns is desired. The machine breakdown network is a relatively simple matter. In a FORTRAN user function, the model assigns two values to the machine entity, one for the time between failures and the other for the repair time. These values are set according to the frequency distributions specified in the harvesting model. The entity is then delayed for the assigned time between failures. After this time has passed, the machine is seized and made unavailable for normal processing. The repair time is lapsed after which the seized machine resource is released, thus making it available to resume normal operation. The cycle is then started again by assigning the next pair of time values to the entity. When a process is finished, the respective machine entities in the machine breakdown network are disposed, freeing the SLAM processor of these entities.

If the ending condition of a process is reached, the model sets a flag accordingly to indicate this fact, calls a user function to calculate the final statistics, and gives an output of these statistics. Statistics are given in the sequence of completed processes. When the complete harvesting system is finished, the output user function is called one more time and the statistics for the whole system are compiled and given out (see section V.*B*.2). The end of simulation message is then displayed and the SLAM processor exits to the DOS prompt.

3. MODELING INVENTORIES

For modeling material buffers throughout the system and their restrictions and capabilities, the modeler is given several tools to analyze this harvesting system component. He can describe the inventory levels of the material input buffers of Processes 2 to 13 if desired. Since Process 1 feeds directly from the stand, there are no restrictions on the input buffer of this process. The description of inventory sizes is done by specifying the minimum size of the inventory required for this process or the maximum size the buffer/inventory can bear. Also a startup level which has to be reached after the inventory minimum or maximum has been triggered can be specified. Lastly the modeler can set an inventory level for the initial startup of the process, thus, describing starting conditions. Throughout the model the input buffer of one process is the output buffer of the previous process (see Figure 6).

Figure 6: INVENTORY BUFFER CONFIGURATION



During the simulation the model automatically checks the inventories of the active processes each time an inventory transaction is performed. If the minimum inventory level is reached, the machines of the current pro-cess are deactivated. If the maximum level is reached, the machines of the previous process are deactivated. Accordingly, the respective processes are reactivated when the startup limits are reached. When a process is deactivated, the machines currently engaged in a processing action still will finish their immediate job, not simply be preempted and stopped. This is done by altering the available resources allocated to the respective machine types. The model will collect statistics of inventory downtimes throughout the simulation and will show them in the simulation results.

Throughout the simulation a set of flags are set to indicate the inventory state (maximum reached, minimum reached etc.). These flags can be easily accessed and used to control the simulation if the model should be extended (ARRAY lines 15, 19, 20).

Also, the model keeps track of the inventories in transit, meaning that the inventory amount involved currently in processing activities. This is done to prevent premature ending of processes, that can occur if the previous process has an inventory of zero in its input buffer, but the machine carrying the last load is still engaged in processing. Therefore, inventory is still on its way to be processed by the next process. By checking the inventory in transit the program assures that no premature ending of processes can occur.

4. MODELING MACHINES

The simulation model can simulate up to 42 different types of machines. These machine types are divided between the thirteen process (see Appendix A. 5). For each of these 42 machine classes, the modeler can specify a different set of parameter values through the input user-interface. This set of parameters describe the machine capacity, processing times, machine breakdowns, costs and how many machines of each type are available (see section V.*A*.1.).

A total of approximately 90 machines can be handled by the model at any given point in time during the simulation. Since the model places for each machine one entity in the processing network and one additional entity in the machine breakdown network , it is possible to simulate up to 180 machines simultaneously if the modeling of machine breakdowns is omitted. Another route to increase the possible number of machines is to use the free space of approximately 2200 words in the network description to increase the number of entities preset with the LIMITS statement at the beginning of the network. This would yield another 200 entities or 100 additional machines.

In the following paragraphs, we will describe how the model handles different aspects of the simulation of machine actions. However, the machine types representing loading actions are handled somewhat differently than normal machines and are, therefore, described in more detail in section IV.*B*.10.

In order to model time delays due to machine actions, the model gives the user two principal choices: the first is to use the built-in routine to model time delays; the second is to write a FORTRAN user function and link it to the SLAM Execution Processor. As stated before, times are defined in a decimal format, where 1.0 hrs equals one hour.

To use FORTRAN user functions for modeling more complicated time delays requires a throughout understanding of FORTRAN, SLAM and the simulation model. A much easier way is to employ the built-in capabilities of the model by using the first modeling option. For each machine, the user has the choice to specify three different types of time delays. The values for these delays are specified in the input user-interface when the parameters for the respective machine are set.

The first type of built-in time delay is the time required per tree. When the model sets the load size for each machine run according to the specified distribution and machine capacity, the number of trees for the particular run is also stored in an attribute of the machine entity. The model then simply multiples the number of trees in the run times the time required per tree and delays the entity accordingly. Examples for this type of time delay are time required to fell a tree, average time to set a choker for a skyline, and time for chipping a tree.

The second type of time delay is the time per load. This is a constant time required by the process for each load. Examples for this type of delay are the average time a skidder operator needs to hook-up a skid load, time to load a self-loading truck, and time to prepare a truck for a hauling action.

The third built-in time delay models hauling times. The user specifies how much time one way of the haul requires. During the simulation the model will then delay the required time, adds the transported load to the inventory of the next process and performs the required inventory checks. Thus, the transported inventory is made available for the next process after a one way haul. The model will delay the entity a second time, simulating the haul back. Only then the resource attached to the entity is released and made accessible for the next processing cycle.

Examples for this type of time delay are easy to find: transporting logs to the sawmill, average skidding times and so on.

With the help of these three build-in time modeling alternatives, the user should be able to simulate a vast array of machines and processes. For additional examples of processing-time modeling see section VI.A. Example runs.

The SLAM network of LOGSIM already incorporates an interface for FORTRAN user functions, making it easy to use them if necessary. The user simply indicates that he wishes to use user functions to simulate the time delays for a certain process. This is done in the input user-interface when the parameters for the considered process are set. He then writes the required user function in which he assigns values to the machines used in this process. Since each machine entity carries the machine type and process number as an attribute value (see Appendix A. 2), it is quite simple to route the entities accordingly and distinguish between the different machine types used within the same process. The user function has to be appended to the already existing user functions (see Appendix B. 4). The addition has then to be indicated right at the beginning of the subroutine USERF, where the program jumps to the desired program label according to the value of XX(5). The values 1 to 99 are already reserved to indicate the jump address. The subroutine USERF has then to be recompiled and linked to the SLAM Execution Processor. When the simulation model is executed, the model checks automatically if the machine entity belongs to a process for which time delays are modeled with the help of a FORTRAN user function. The entity is then routed accordingly. Thus, the user is not required to alter the SLAM network to incorporate user functions for time delays. To ease the task of modeling constant time delays per load and hauling times, the program still will perform those time delays if specified for a certain machine, even if a FORTRAN user functions is used. Only the variable time per load is skipped by the network when using the user function option to model time delays. Therefore the input user-interface will still prompt the user for values for these time parameters.

At the beginning of the main processing routine, the actual load size of the current machine run is determined according to the specified distribution for this process and the load size capacity of the machine. As stated before, this is done by a FORTRAN written user function (see Appendix B. 4). In this user function, the model first determines if there is actually enough inventory for a full load according to the capacity of the machine. If there is not enough inventory available, the entity is routed back to the main processing routine and is sent to a waiting loop where the previously seized resources are released and the entity delayed according to the value of the "time delay parameter". This "time delay parameter" describes the time intervals that the model checks if inventory for processing is available. This parameter is set in the input user-interface right at the beginning when a new process is defined (see section V.A.1.a). From the above we see that the model will perform an actual processing activity only when a full load according to the machine capacity is available.

In the next step the user function performs a loop in which trees are generated according to the specified distribution. A uniform distributed random number between 0 and 100 is fetched from the SLAM processor. The program then looks up the matching percentage class of the material distribution for this number, sets the tree volume accordingly and adds this volume to the load size. The tree count for the load is raised by one and the loop repeated until the maximum load size is reached. Therefore, the actual load size will vary throughout the simulation as in reality where the next tree is simply too big to add to the load even through some capacity is still available. The entity is then routed back to the main processing routine in the network.

However, one special case is when the user wants to model the processing of one tree or work piece at a time as in a manual felling process using chainsaws or the debarking with a transportable rotary debarker. In this case, the user specifies a machine capacity of 99,999.0 cubic feet (cuft) when asked to set the load size for a given machine type in the input user-interface. The program then fetches the values for only one tree according to the specified distribution in the above described manner and proceeds.

Another special case is if a process is completed and the current entity represents the last run where only a rest inventory has to be processed, not yielding a full load size. The program detects this and assigns the remaining inventory as a load. It determines the number of trees and load size as described and flushes the rest inventory that does not yield the volume of a tree on top of the load, adding one additional tree to the load size. If the processing of a single tree was specified and the rest of the inventory is smaller than the volume of the largest tree possible, the load size is set to the remainder and one tree is processed additionally. Therefore, the model processes mathematically exact all inventory, leaving no remainders in the input buffers of the processes used.

Throughout the simulation, the model keeps track of the actual hours a machine is really active and accumulates those productive machine hours. Later these figures are used to determine machine utilizations and costs. The results are shown in the simulation results (see section V.B.2.).

5. PROCESS #1, FELLING

The first process the simulation model starts with is Process #1. This process will mostly model a felling operation in a harvesting system. However, it also could stand for any other work element if the partial modeling of a harvesting system is desired. For example, it could represent a primary transportation function if the simulation of a previously logged site is wanted. Another alternative is that the modeler could use the first process to simulate the first three work-elements of a harvesting system only describing the outgoing stream of inventory and modeling the following processes in greater detail. To provide the greatest possible flexibility, the user is also allowed to model a loading action by requesting the use of a loader when setting the parameters for this process in a harvesting model.

Up to four different machine types can be specified for this process to simulate machine activities (see Appendix A. 5) by setting machine types 1 to 4 active. This is done in the input user-interface where the modeler will be asked how many machines of each type he wants to employ. By setting this parameter greater than zero, the modeler activates the respective machine.

6. PROCESSES #2 - #10, NORMAL PROCESSES

Processes #2 to #10 are thought to be used to model the bulk of the work elements of a harvesting operation. They can stand for skidding, delimbing, bucking or a swinging process. The processes can be arranged in any desired order. For each of these processes, the modeler is able to use up to three different machine types with different machine capabilities and costs, see Appendix C. 2.

The behavior of the model when simulating such a process is described in IV.*B*.2..

7. PROCESS #11, SORTING

This process provides the modeler with the means to simulate the dividing of the material flow through the model into two separate branches. This makes it possible to simulate harvesting systems where two primary products like pulpwood and sawlogs are produced, which require different processing after the products have been sorted out.

Process #11 can be activated through the input user-interface. The modeler will be asked for the numbers of the following processes for each route, which can be any of the others, except of course Process #1. In both routes there is no limit on how many processes follow subsequently. The only restriction is that each branch ends either with Process #12 or Process #13. Later in the input user-interface the modeler specifies how much of the incoming materials stream is directed to each route by stating the desired percentages. As with any other process, minimum and maximum inventory sizes can also be specified, thus, completely modeling a sorting deck.

To simulate the sorting process, the user can optionally activate a loading device for this process by modeling the machine actions required for a sorting process. If Process #11 represents a log deck from which the following processes draw their input inventory, the use of a loader simply may be omitted. It is possible to simulate the partition of the material flow with or without time delay.

The model checks every time interval according to the value of the time delay parameter if new inventory has arrived at the input buffer of the process. It then calculates how much of the incoming inventory goes to each path in confirmation to the specified percentages for the individual routes. Internally, the model keeps track of how much inventory in the current input buffer belongs to each route changing the amounts dynamically throughout the simulation. The program then determines how much inventory it can route through the two different material flow paths, checking the current inventories of the following processes, and setting the amount to be routed through each path according to the inventory limits imposed on the following processes. The calculated amount is then transferred, with or without time delay, to the new input inventories calculated, and the cycle is started again.

Since the model always calculates the exact amount of inventory it can allocate through the different paths according to the maximum size of the following inventories, the subsequent input buffers will never be overloaded. The modeler should keep this in mind when analyzing the performance of a specific harvesting model in regard of the buffer sizes. If for subsequent processes the maximum inventory sizes are too small dimensioned, it will be reflected in the statistics of Process #11. They will cause either inventory downtime because the inventory is too high or an uncharacteristic high average inventory level/maximum inventory.

Process #11 is also a good example of how to incorporate a full processing function into the model without using the main processing routine. This process is a complete self-contained network within the third network, only using commonly accessible subroutines to check for inventory statuses and the end of the process. It is an example of how to write network submodels to simulate special machines or processes where the normal network does not provide sufficient modeling support. It also demonstrates the flexibility and capacity for expansions of the existing simulation model, thus, making it easy for the researcher or user to customize the existing framework to analyze more complex systems.

8. PROCESS #12, CHIPPING

Process 12 was modeled with the simulation of chipping operations in mind. As stated before, the in-woods processing of stems has increased rapidly with the growing share of second-growth forests in logging operations, thus, making this process a very important one.

The machine configuration modeled in this process is as followed: one main machine type, which requires a second machine type for machine actions. The second machine type may or may not require a third machine type when beginning processing itself. The real world machine configuration that was the model for this process is a chipper, the main machine type, which blows the chips into a chip van or chip trailer, the second machine type. To allow modeling of configurations where a chip trailer and towing truck combination is used, the third machine type has been introduced. In this case, the trailer is represented with the second machine type while the towing unit is modeled with the third type.

When defining Process #12 in the input user-interface, the primary or main machine type (machine type 37, see Appendix A. 5) is automatically set active making one machine available. The user is asked how many primary transporting devices he wants to employ, specifying the number of machines in the second machine type (machine type 39). He then has to specify how many secondary transporting devices he wants to use, the third machine type (machine type 40). The primary transporting device stands for the chip trailer, the secondary transporting device for the towing truck. If no towing truck is used, e.g. the chip van is one unit including towing device, the user simply sets the number of secondary transporting devices equal zero, thus not using this option. All machine parameters concerning the time delay caused by transporting action will be requested from the input user-interface when specifying the parameters for the primary transporting device, machine type 37. The model allows the user to specify different cost values for the primary and secondary transporting devices, that would make it possible to obtain fairly exact results on the cost structure of a given transporting system. A typical working cycle of process #12 is described in the next paragraph.

When starting a work cycle, the model seizes the main machine and a primary transporting device. If no primary transporting device is available, the entity is routed to the waiting loop and delayed until a primary device is serviceable. It then checks if a loading action is required and if so also seizes a loading device. The batchsize of the primary machine is then determined with the FORTRAN user function provided in the usual manner. The program performs the necessary inventory calculations, models the loading action if requested and continues with the machine actions of the main machine. After these time delays, the processed inventory is added internally to a buffer that represents the amount of material in the primary transportation device and the cycle for the main machine is started again. Therefore it is possible to specify different machine capacities for the main machine and the transportation devices. When the buffer for the transporter exceeds the amount specified as the capacity of the primary transporting device, an entity representing this device is released modeling the transporting action. The buffer is set to zero and a flag is set, that a new primary transportation device is to be seized by the entity representing the main machine. The model will seize a new primary transportation device only when the current one is fully loaded. The transportation entity then checks if a secondary transporting device is necessary. If a secondary device is necessary, it is seized and the process continues with the modeling of the time delays caused by transporting. The required inventory calculations are performed, the haul back is simulated, and the seized primary and secondary transporting devices are released making them available.

Since the model allows the user to specify the capacity of the main machine independent from the primary loading device, it is possible to model the processing of a single tree (machine capacity 99,999.0), little chunks representing a material input buffer at the chipper itself or a whole trailer load at once. However, when modeling little chunks, care should be taken that the capacity of the primary loading device is a multiple of these chunks or the chunk-size is set accordingly. Since the model only checks if the inventory in the primary loading device has reached a certain level and then advances the loading device, an error is introduced when adding the last chunk which might lead to a significant overloading. This results in an increase of transporting capacity which is actually not available. Generally when not using the single tree option at the main machine, we recommend that when the processing of whole trailer loads is modeled the capacity of the primary transporting device is set to its nominal capacity minus the largest possible tree size according to the used distribution. When the processing of little chunks was modeled, we recommend a primary transporting device capacity of nominal capacity minus half a chunk size. This will reduce possible errors and we feel that the resulting error is negligible.

If the user wants to model a harvesting system where the material stream is divided into two branches but does not want to simulate a chipping process, he can use process #12 as a finishing function by setting the processing times for machine #37 (the chipper) to zero, thus causing no time delay. The capacity of machine #37 should be set equal to the capacity of the transportation device used. The capacity of the primary transportation device, machine #39, should be set to the same amount minus the greatest possible tree size according to the material distribution used. By doing so, the model will behave just like the normal transporting function as described in the next section.

9. PROCESS #13, FINAL TRANSPORT

Process #13 models the transportation of the logs to the sawmill. It has the same structure of the transporting procedure as Process #12. The modeler can use a primary transporting device and an optionally secondary one. The primary device represents either a log truck including the towing unit or a log trailer while the secondary transporting device represents the towing unit in the later case. Machine type 41 represents the primary transporting device, machine type 42 the secondary one (see Appendix A. 5). As usual the user can model the loading of the transportation unit with a loading function.

The above described machine configuration is very flexible. The user can model transportation systems which for example can consist of four log trailers and two towing units. Each of these machine types can have a different cost structure, which is normally the case, thus, a good cost analysis is possible. Normal transportation configurations where trailer and towing truck are one unit can be simulated as well by simply omitting the secondary transporting device.

When starting a working cycle, the model behaves in the usual manner. It seizes a primary transportation device, checks if a loading function is required and if so seizes also a loading device. It then determines the load size of the current run, performs the inventory calculations and continues with the load function if one is required. The program then seizes the secondary transporting device, if one was activated, and delays the entity according to the specified times. The machine statistics are updated and the seized machine released, then the cycle starts over again.

Process #13 is, along with Process #12, the process that should stand at the end of the modeled harvesting configuration. The reason for this is to properly enable the simulation model to detect the end of the harvesting process. However, by simply labeling this process and its machines accordingly, the analyst is able to model any other function if desired. The only limitation in this case is that only one machine type can be used. If this limits the modeling process, the analyst uses one of the normal processes to model the desired function and specifies a simple transporting function with no time delays and no costs, a dummy process. The produced simulation statistics will show results without any significant influence of this dummy process.

10. LOADING DEVICES

For each of the thirteen processes, the analyst is able to model the use of a loading device to feed the main machine with material. The activating of these loading devices is done in the input user-interface when the processes are specified in more detail. The user is asked which loader type he wants to use for a process. He can activate one loader type per process and has the choice between five machine types, machine type 32 to 36 (see Appendix A. 5). It is possible that the different processes share the same loading device type throughout the simulation, which is quite common in real-world operations. Right after the specification of the processes, the user is asked to input how many machines of each activated loader types are available. Later the characteristics of the loader types are set when the parameters of all machines are entered.

The user can specify machine capacities, delay times, machine costs and machine breakdown distributions for the loading machines just as for any other machine. However, the model handles loading actions a little bit differently than normal machine actions. Instead of determining for each run of the loader the batchsize and the number of trees processed like the main machine with the FORTRAN user function, the model uses the batchsize numbers of the main machine. It simply divides the load size of the main machine by the capacity of the loader, thus calculating the number of runs needed to load the main machine. The program multiplies this number with the specified time per load. It then adds this to the time for one way hauling and the time per tree times the number of trees. Thus, the entire processing time needed to load the main machine is calculated. It then delays the entity of the main machine accordingly, records the machine time for the loading device and continues the normal processing cycle.

When executing a loading function, the simulation model uses the entity of the main machine to control the flow of the loader through the model. Therefore, for each loader, only one entity is created in the network, which is needed for the machine breakdown network. If the simulation of breakdowns is omitted, no entity representing a loader exists within the network during a simulation run.

Since loaders can be shared throughout the simulation by different processes, statistics compiled for loading actions are based on the entire simulated harvesting time (see also V.*B*.2., output front end). Thus, the statistics for the scheduled hours and the given utilizations have the whole harvesting time as basis. This also means that loaders are not included in the statistics of the processes where they have been used. At the end of the simulation the, output front-end compiles separate statistics for the loading devices and shows them in the simulation report as a separate topic (see V.*B*.2.c). If a loader type is only used by one process, the user can recalculate the statistics accordingly by hand if desired, since all necessary numbers are given in the output report.

V. FRONT-END DESIGN

The Front-ends or user-interfaces provided with the model where developed to ease the task of modeling for the analyst. They should enable analysts with no prior knowledge of SLAM as a simulation language to use the developed model of the harvesting process as an analyzing tool. However, a knowledge of the principles of simulation should be a prerequisite when using any kind of simulation as a management tool.

The user-interfaces are divided into two major parts, the input front-end and the output front-end. The input front-end is used to enter the different parameters of a harvesting configuration into the model, while the output front-end calculates the statistics during the simulation and presents the results in the simulation report. In the following section, these two user interfaces are presented.

A. INPUT FRONT-END

The input front-end, written using the Microsoft FORTRAN compiler version 3.31, requires approximately 200 K bytes of available memory to run on the IBM-PC. The program is invoked at the DOS prompt by typing "FRONTEND.EXE". It consists of three parts:

1) A module to define a new harvesting model.
2) A module to edit an existing harvesting model.
3) A module to print out an existing harvesting model for documentation purposes.

The program is completely menu-driven and the user is prompted for each input. When the program is started, a greeting message is displayed along with the main menu from where the user can access the different modules. A listing of an example session with the input user-interface is given in Appendix C. 2 - C. 4. A figure describing the file structure of the FORTRAN programs is also given in Appendix D. 1.

1. DEFINING A NEW HARVESTING MODEL

The module to define a new harvesting model is carried out by choosing the menu option 1 in the main menu of the LOGSIM input user-interface. After choosing this option the program shows the opening screen of this program module and verifies that the user wants to continue with the defining of a harvesting model. If not, the program jumps back to the main menu so the user can choose another option. The input of a harvesting system is structured into seven phases:

PHASE 1:

Specification of the general harvesting parameters like file-name of the model, amount to be harvested and value of the time delay parameter.

PHASE 2:

Definition of the material flow through the harvesting system.

PHASE 3:

Entering of the material frequency distributions used.

PHASE 4:

Specification of the process parameters like optional name of process,inventory levels, material distribution used etc.

PHASE 5:

Input of how many machines per machine type are used.

PHASE 6:

Definition of the machine parameters like processing times, capacity and costs.

PHASE 7:

Specification of the machine breakdown distributions.

At the start of each phase an introduction screen is given which tells the user what he has to enter next. Default values are given at the input prompt in square brackets throughout the program. To use them the modeler needs only to press the ENTER key.

Besides the entering of the necessary values for the simulation parameters, the user also can enter optional labels and descriptions for machines and processes. This information is used later on in the simulation results and the harvesting system description to make those outputs more readable and easier to understand.

a) PHASE ONE, GENERAL PARAMETERS

In phase one the user inputs first a filename under which the model yet to be entered will be stored. This filename should follow the DOS conventions for filenames and be a unique name to identify a harvesting system according to any scheme you choose. Normally the naming of a file consists of two parts: a *filename* and a filename *extension*. The *filename* and its *extension* are separated by a period. A *filename* can be from 1 to 8 characters long. The filename *extension* is optional, but recommended, and can be from 1 to 3 characters in length.

The next item to be entered is the amount of wood to be harvested. This number can have a range from 1 to 9,999,998 cuft with no decimal digits. Care should be taken to enter the decimal point when entering this number.

The third and last item to specify is the value of the time delay parameter. This number describes which time interval the model will use to check the inventory buffers of the activated processes if enough inventory is available to process a machine run. The time delay parameter can have a range from 0.0001 to 999.0 decimal hours. However, if the value was chosen either too big or too small the model will obviously produce either unreliable simulation results or needs an excessive amount of computer runtime. We recommend 0.1 hrs, which equals 6 real time minutes, or 0.01 hrs, equal 36 real time seconds, as values if the modeler is not sure about the real time value of this parameter.

b) PHASE TWO, MATERIAL FLOW

Phase two defines the material that flows through the model, e.g. the sequence of processes. The program will ask in sequence for the incoming origin and the outgoing destination of the material stream for each of the thirteen processes. If a process is not used, simply press ENTER on both questions and the process will not be activated. For Process #11, sorting, the program will ask for the outgoing destination route 1 and the outgoing destination route 2 to divert the material stream into two branches.

After the user has given all the information, a table of these numbers will be displayed so the modeler can check if they are correct. At the bottom of the table a message is displayed prompting the user to indicate if all values are correct. If answered negative, the program jumps back to the beginning of Phase two and starts again. If answered positive, the model performs a check if all numbers match logically. When the program finds the table not correct it displays an error message indicating where it found the first mismatch and returns to the beginning of Phase two after the ENTER key has been pressed.

c) PHASE THREE, MATERIAL DISTRIBUTIONS

The third phase is used to specify the material frequency distributions to describe the trees, logs and pulpwood pieces which are handled by the machines. Up to four frequency distributions, each with up to ten frequency classes can be specified.

These distributions are based on the volume in cubic feet of the respective product. However, the modeler can use any other measurement units or any other parameter suitable to describe the material. Care should be taken to describe the machine capacities in the same units since these distributions are later used to determine the number of pieces in a load and the actual load size per machine run.

The program starts out with the usual introduction screen. It then asks for an optional name for the first distribution, which can be up to 20 characters long. For each of the ten possible frequency classes the cumulative relative frequency and the volume in cubic feet are then requested. Cumulative relative frequency distribution means that the frequency percentages for the different classes build an increasing sequence, ending with 100.00 %. Throughout the entering process the program checks that each frequency number is larger than the previous one and that the last class specified ends with the value 100.00. The range for these numbers are from 0.01 % to 100.00 %. The range for the values of the volumes is 00000.01 to 99,999.99 cubic feet. Care should be taken when compiling the frequency distributions so that the lowest class represents the smallest piece size with the following classes specifying the piece size in increasing order. When the program encounters a class with 100.00 % cumulative frequency or the tenth class is entered, it stops prompting for new values and a table of the just entered distribution is displayed. The user is given the option to accept the entered values or to start over again. If the distribution is accepted and no errors are found, the input cycle for the next distribution is started.

To omit any of the four distributions, the modeler simply presses ENTER when the name of the distribution is asked and also ENTER for the first cumulative relative frequency and the first volume information. However, at least one frequency distribution with one valid frequency class has to be specified. The program will issue an error message if this is not the case and returns at the beginning of Phase three.

d) PHASE FOUR, PROCESS PARAMETERS

In Phase four the processes activated with Phase one are defined in more detail. After a process has been defined, the user as usual gets a table of the just entered values and the option to accept them or to enter them over again.

For each of the active processes, the program automatically asks for values of the following parameters.

The first parameter to enter is an optional label for the process, again up to 20 characters long.

The second prompt asks the user to name the material frequency distribution he wants to use to model this process. An integer number from 1 to 4 can be entered, which represents the distribution number. The program then checks if the specified distribution was set active during Phase 3 and if not, issues an error message with the request to enter the distribution number again. Otherwise, the value will be accepted and the program continues.

The third parameter is the startup inventory level. This is the inventory level needed for the first initial start of a process. It can have a range from 000000.1 to 999,999.9 cubic feet. The program will check that this number is less than the total amount to be harvested. However, when dividing the material stream into two branches care should be taken that the values for this parameter can be achieved during the simulation according to the specified percentages. The user interface uses a default value of 1.0 cuft for the initial startup-inventory when ENTER is pressed. The initial startup-level is also used by the program to determine the point in time when the process actually started (see also V.*B*.2.a)). To get meaningful simulation results we therefore recommend that the user sets this parameter to at least the largest load size of the machine types used in the respective process. Otherwise the program will start the process despite the fact that not enough inventory for a machine run is available.

Next, the minimum infeed inventory level has to be entered. This parameter defines the level of inventory to be maintained in the input buffer throughout the simulation. A range from 000000.0 to 999,999.9 can be specified while the default value is 0.0. The default means that all inventory can be used for processing.

The fifth parameter sets the inventory level to start up again if the minimum inventory level has been reached. Numbers from 000000.0 to 999,999.9 are accepted, with a default value of 0.0 . The program cross checks that the entered value is equal or greater than the minimum level specified previously and will report an error if this is not the case.

Next the maximum size of the inventory buffer has to be specified with a number range of 000000.1 to 999,999.9. The default is 999,999.9, which represents an unlimited size of the buffer. The program checks automatically that the maximum is greater than the minimum inventory level and greater than the startup level for the inventory minimum. When specifying this parameter, care should be taken to set this number at least as large as the largest capacity of the machines employed in this process, to ensure that enough inventory for a machine run is available.

The seventh parameter to be entered is the startup level to which the inventory of the current process has to drop after the maximum inventory has been reached so that the previous process can be reactivated. Again the range for the values is 000000.1 to 999,999.9 with a default of 999,999.9. Checks are performed to insure that the entered value is at least greater then the minimum inventory level and less than the maximum level.

The eighth prompt asks the user to indicate if he wants to employ a loading function for the modeling of the process. By entering the number of the respective loader type, an integer between 32 and 36, the required loader is set active. When no loading function is necessary, the user simply should press ENTER to use the default of 0 which indicates that no loader is used.

The last parameter requested by the program to describe a process is if the modeler wants to use his own FORTRAN user functions to model the time delays machine actions require or use the built-in modeling functions. The default is 0, which means the built-in functions are used. If the user enters a value of 1, the simulation model will use FORTRAN user functions supplied by the user during the simulation to model this process.

e) PHASE FIVE, NUMBER OF MACHINES

The fifth phase defines, for each of the processes, which machine types will be used and how many machines of each machine type will be available throughout the simulation.

However, if any processes were defined in the previous phase that use a loading function, the program first will prompt to specify how many machines for each activated loader type are available. It will accept integer values from 0 to 80. If an activated loader type is set to 0, the program will display an error message and prompts again for the number of machines available. As usual the user will be presented with a table of the entered values with the option to re-enter them if desired.

The input interface will then continue in sequence of the activated processes to prompt the user for each of the available machine types per process and how many machines he wants to employ during the simulation run. The prompts will already show the machine type number according to Appendix A. 5.

If process #11 was activated the program will ask the user to specify the percent of the incoming material stream that goes to route 1 and how much goes to route 2. It will accept values between 0.01 and 99.99 percent and checks that the sum of both percentages equals 100.00 %.

When process #12 is utilized, the user is prompted to indicate how many primary and secondary transporting devices he wants to use. The chipper, machine type 37, is automatically activated by the program and set to 1 available machine.

For process #13, final transportation, the interface again prompts for the number of primary and secondary transporting devices.

The program checks that for each of the utilized processes at least one active machine exists. In case of process #12 and #13 it verifies that at least one primary transportation unit is available. After each process the user can inspect on screen the values just entered and is given opportunity to change them.

f) PHASE SIX, MACHINE PARAMETERS

In this phase the actual specification of the machine types takes place. Again the program will ask the user in sequence of the machine types for several parameter values. After one machine type has been declared, the entered values are displayed so that the user can reenter them if desired.

First the modeler can input an optional name for the machine type, up to 20 characters long. The second prompt asks for the average processing time per tree, which can obtain a value from 000.0000 to 999.999. The default is 0. Then the fixed constant time per load is requested, which can have the same range as the average processing time and has also the same default. The forth prompt asks the fixed constant time of one way hauling.

The fifth parameter defines the capacity of the respective machine type. The allowable range for this number is 00000.01 to 99,999.99 cubic feet; the default is 1.0 cuft. However, care should be taken that this value is at least as large as the largest tree volume value specified in the material frequency distribution used for the process to which the machine type belongs. If the analyst wants to model a machine which only processes one tree at a time such as certain types of delimbing and debarking machines, he simply has to enter a machine capacity of 99,999 cubic feet. The simulation model will then behave accordingly during run time and only assign one tree per machine cycle with a volume from the material frequency distribution.

The next two parameters are concerned with the cost structure of the machine. The first one sets the fixed cost per scheduled hour, the second one the variable cost per machine hour. These values are later during the simulation run used to compile cost related statistics. For an explanation of what a scheduled hour and a machine hour means please see chapter V.B.2., the output front end design. All the terms related to the simulation results are described there. The input range for both cost parameters is 00000.00 to 99,999.99 with a default of 0.0. Therefore, when using the default option, no cost statistics are produced by the simulation model.

g) PHASE SEVEN, MACHINE BREAKDOWNS

The seventh phase completes the description of the machine types by specifying the breakdown behavior of the machine type. The analyst has to enter one frequency distribution for the times between failures and one for the repair times. If the modeling of machine breakdowns is not desired, it can be omitted by using the default values of 0 for the first frequency class of the frequency distribution for times between failures. The program will then skip the entering of repair times for this machine and will display a message that this particular distribution is not used. As usual, the program displays after each machine the entered values to give the user the opportunity to make changes. The conventions to enter the frequency distributions for the machine breakdowns are just like the ones for the material frequency distributions, and may be read in section

V.A.1.c) if desired. The times between failures and repair times have a range from 00000.01 to 99,999.99 and should be entered in the usual decimal time format. The program performs checks to ensure that the values are entered properly and will prompt the user with a message if a fault is detected.

After Phase 7 has been completed the program asks the user if the just defined harvesting model should be saved or not. To use the harvesting model for simulation purposes with the developed SLAM network it must be saved! The model uses the filename entered in Phase 1 to write the file to the default mass storage media. Before doing so it will check if a file with the same filename exists. When this is the case it prompts the user to enter a new filename for the harvesting model. After the program has stored the model successfully, it displays a message that it has done so and will return to the main menu after the user presses the ENTER key. By choosing the appropriate modules the user can then either edit or print the harvesting model.

2. PRINT A HARVESTING MODEL

By choosing the menu option 2 at the main menu prompt the user can route any previously defined harvesting model to a printer. The program will prompt the analyst if he wants to continue and if so asks for the filename of the harvesting model to be retrieved. The model will check if the file is in the current directory and will load it. If no file under the specified filename is found an error message is displayed and the user prompted for a substitute filename. When the model has been successfully fetched, a message is displayed accordingly.

The user is given the choice either to display the harvesting model on screen or to route the output to a printer. Example outputs can be seen in Appendix E. 2, E. 5, E. 8, and E. 11. An example of the dialog between user and computer is given in Appendix C. 3.

These printouts can be used to document the harvesting models and will show the entered data in an easy to understand and well organized manner.

After the program has produced the desired output it returns back directly to the main menu to allow the user the continuation of the program.

3. EDITING AN EXISTING HARVESTING MODEL

The last module in the input user-interface can be used to modify an existing harvesting model. It is invoked by entering the number 3 at the input prompt of the main menu. After the desired file has been retrieved, a menu with seven choices is displayed from which the user can edit all harvesting system parameters. The only exception is that the material flow, the process configuration, can't be modified. If a different process configuration is desired a new harvesting model has to be entered. An example run of this front-end module is given in Appendix C. 5.

When the user has made his choice from the modify menu the program will first prompt for the identification number of the desired machine, process or distribution. For the material distributions and processes, it will indicate which of them are currently activated to give some assistance to the user. If the user enters a zero at the prompt, that is the default, the program jumps back to the modify menu.

After the identification number has been entered, the user-interface displays the current values on screen in tabular form and asks if the user actually wishes to continue with the editing process, thus, entering new values. If the modeler does not want to continue, the program jumps back to the previous menu so that the next identification number can be entered. This is the default set by the program.

If the user continues, the program will prompt him for the new parameter values. The same value ranges, defaults, and restrictions apply for each of the new parameters as described earlier in the section V.*A*.1., definition of a new harvesting model. The program will perform the required cross checks to prevent mistakes and will display error messages if it detects one.

After the values have been entered an updated table of the values is displayed so the user can check his work. Again he is asked if he wants to continue editing, thus, changing the values. If not, the program jumps back to the input prompt for the identification number as explained previously.

When the user is done with the editing, the modified harvesting system must be saved. This is done by choosing option 6 in the modify menu. The program will asked if it should save the file. When answered positively, it checks if a file with the same filename already exists. If this is the case the program will display an error message. The user is prompted to indicate if he wishes either to enter a new filename or to overwrite the old file. When a new filename is entered, the same check is performed again. If no matching filename is found the edited harvesting model is saved under the new filename, otherwise the user is prompted again with the error message. When choosing the overwrite option, the values of the old file will be unrecoverably lost.

When the analyst wants to add a loading function to a process, he first has to check if the desired loader type is activated, e.g. a positive number of machines have been specified for this machine type previously. If this is not the case the user can set this machine type active by choosing the menu option 4 and enter a number greater then zero if prompted for the initial number of machines. He then should enter the other machine parameters and a machine breakdown frequency distribution if desired. This applies also for the activation of all other machine types. If the deactivation of a machine type is wanted, the initial number of machines has to be set to zero, therefore, making them unavailable. Simulation results will only be generated for activated machines in activated processes.

B. OUTPUT FRONT-END

1. SIMULATING A HARVESTING SYSTEM

The output front-end was developed to provide the user with easy to read output of the simulation results. It is integrated into the FORTRAN user functions that builds in conjunction with the initialization subroutines a customized SLAM Execution Processor that performs the actual simulation of a harvesting system.

This customized SLAM Execution Processor is invoked from the DOS prompt by entering <u>LOGSIM.EXE</u>. The program will then be loaded and executed. For an example session with the customized execution processor see Appendix C. 1. The SLAM Processor will ask first for the filename of the network model, which is <u>HARVEST.TRA</u>. After that the user is prompted for the filename of the harvesting system he wants to simulate and to which output device the simulation results should be routed. Simulation results can be routed either to the screen only or to the screen and the attached line printer. Then the number of simulation runs to be performed has to be entered.

The preset maximum number of simulation runs is 10, the default used by the program is only 1 run. Between each of the simulation runs, the SLAM Processor clears all statistical arrays and variables, initializes the internal filing system and, therefore, re-initialized the whole simulation system. Only the seeds for the random number streams are not re-initialized to provide different starting seeds for each simulation run.
The complete harvesting system model is read in again and the next run is performed. Therefore, the program will perform multiple runs of a harvesting model but does not require any actions by the user between runs to re-start the simulation.

During the simulation, the program displays the current simulation run, the total amount harvested so far, the current real time and the simulated time. This is done to provide the user with some means of control for models which require excessive simulation time.

When done with the simulation, the customized SLAM processor will return to the DOS prompt, from there the user can continue his computing session in the usual manner.

2. SIMULATION RESULTS

When the program detects the end of a process during the simulation, it will calculate the simulation statistics for this process and present them. Therefore, the simulation results are given in the order of finished processes. At the end of each simulation run when the complete harvesting system is done, the statistics for loading functions and a summary statistic for all processes are compiled and presented.

Generally, the simulation result output can be divided into the following sections:

- A header, describing which harvesting model was used. Computer time, computer date, and the number of the current simulation run to identify the output.
- The simulation results for a process. These results consists of two parts. The first one is concerned with the performance of the process overall, incorporating all active machine types for this process except loading devices. The second one is
- The results for each of the activated machine types for a process.
- The performance of the loader devices, if any were activated.
- The complete harvesting system statistics

In the following sections we will define what each of the compiled numbers means and for what it stands for.

a) PROCESS STATISTICS

For each process, the customized SLAM processor produces the following statistical numbers. Note, however, that these numbers do not include any loading devices the process may have used. Since a loading device can be shared by multiple processes throughout the harvesting system, the program will compile statistics for loaders separate.

If process #11 was used in the simulated harvesting configuration, the program will compile process statistics for this process just as if it were for a normal process. However, no statistics for the sum of scheduled hours, the sum of machine breakdown hours , and the machine utilizations are given since the employed loader type could be used by other processes as well.

- Time begin of Process. [1]

  This is the time recorded by the model when the initial startup inventory level has been reached (see also V.A.1.d)).

- Time end of Process. [2]

  Time the process is finished and all machines employed in this process have finished their tasks.

- Duration of process. [3]

  The calculated amount of time a process was active:

  [3] = [2] - [1]

- Time inventory too low. [4]

  Cumulated time the input inventory buffer of the respective process was below the specified inventory minimum.

- Time inventory too high. [5]

  Cumulated time the input inventory buffer of the respective process was above the specified inventory maximum.

- % Inventory downtime. [6]

  This number represents the portion of inventory downtime in relation to the duration of the process in percent.

  [6] = (([4] + [5]) ÷ [3]) * 100

- Total # of machines. [7]

  The sum of all machines employed for processing in this process.

$$[7] = \sum_{i}^{j} [22]_i$$

- Sum scheduled hours. [8]

  The total sum of machine hours scheduled for this process (except loading machines).

$$[8] = [7] * [3]$$

- Sum machine breakdown hours. [9]

  Total sum of machine breakdown hours recorded for all machines types involved with this process.

$$[9] = \sum_{i}^{j} [24]_i$$

- Sum productive hours. [10]

  Is the sum of all recorded time delays caused by processing actions for all machines used by this process.

$$[10] = \sum_{i}^{j} [25]_i$$

- % Net utilization machines. [11]

  The portion of time machines were really processing in relation to the sum of scheduled hours.

$$[11] = ([10] \div [8]) * 100$$

- % Gross utilization machines. [12]

  The same as [11], only including machine breakdown hours. This number was included because often machine breakdown is a parameter that cannot be influenced in the real world.

This number represents the portion of time machines are generally "busy" doing something.

[12] = (([10] + [9]) ÷ [8]) * 100

- Average inventory. [13]

  The average inventory level of the input buffer of the respective process for the simulated time, calculated by the normal formula for statistics based on observations.

$$[13] = X_n = \frac{\sum_{i=1}^{n} X_n}{n}$$

- Maximum inventory. [14]

  The observed maximum value for the input buffer of the process.

- Minimum inventory. [15]

  The observed minimum value for the input buffer of the process.

- Standard deviation inventory. [16]

  Standard deviation of the observed inventory buffer values.

$$[16] = S_x = (M \div (n*(n-1)))$$

$$\text{where} \quad M = n * \sum x^2 - (\sum x)^2$$

- Number of observations inventory. [17]

  The number of observations made during the simulation run for the size of the inventory buffer of the respective process.

- Sum units processed. [18]

  How much material has been processed by the machines of this process.

- Sum cost of process. [19]

  The sum of fixed and variable costs of all machines employed by the process.

  $$[19] = \sum_{i}^{j} ([26]_i * [22]_i)$$

- Cost per unit. [20]

  The cost to process one unit through this process.

  $$[20] = [19] \div [18]$$

- Cost per scheduled hour. [21]

  The cost per scheduled hour for the machine configuration of this process.

  $$[21] = [19] \div [8]$$

b) MACHINE STATISTICS

For each machine type employed by a process the program calculates statistics upon this machine type and will show them after the summary statistics for the process. These statistics are as followed:

- Total # of machines. [22]

  The number of machines set active for this machine type.

- Sum scheduled hours. [23]

  The sum of scheduled machine hours for this machine type.

  $$[23] = [22] * [3]$$

- Sum machine breakdown hours. [24]

  The accumulated machine breakdown hours for this machine type.

- Sum productive hours. [25]

  The accumulated machine hours where the machines actually processed material.

- Cost per machine. [26]

  The cost share for one machine of this machine type on the total process costs.

  $$(26) = ([23]_i * C_{fixed\ i}) + ([25]_i * C_{variable\ i})$$

- Cost per scheduled hour. [27]

  Actual cost of one machine of this machine type for one scheduled hour.

  $$[27] = [26] \div [23]$$

- % Net utilization machine. [28]

  The percentage of time machines of this machine type were processing inventory in relation to the sum of scheduled hours.

  $$[28] = ([25] \div [23]) * 100$$

- % Gross utilization machine. [29]

  The same as [28] only including machine breakdown hours. This number represents the portion of time machines of this machine type have been generally "busy".

  $$[29] = (([25] + [24]) \div [23]) * 100$$

c) LOADER STATISTICS

Statistics for the loading functions are generally the same as described in V.*B*.2.a) and V.*B*.2.b). The only difference is that for formulas incorporating time the complete harvesting time for the simulated system is used [49].

Statistics for the complete loading process are:

- Total # of machines. [30]

  Sum of all activated machines used by all loader types.

$$[30] = \sum_{i}^{j} [40]_i$$

- Sum scheduled hours. [31]

  Sum of all accumulated machine hours scheduled for loading functions.

  [31] = [30] * [58]

- Sum machine breakdown hours. [32]

  Cumulated machine breakdown hours of all loading devices.

$$[32] = \sum_{i}^{j} [42]_i$$

- Sum productive hours. [33]

  Sum of all productive hours of loaders.

$$[33] = \sum_{i}^{j} [43]_i$$

- % Net utilization machines. [34]

  The percentage of time the loaders were actually engaged in loading actions.

  [34] = ([33] ÷ [31]) * 100

- % Gross utilization machines. [35]

  The same as [34] only including machine breakdown hours as active time.

  [35] = (([32] + [33]) ÷ [31]) * 100

- Sum units processed. [36]

  Sum of material processed by loaders.

- Sum cost of process. [37]

  Sum cost of loading functions.

$$[37] = \sum_{i}^{j} [44]_i * [40]_i)$$

- Cost per unit. [38]

  Cost of loading for one processed material unit.

  $$[38] = [37] \div [36]$$

- Cost per scheduled hour. [39]

  Cost of loading actions per scheduled hour.

  $$[39] = [37] \div [31]$$

The statistics compiled for the individual loader machine types include:

- Total # of machines. [40]

  The number of machines set active for this machine type.

- Sum scheduled hours. [41]

  The sum of scheduled machine hours for this machine type.

  $$[41] = [40] * [58]$$

- Sum machine breakdown hours. [42]

  The accumulated machine breakdown hours for this machine type.

- Sum productive hours. [43]

  The accumulated machine hours where the machine was actually processing material.

- Cost per machine. [44]

  The cost share for one machine of this machine type on the total loading costs.

  $$(44) = ([41]_i * C_{fixed\ i}) + ([43]_i * C_{variable\ i})$$

- Cost per scheduled hour. [45]

  Actual cost of one machine of this machine type for one scheduled hour.

  [45] = [44] ÷ [41]

- % Net utilization machine. [46]

  The portion of time machines of this machine type were processing material in relation to the sum of scheduled hours.

  [46] = ([43] ÷ [41]) * 100

- % Gross utilization machine. [47]

  The same as [46] only including machine breakdown hours. This number represents the percentage of time machines of this machine type have been generally "busy".

  [47] = (([42] + [43]) ÷ [41]) * 100

d) COMPLETE HARVESTING SYSTEM STATISTICS

The statistics compiled for the complete simulated harvesting system give an overview of the overall performance of the modeled harvesting configuration. It is the last item in the simulation results output. The statistics for the complete harvesting configuration include:

- Computer time start simulation. [48]

  The real-time date and time when the computer started with the simulation of the current run.

- Computer time end simulation. [49]

  The real-time date and time when the computer ended the simulation of the current simulation run. The figures [48] and [49] were included in the output to give the analyst the opportunity to time the use of computing equipment. These figures can be used later for fee calculations if desired.

- Begin of harvesting. [51]

  Simulated time when the harvesting process began.

- Total # of machines. [52]

  Sum of all machines in all machine types used for the harvesting model, includes loaders.

$$[52] = \sum_{i=1}^{j=13} [7]_i + [30]$$

- Sum scheduled hours. [53]

  Sum of all scheduled machine hours for all activated machine types.

$$[53] = \sum_{i=1}^{j=42} [23]_i + [31]$$

- Sum machine breakdown hours. [54]

  Cumulated machine breakdown hours for all activated machine types.

$$[54] = \sum_{i=1}^{j=42} [24]_i + [32]$$

- Sum productive hours. [55]

  Total sum of time spent actually processing material.

$$[55] = \sum_{i=1}^{j=42} [25]_i + [33]$$

- % Net utilization machines. [56]

  The share of productive hours on scheduled hours in percent. This figure represents the overall efficiency of the simulated harvesting configuration.

[56] = ([55] ÷ [53]) * 100

- % Gross utilization machines. [57]

  The same as [56] only including machine breakdown hours.

  [57] = (([54] + [55]) ÷ [53]) * 100

- End of harvesting. [58]

  Total simulated time it took to complete the whole harvesting process for the specified harvesting configuration.

- Sum of units harvested. [59]

  Total amount of material processed.

- Sum cost of system. [60]

  Total cost for the specified harvesting model including loading actions.

$$[60] = \sum_{i=1}^{j=13} [19]_i + [37]$$

- Sum cost per unit. [61]

  Cost of one unit material after it has been processed through the whole harvesting system. This number is only correct if process #11, the sort, is not used.

  [61] = [60] ÷ [59]

- Cost per system hour. [62]

  Cost of one hour for the simulated harvesting system configuration.

  [62] = [60] ÷ [58]

With the provided simulation results, the modeler should be able to thoroughly analyze the performance and cost structure of any desired harvesting configuration.

VI. RESULTS

A. EXAMPLE RUNS

To verify the correct functioning of the simulation model extensive test runs have been performed. In these test runs the behavior of the modules and mechanisms employed to control the simulation was examined by using test data that simulated the different situations possible in a harvesting system. The results of the test runs were then compared with manual calculations.

Also four complete harvesting systems (stump-to-mill) were simulated. The production and cost information used were provided by Don Schuh, Research Assistant, Department of Forest Engineering, Oregon State University. Each of these four harvesting systems uses a different machine configuration with different capabilities. The systems are outlined briefly below:

1) Traditional manual sawlog operation:

   Manual felling, delimbing, and bucking. Cable skidder for primary transport. Self-loading highway log truck for secondary transport.

2) Contemporary mechanized sawlog operation:

   Felling by swing-boom feller-buncher. Primary transportation by grapple skidder. Delimbing and bucking with a Hahn Harvester. Swing-boom loader for decking and loading and a highway log truck for secondary transport.

3) Potential mechanized sawlog system:

   Felling and primary transport by TJ Clambunk Skidder with sawhead. Delimbing and bucking done by a grapple processor. Swing-boom loader for loading and a highway log truck for secondary transportation.

4) Mechanized pulpwood/sawlog operation:

   Felling with swing-boom feller-buncher. Primary transport with TJ Clambunk Skidder. Swing-boom log loader for loading and a set-out tractor-trailer combination for swinging to the central site. Here the material stream is divided into the two products. The chip processing is done with a multistem delimber-debarker machine feeding into a Morbark Model 22 chipper. The chips are blown directly into a chip trailer that needs a separate towing truck for the haul to the mill.

   The sawlogs are directly hauled to the mill by a tractor-trailer combination. The delimber-debarker and the log trailer both use the same swing-boom log-loader for loading actions.

All four harvesting models were used to harvest the same amount of wood for comparison reasons. In Appendix E the printouts for the different harvesting model configurations and the simulation results for each harvesting system are given. To ease the task of entering the required information with the input user-interface we recommend that the user may draw a schematic flowchart of the harvesting configuration. Examples of how such a flowchart may look like can be seen in Appendix E. 1, E. 4, E. 7, and E. 10. These flowcharts give an immediate overview of the processing configuration.

Table 1 summarizes the simulation results. As expected, the manual sawlog operation used the most time to finishing the harvesting process.

Table 1: Simulation results

Parameter	System 1	System 2	System 3	System 4
Duration hrs	3116.75	196.12	259.15	175.61
% Net. Utiliz.	74.61	63.65	47.76	68.97
% Gross Utiliz.	84.09	69.44	57.16	71.22
Cost of system	335893.40	83709.98	78280.74	119127.50
Cost/unit	1.31	.33	.31	.47
Cost/system.hr.	107.77	426.82	302.06	679.38
Runtime hrs.	1.13	.41	.36	.66

The systems 1 to 3 can be compared together, while system 4 has to be examined separate because it uses different material distributions due to the two products in the system.

From table 1, we can see that system 1, as expected, needs the most time to accomplish the task with a duration of 3116.75 decimal hours. The clear winner is configuration three in terms of costs. However, if the shortest duration of the harvesting process is desired to free machines for the next task, then system 2 represents an alternative with a slightly higher cost per unit of 2 cents. Under this view, we compare the performance of these harvesting systems in regard to a given stand and logging environment.

We could also examine each harvesting system on its own and see where we can improve. For example, system 3 has a net utilization of only 47.76 %. By examining the simulation results of system 2, (see Appendix E. 9), we see that the gross utilization in process #2, skidding, was only 39.43 %. This suggests that instead of two Grapple-skidders only one is probably required to do the job. In process #13, final transport, we discover that from 195.29 hrs this process was active the inventory buffer overflow existed for 73.04 hrs. An increase of the buffer size would yield an increase in utilization for the preceding process # 3, delimbing & bucking. When increasing the utilization of process #3, the utilization of its predecessor, process #2 skidding, will also increase since the inventory overflow condition set for process #3 will not be reached so often. In this harvesting configuration the inventory limit imposed on process #13 creates a serious bottleneck for all the preceding processes. Thus, by examining the results for a given harvesting configuration, we could focus on variables like inventory downtimes, machine utilizations, and machine breakdown hours. These suggestions could then be implemented and tested by simulating the modified harvesting model. By using the simulation model in this way the optimization of a given harvesting system can be attempted, where the optimization goals could be costs, machine utilizations, or duration of the logging operation.

Also the impact of environmental issues could be tested, for example, restriction on landing sizes, travel-speed limits for machines due to soil conditions, in-woods limitations on inventory buffers, and the required use of a specific machine configuration to prevent damage to the environment. The impact of those restrictions on the cost structure and machine utilization could be analyzed and different machine configurations can be tested to find the best solution.

The fourth harvesting model demonstrates a system with multiple products. As described above, the system performance and behavior of the chosen harvesting configuration can be examined. However, this system also analyzes the cost structure for a given product. Since the model shows the cost per unit for each of the activated processes, the analyst can determine the actual cost per unit for each of the end products, chips and sawlogs. By simply adding the appropriate values of the processes used by each product, the individual cost per unit chips and per unit sawlog can be determined (see Table 2). Chips have a cost of approximately 52 cents per unit, sawlogs approximately 45 cents. By setting the costs in relation to the expected selling prices an estimate of the profits can be made.

Table 2: Cost per unit

Cost per unit chips:

Process		Cost per unit
#1	Felling	0.0551
#2	Skidding	0.0604
#3	Swinging	0.0291
#11	Sorting	0.0000
#4	Delimbing & Debarking	0.0681
#12	Chipping & Transport	0.2390
	Loading actions	0.0680
Sum cost per unit		$ 0.5197

B. RUNTIME AND HARDWARE CONSIDERATIONS

The model was created to be used with a hard disk as mass storage medium. Since the programs are too large to fit on one floppy disk, excessive disk swapping is necessary, making the handling of the simulation system on a floppy disk based system inconvenient.

We also recommend the that the system used for simulation should be equipped with a Math co-processor for floating point arithmetic such as the INTEL 8087/80287. Besides an execution speed gain of two to three times the coprocessor will improve the mathematical accuracy of the simulation. The SLAM simulation processor uses a FORTRAN data format of REAL*4 for most variables. This format has advantages in terms of memory requirements but has a poor performance for floating point arithmetic in terms of accuracy if no Math co-processor is used.

The examples where all computed on an IBM-AT compatible computer system equipped with a 10 Mhz, no wait state motherboard and an 8 Mhz 80287 Math co-processor unit. This system runs approximately 4 to 7 times faster than a plain IBM-PC. Therefore, at least an AT size machine is recommended for the simulation.

When simulating a larger harvesting system that requires multiple simulation runs the task of averaging all those numbers for all the given simulation results can be quite tedious. By redirecting the simulation output to a Disk file (start the simulation with LOGSIM.EXE>FILENAME.DOC) these results can later be imported to a Spreadsheet program like LOTUS 1-2-3 that could be used to ease this task.

C. MODELING CONSIDERATIONS

The potential user of the LOGSIM system should be aware that the produced simulation results are only as accurate as the entered simulation parameters. The task of collecting the required data for the simulation might be difficult. The performance parameters for machines such as processing times and machine breakdowns might require extensive time studies to establish. The mathematical relationships are needed between machine actions and environmental parameters. These include the influence of stand parameters and terrain conditions on the productivity of a given machine. Extensive research has been done to establish some of these mathematical relationships (for examples see McMoreland, 1977;

O'Hearn, 1977; Powell, December 1981; Powell, July 1981; and Stuart et al, 1981), but there is still much to be done.

The variables that influence the machine productivity for a given machine have to be specified. The modeler or analyst should then decide which ones are essential for the modeling of a given harvesting configuration and which level of accuracy is desired. Then the actual values for these parameters should be developed and incorporated into the harvesting model.

It is the responsibility of the modeler to judge if the built-in functions of the model are sufficient or if FORTRAN written user functions have to be employed.

D. STATISTICAL ANALYSIS OF SIMULATION RESULTS

Another aspect of simulation, the statistical analysis of simulation results, should also be considered by the modeler. Simulation represents a tool to generate, collect, and analyze statistical data for a given system. Therefore simulation is a statistical experiment that should be planned carefully. The analyst should be aware what the variables of interest are and plan accordingly.

There are two types of simulations with regard to analysis of the output data. A terminating simulation is one for which the desired measures of system performance are defined relative to the interval of simulated time. Examples for this type of simulation are the time it takes to harvest a given stand with a given machine configuration, and the complete costs to harvest a given stand. A steady-state simulation is one that defines the measures of performance as limits as the length of the simulation goes to infinity. Examples are the cost per unit or the average inventory size of a given process.

The terminating simulation type requires multiple simulation runs to achieve a statistically acceptable number of observations or sample size. These samples should then be averaged to obtain representative simulation results. By calculating the confidence intervals for the desired simulation results the user can determine if additional runs are necessary to achieve the desired level of statistical confidence.

When the variable of interest are of the steady-state type, the user has to set the amount to be harvested large enough to reach the steady state condition. Again, by calculating the appropriate confidence intervals it can be determined if the length of the performed simulation run was large enough. For an excellent discussion of the statistical techniques used to perform these analysis see Law and Kelton, 1982. When the performance of different harvesting configurations under the same stand conditions are analyzed, statistical tests like the T-test should be employed to test the hypothesizes in question.

Therefore, a preliminary analysis before each simulation project should be done. This analysis should determine if the expected simulation results and the benefits of using simulation as an analytical tool will outweigh the considerable efforts of preparing the required data for the simulation. Also by taking into account the variable type, the appropriate sampling method (multiple runs or one single run) should be chosen to insure the statistical validity.

VII. CONCLUSIONS AND SUGGESTIONS FOR FUTURE RESEARCH

A. CONCLUSIONS

The simulation model for log harvesting represents a general solution for modeling any kind of production process that has to deal with restrictions on inventory sizes between processes. By simply labeling the processes accordingly, nearly any kind of operation that has the same structure can be modeled. We feel, that with the incorporation of FORTRAN user functions, the modeler should be able to handle the modeling of operations with a minimum of new programming.

Currently computers build around the advanced 80386 processing unit have been introduced with speeds up to 20 Mhz. With the increased availability of such computers and multitasking operating systems, simulation and numerical analysis will be even more practical.

B. SUGGESTIONS FOR FUTURE RESEARCH

Future research could be done mainly in two areas. The first is to concentrate on the model itself and enhance it in various ways. The model could be enlarged to simulate multiple harvesting sites accessing the same processes or sharing equipment. Instead of using the distribution approach for modeling, userfunctions could be developed to build a spacial model of the stand and the harvesting area to better investigate environmental influences. The simulation output could be enhanced to make it more readable and to include additional information. The input user-interface could be improved to make the entering of data easier. The build-in time handling functions for processing times also can be improved by including processing times based on statistical distributions and processing times based on the tree volume.

The second type of research would be to undertake time studies of the various machines used in the timber harvesting process. This research is needed to find the variables of interest that influence the performance of the system.

BIBLIOGRAPHY

Anonymous; 1971
National timber supply is in the public eye.
Forest Industry Journal 98(4):11

Bare, D.B.; B.A. Jayen; B.F. Anholt; 1976
A simulation based approach for evaluating logging residual handling systems.
USDA Forest Service Report PNW-45;
Portland, Oregon

Bradley, D.P.; R.E. Biltonen; S.A. Winsauer; 1976
A simulation model for full-tree chipping and trucking.
USDA Forest Service, Research Paper NC-129;
St. Paul, Minnesota

Bussel, W.H.; J.N. Hool; A.M. Leppet; G.R. Harmon; 1969
Pulpwood harvesting systems analysis.
Report to the Southern Executive Association, Auburn University;
Auburn, Alabama

Carson, Barry; March 1984;
Evaluation of six short rotation harvesting systems.
PH.D.-Thesis, University of Washington;
Seattle, Washington

Conway, Steve; 1976
Logging Practices: Principles of Timber Harvesting Systems.
Miller Freeman Publications, Inc.;
San Francisco, California

Etter, D.M.; 1984
PROBLEM SOLVING with Structured FORTRAN 77.
The Benhamin/Cummings Publishing Company, Inc.
Menlo Park, California

Garland, John J.; Spring 1986
Seminar for mechanized harvesting operations.
Forest Engineering Institute,
Oregon State University;
Corvallis, Oregon

Gerstkemper, John C.; 1982
A simulation of the operation of a log landing for a Heli-Stat Airship in old growth timber.
Research paper, Oregon State University,
Dep. of Forestry; June 1982

Goulett, Daniel v.; Ronald H. Iff; Donald L. Sirois; 1979
Tree-to-mill forest harvesting simulation models: Where are we?
Forest Products Journal: 50-55; October 1979

Johnson, L.R.; 1970
Simulation of the loading and hauling subsystems of a logging system.
MS-Thesis, Montana State University;
Bozeman, Montana

Johnson, L.R.; 1976
SAPLOS: Documentation and use.
Report to USFS Northwestern Forest Experiment Station, Morgantown W.- Virginia;
University of Idaho, Moscow, Idaho

Kellogg, Loren D.; July 1986
Center for wood utilization research, mechanized harvesting of small timber: Study plan, years 2-5.
Department of Forest Engineering,
Oregon State University;
Corvallis, Oregon

Killham, J.R.; 1975
The development of a forest harvesting simulation model.
MS-Thesis, Auburn University;
Auburn, Alabama

Law, Averill M.; W. David Kelton; 1982
Simulation Modeling and Analysis.
McGraw-Hill Book Company, Inc.
New York, N.Y.

Ledoux, Chris B.; 1975
Simulation of a helicopter yarding system in old growth timber stands.
MS-Thesis, Oregon State University;
Corvallis, Oregon

Lilegdon, William R.; Jean J. O'Reilly, 1986
SLAM II PC Version User's Manual.
Pritsker & Associates, Inc.;
West Lafayette, Indiana

Lohman, H.; Lehnhausen, H.; 1983
Systemanalyse eines Holzhofes durch die Simulation des Materialflusses.
Forstarchiv 54(6):221-228;
Goettingen, W.-Germany

Martin, A.J.; 1976
A user's guide for THATS.
USDA Northeastern Forest Experiment Station;
Princeton, W.-Virginia

McMoreland, B.A.; 1977
Evaluation of Volvo VM 971 Clam Bunk Skidder.
FERIC Tech. Report

Microsoft Corporation, 1985
Microsoft FORTRAN Compiler, User's Guide.
Microsoft Corporation,
Redmond, Washington

Microsoft Corporation, 1985
Microsoft FORTRAN Compiler, Reference Manual.
Microsoft Corporation,
Redmond, Washington

Newnham, R.M.; S. Sjunnesson; 1969
A FORTRAN program to simulate harvesting machines for mechanized thinning.
Forest management research and service Institute, Report FMR-X-23
Ottawa, Ontario Canada

O'Hearn, S.E.; B.W. Stuart; T.A. Walbridge; 1976
Using computer simulation for comparing performance criteria between harvesting systems.
1976 Winter Meeting, American Society for Agricultural Engineers,
Paper No. 76-1567

O'Hearn, S.E; 1977
Economic and productivity comparisons between full tree chipping and conventional harvesting systems on a variety of stand types.
M.S. thesis, Virginia Polytechnical Institute and State University;
Blacksburg, Virginia

O'Reilly, Jean J.; 1984
SLAM II Quick Reference Manual.
Pritsker & Associates, Inc.;
West Lafayette, Indiana

Powell, L.H.; December 1981
Interior limbing, bucking, and processing study - evaluation of Hahn Tree-length Delimber.
For.Eng.Res.Inst. of Canada;
Tech. Note No. TN-51

Powell, L.H.; July 1981
Interior limbing, bucking, and processing study - evaluation of Barko 450 Loader.
For.Eng.Res.Inst. of Canada;
Tech. Note No. TN-46

Pritsker, A.Alan B., Claude Dennis Pedgen; 1984
Introduction to simulation and SLAM II.
2 nd edition, Halsted Press, a Division of
John Wiley & Sons, INC.;
New York, NY

Sessions, John; 1985;
Class notes FE 365.
Department of Forest Engineering;
Oregon State University;
Corvallis, Oregon

Simmons, Fred C.; 1979
Handbook for Eastern Timber Harvesting.
U.S. Dep. of Agriculture, Forest Service,
Northeastern Area, State & Private Forestry,
Broomall, Pennsylvania

Stark, J.I.; 1975;
A simulation model for the common pulpwood harvesting systems of the southern pine region.
MS-Thesis, Dep. of Ind. Eng., Georgia Tech Inst. of Technology;
Atlanta, Georgia

Stuart, W.B.; J.V. Perumpral; T.A. Walbridge;
S. Shartle; 1981
Pine plantation data for future equipment design.
Am.Soc. of Agric.Eng. Transactions Vol.24 No.3

Webster, D.B.; 1984
Guidelines for the development of simulation models.
Paper presented at the conference COFE/IUFRO;
SAF Publication No. 84-13:81-86
Orono, Maine

Winsauer, S.A.; Bradley, P. Dennis; 1982
A program and documentation for simulation of rubber-tired feller-buncher.
Research Paper NC-212, U.S. Dep. of Agriculture,
Forest Service, N. Cen. Forest Exp. Station,
St. Paul, Minnesota

Winsauer, Sharon A.; 1984
Simulation of mechanized felling in dense softwood plantations.
Paper presented at the conference COFE/IUFRO;
SAF Publication No. 84-13:175-180
Orono, Maine

APPENDICES

APPENDIX A

TABLE OF CONTENTS:

APPENDIX A

1. Listing, SLAM Network

```
;**************************************************************
;*                                                            *
;*                 OREGON STATE UNIVERSITY                    *
;*                        JUNE 1986                           *
;*                                                            *
;*                  >>>  L O G S I M <<<                      *
;*                                                            *
;*     SIMULATION OF MECHANIZED LOG HARVESTING SYSTEMS        *
;*                                                            *
;*                                                            *
;*     DESIGNED BY : CHRISTOPH WIESE                          *
;*                   MASTERS CANDIDATE, DEP. OF INDUSTRIAL    *
;*                   ENGINEERING, OREGON STATE UNIVERSITY     *
;*                                                            *
;*     DESIGNED FOR: DEPARTMENT OF FOREST ENGINEERING         *
;*                   OREGON STATE UNIVERSITY                  *
;*                                                            *
;*                                                            *
;*     SUPERVISION : DR. ELDON OLSEN                          *
;*                   ASSOCIATE PROFESSOR, DEP. OF INDUSTRIAL  *
;*                   ENGINEERING, OREGON STATE UNIVERSITY     *
;*                                                            *
;*                                                            *
;**************************************************************
;*                                                            *
;*               SLAM II  NETWORK: HARVEST.TRA                *
;*                                                            *
;*                     31-MAY-87  18:55                       *
;*                                                            *
;**************************************************************
;
;
;
;        SYSTEM STATEMENTS:
;        =================
;
;
GEN,CHRISTOPH WIESE,MECH.LOG HARVEST SIM,5/31/1987,10,Y,N,Y,Y,N;
INITIALIZE,0,99999.0;
LIMITS,63,7,200;
;MONTR,TRACE,15.75,50,II,TNOW,1,2,3,4,5,6,7,-1,-2,-3,-4,-5,-7,-38,-31,-40;
;
;
;
;
;        ARRAY:
;        ------
;
ARRAY(1,42)/0.00;
;     AVERAGE PROCESSING TIME PER UNIT
ARRAY(2,42)/0.00;
;     FIXED CONSTANT PROCESSING TIME PER LOAD
```

```
ARRAY(3,42)/0.00;
;    FIXED CONSTANT TIME FOR ONE WAY HAULING
ARRAY(4,13)/0.00;
;    WHICH LOADER TO USE IF TRANSPORT FUNCTION DESIRED 0=NONE
ARRAY(5,13)/0.00,0,0,0,0,0,0,0,0,0,0,0,0;
;    ARE WE DEALING WITH 0=TREES  1=LOGS 2=SAWLOGS 3=PULPLOGS
ARRAY(6,42)/0.00;
;    # OF INITIALLY AVAILABLE RESOURCES
ARRAY(7,18)/0.00;
;     SORTING PARAMETERS
ARRAY(8,42)/0.00;
;    THRESHHOLD LEVEL FOR INDIVIDUAL BATCHES FOR THE MACHINES
ARRAY(9,42)/0.00;
;    WHAT TO MODEL AT EACH PROCESS  0=AVERAGE  1=USERF
ARRAY(10,42)/0.00;
;   ACCUMULATED MACHINE HOURS
ARRAY(11,13)/0.00;
;   ACCUMULATED INVENTORY DOWNTIME HOURS, INFEED
ARRAY(12,13)/0.00;
;   ACCUMULATED INVENTORY DOWNTIME HOURS, OUTFEED
ARRAY(13,2)/0.00;
;   ARRAY LINE FOR CHIPPING PARAMETERS
ARRAY(14,2)/0.00;
;   ARRAY LINE FOR FTRAPO PARAMETERS
ARRAY(15,14)/0.00;
;   FLAG PROCESS IS UP & RUNNING 0=NOT STARTED 1=UP 2=ENDED 3=ENDED+STATS
ARRAY(16,13)/0.00;
;   START TIME FOR PROCESS & CALC. SCHEDULED HOURS
ARRAY(17,13)/0.00;
;   START TIME INVENTORY DOWNTIME
ARRAY(18,42)/0.00;
;   CUMULATED MACHINE BREAKDOWN TIMES OF MACHINES
ARRAY(19,13)/0.00;
;   FLAG PROCESS ALTERED BECAUSE OF INFEED INVENTORY 0=NO 1=YES
ARRAY(20,13)/0.00;
;   FLAG PROCESS ALTERED BECAUSE OF OUTFEED INVENTORY 0=NO 1=YES
ARRAY(21,42)/0.00;
;   FIXED COSTS (MACHINE = OPERATOR) PER SCHEDULED HOUR
ARRAY(22,42)/0.00;
;   VARIABLE COSTS PER MACHINE HOUR
ARRAY(23,13)/0.00;
;   INDEXES WHERE GOES THE INVENTORY TO
ARRAY(24,13)/0.00;
;   INDEXES WHERE COMES THE INVENTORY FROM
ARRAY(25,6)/0.00;
;    ARRAY FOR STATISTICAL VALUES AT END OF PROCESS
ARRAY(26,13)/0.00;
;   INVENTORY IN TRANSIT
;
;
```

```
; EQUIVALENCE STATMENTS TO INDEXING THE BEGINNING OF XX-VARIABLES BLOCKS
; --------------------------------------------------------------------
;
EQUIVALENCE/15,LEVEL1;  HOW MANY CU FT HAVE BEEN PROCESSED
EQUIVALENCE/27,LEVEL2;  INFEED INVENTORY
EQUIVALENCE/39,LEVEL3;  STOPPING LEVEL INFEED INVENTORY TOO LOW
EQUIVALENCE/51,LEVEL4;  STOPPING LEVEL OUTFEED INVENTORY TOO HIGH
EQUIVALENCE/63,LEVEL5;  STARTUP INVENTORY INFEED
EQUIVALENCE/75,LEVEL6;  STARTUP INVENTORY OUTFEED
EQUIVALENCE/87,LEVEL7;  STARTUP INVENTORY LEVEL FOR PROCESS
;
;
;
; TIMST FOR COLLECTING STATISTICS ABOUT INVENTORY
; ---------------------------------------------
;
;
STAT,2,INV.PROCESS 2;
STAT,3,INV.PROCESS 3;
STAT,4,INV.PROCESS 4;
STAT,5,INV.PROCESS 5;
STAT,6,INV.PROCESS 6;
STAT,7,INV.PROCESS 7;
STAT,8,INV.PROCESS 8;
STAT,9,INV.PROCESS 9;
STAT,10,INV.PROCESS10;
STAT,11,INV.DISTRIUBTION;
STAT,12,INV.CHIPPING;
STAT,13,INV.FTRAPO;
;
;
;
;
NETWORK;
;
;
;       RESOURCES USED:
;       ==============
;
;
;
        RESOURCE/FELLER1(0),20,1;
        RESOURCE/FELLER2(0),20,2;
        RESOURCE/FELLER3(0),20,3;
        RESOURCE/FELLER4(0),20,4;
        RESOURCE/PROC.1.1(0),20,5;
        RESOURCE/PROC.1.2(0),20,6;
        RESOURCE/PROC.1.3(0),20,7;
        RESOURCE/PROC.2.1(0),20,8;
        RESOURCE/PROC.2.2(0),20,9;
        RESOURCE/PROC.2.3(0),20,10;
        RESOURCE/PROC.3.1(0),20,11;
        RESOURCE/PROC.3.2(0),20,12;
```

```
        RESOURCE/PROC.3.3(Ø),2Ø,13;
        RESOURCE/PROC.4.1(Ø),2Ø,14;
        RESOURCE/PROC.4.2(Ø),2Ø,15;
        RESOURCE/PROC.4.3(Ø),2Ø,16;
        RESOURCE/PROC.5.1(Ø),2Ø,17;
        RESOURCE/PROC.5.2(Ø),2Ø,18;
        RESOURCE/PROC.5.3(Ø),2Ø,19;
        RESOURCE/PROC.6.1(Ø),2Ø,2Ø;
        RESOURCE/PROC.6.2(Ø),2Ø,21;
        RESOURCE/PROC.6.3(Ø),2Ø,22;
        RESOURCE/PROC.7.1(Ø),2Ø,23;
        RESOURCE/PROC.7.2(Ø),2Ø,24;
        RESOURCE/PROC.7.3(Ø),2Ø,25;
        RESOURCE/PROC.8.1(Ø),2Ø,26;
        RESOURCE/PROC.8.2(Ø),2Ø,27;
        RESOURCE/PROC.8.3(Ø),2Ø,28;
        RESOURCE/PROC.9.1(Ø),2Ø,29;
        RESOURCE/PROC.9.2(Ø),2Ø,3Ø;
        RESOURCE/PROC.9.3(Ø),2Ø,31;
        RESOURCE/LOADER1(Ø),2Ø,32;
        RESOURCE/LOADER2(Ø),2Ø,33;
        RESOURCE/LOADER3(Ø),2Ø,34;
        RESOURCE/LOADER4(Ø),2Ø,35;
        RESOURCE/LOADER5(Ø),2Ø,36;
        RESOURCE/CHIPPER1(Ø),2Ø,37;
        RESOURCE/CHIPPER2(Ø),2Ø,38;
        RESOURCE/CHIPTRA1(Ø),2Ø,39;
        RESOURCE/CHIPTRA2(Ø),2Ø,4Ø;
        RESOURCE/FTRAPO1(Ø),2Ø,41;
        RESOURCE/FTRAPO2(Ø),2Ø,42;
;
;
;
;
;
;
;
;
;
;       GATES USED:
;       ===========
;
; SITE 1:
;
        GATE/1,GATE1,CLOSE,51;
        GATE/2,GATE2,CLOSE,52;
        GATE/3,GATE3,CLOSE,53;
        GATE/4,GATE4,CLOSE,54;
        GATE/5,GATE5,CLOSE,55;
        GATE/6,GATE6,CLOSE,56;
        GATE/7,GATE7,CLOSE,57;
        GATE/8,GATE8,CLOSE,58;
        GATE/9,GATE9,CLOSE,59;
```

```
        GATE/10,GATE10,CLOSE,60;
        GATE/11,GATE11,CLOSE,61;
        GATE/12,GATE12,CLOSE,62;
        GATE/13,GATE13,CLOSE,63;
;
;
;
;
;
;
;
;*****************  FIRST NETWORK   *********************
;*                                                      *
;*   NETWORK TO START-UP THE SIMULATION                 *
;*                                                      *
;********************************************************
;
;
;
;  CREATING INITIALIZATION ENTITY AND READING VALUES FROM FILE
;  ----------------------------------------------------------
;
;
         CREATE,,,,1,1;                  CREATE ONE ENTITY FOR START
         GOON,1;
          ACT,,XX(1).EQ.0,SEN6              NO SIMULATION AT ALL
          ACT;
;
;
         ASSIGN,II=ARRAY(4,11),1;         INDEXING LOADER FOR DISTRIB
          ACT,,II.EQ.0,SY0;
          ACT;
         ASSIGN,ARRAY(4,9)=ARRAY(8,II);   STORE BATCHSIZE OF THIS LOADER
;
;
;
;  PLACING LOAD ENTITIES INTO THE NETWORK
;  -------------------------------------
;
;
SY0      ASSIGN,XX(3)=0;
SY1      ASSIGN,XX(3)=XX(3)+1,1;          PROCESS COUNTER
         GOON,1;
          ACT,,XX(3).EQ.1,SY2;
          ACT,,XX(3).GT.1.AND.XX(3).LE.10,SY3;
          ACT,,XX(3).EQ.11,SY1;
          ACT,,XX(3).EQ.12,SY4;
          ACT,,XX(3).EQ.13,SY5;
          ACT,,XX(3).GT.13,SY16;
;
;
```

```
SY11     ASSIGN,XX(2)=0,1;
          ACT,,XX(1).EQ.0,SY1;
          ACT;
;
SY12     ASSIGN,XX(2)=XX(2)+1;              COUNTER
         GOON,2;
          ACT,,XX(2).LT.XX(1),SY12;
          ACT,,XX(3).EQ.1,FELL;
          ACT,,XX(3).EQ.2,PRO1;
          ACT,,XX(3).EQ.3,PRO2;
          ACT,,XX(3).EQ.4,PRO3;
          ACT,,XX(3).EQ.5,PRO4;
          ACT,,XX(3).EQ.6,PRO5;
          ACT,,XX(3).EQ.7,PRO6;
          ACT,,XX(3).EQ.8,PRO7;
          ACT,,XX(3).EQ.9,PRO8;
          ACT,,XX(3).EQ.10,PRO9;
          ACT,,XX(3).EQ.12,CHIP;
          ACT,,XX(3).EQ.13,FTRA;
          ACT,,,SY1;
;
;
SY2      ASSIGN,XX(1)=ARRAY(6,1)+ARRAY(6,2)+ARRAY(6,3)+ARRAY(6,4);
          ACT,,,SY11;
;
SY3      ASSIGN,II=XX(3)*3-1;
         ASSIGN,XX(1)=ARRAY(6,II),II=II+1;
         ASSIGN,XX(1)=XX(1)+ARRAY(6,II),II=II+1;
         ASSIGN,XX(1)=XX(1)+ARRAY(6,II);
          ACT,,,SY11;
;
SY4      ASSIGN,XX(1)=1;
          ACT,,,SY11;
;
SY5      ASSIGN,XX(1)=ARRAY(6,41);
          ACT,,,SY11;
;
;
;
;
;
;  BRING UP FELLER RESOURCES TO INITIATE THE PROCESS
;  ------------------------------------------------
;
;
SY16      ALTER,FELLER1,ARRAY(6,1);
          ALTER,FELLER2,ARRAY(6,2);
          ALTER,FELLER3,ARRAY(6,3);
          ALTER,FELLER4,ARRAY(6,4);
          ALTER,LOADER1,ARRAY(6,32);
          ALTER,LOADER2,ARRAY(6,33);
          ALTER,LOADER3,ARRAY(6,34);
          ALTER,LOADER4,ARRAY(6,35);
```

```
        ALTER,LOADER5,ARRAY(6,36);
        ASSIGN,ARRAY(15,1)=1;
        OPEN,GATE1;
        ASSIGN,ATRIB(2)=1,ATRIB(3)=4,ATRIB(5)=1,2;      INIT MACH.BREAKDOWN
         ACT,,,SY13;
         ACT;
        ASSIGN,ATRIB(2)=32,ATRIB(3)=36,ATRIB(5)=14,2;   INIT MACH.BREAKDOWN
         ACT,,,SY13;
         ACT;
;
DEST    TERMINATE;          END INITIALIZATION SIMULATION
;
;
;
;
;
;****************** SECOND NETWORK   *********************
;*                                                       *
;*   NETWORK TO SIMULATE MACHINE BREAKDOWN               *
;*                                                       *
;*********************************************************
;
;
;  PLACING ENTITIES INTO MACHINE BREAKDOWN NETWORK
;  ----------------------------------------------
;
;
SY13    ASSIGN,XX(3)=ATRIB(2)-1;
SY14    ASSIGN,XX(3)=XX(3)+1,1;
         ACT,,XX(3).GT.ATRIB(3),DEST;
         ACT;
;
        ASSIGN,XX(2)=Ø;
        ASSIGN,XX(1)=ARRAY(6,XX(3)),1;
         ACT,,XX(1).EQ.Ø,SY14;
         ACT;
;
SY15    ASSIGN,XX(2)=XX(2)+1,ATRIB(1)=XX(3);
        GOON,2;
         ACT,,XX(2).LT.XX(1),SY15;
         ACT,,,SY2Ø;
         ACT,,,SY14;
;
;
;
;
;
;
```

```
;  MAIN ROUTINE TO MODEL MACHINE BREAKDOWN
;  ---------------------------------------
;  ATRIB(1)=MACHINE #, ATRIB(2)=TIME BETWEEN FAILIURES, ATRIB(3)=REPAIR TIME,
;
;
SY20     GOON,1;                          BEGIN MAIN ROUTINE
         ASSIGN,XX(5)=110;                INDEXING USERFUNCTION
         ASSIGN,ATRIB(2)=USERF(110);      ASSIGNING TIME BETWEEN FAILIURES
         ASSIGN,XX(5)=111;                INDEXING USERFUNCTION
         ASSIGN,ATRIB(3)=USERF(111),1;    ASSIGNING REPAIR TIME
          ACT,,ATRIB(2).EQ.0,DEST;        NO BREAKDOWN OF THIS MACHINE
          ACT,,ARRAY(15,ATRIB(5)).GT.1,DEST;  END OF PROCESS
          ACT;
;
         GOON,1;
         ACT,ATRIB(2);                    TIME BETWEEN FAILURE
         GOON,1;
          ACT,,ARRAY(15,ATRIB(5)).GT.1,DEST;  END OF PROCESS
          ACT;
SY24     AWAIT(ATRIB(1)=1,50),ATRIB(1)/1;   MACHINE FAILURE,SEIZE MACHINE
         ACT,ATRIB(3);                    REPAIR TIME
         ASSIGN,ARRAY(18,ATRIB(1))=ARRAY(18,ATRIB(1))+ATRIB(3); ADD DOWNTIME
         FREE,ATRIB(1)/1,1;               FREE RESOURCE, REPAIR OVER
          ACT,,ATRIB(5).EQ.14,SY20;
          ACT;
         OPEN,ATRIB(5),1;
          ACT,,,SY20;                     CONTINUE BREAKDOWN CYCLE
;
;
;
;
;;****************** THIRD NETWORK   *********************
;*                                                        *
;*   MAIN NETWORK TO SIMULATE THE FELLING OPERATION       *
;*                                                        *
;*********************************************************
;
;
;
;
;  ROUTINE FOR ALTERING PROCESS RESOURCES BECAUSE OF INVENTORY/END
;  ---------------------------------------------------------------
;
;
AL       GOON,1;                          GO TO INDEXING WHICH RESOURCES TO ALTER
          ACT,,ATRIB(5).EQ.1,IN1;
          ACT,,ATRIB(5).GT.1.AND ATRIB(5).LE.10,IN2;
          ACT,,ATRIB(5).EQ.11,DEST;
          ACT,,ATRIB(5).EQ.12,IN3;
          ACT,,ATRIB(5).EQ.13,IN4;
          ACT,,,DEST;
;
;
```

```
IN1     ASSIGN,ATRIB(2)=1,ATRIB(3)=4;            FELLING
         ACT,,,INØ;
;
IN2     ASSIGN,ATRIB(2)=ATRIB(5)*3-1,ATRIB(3)=ATRIB(2)+2; PROCESSES
         ACT,,,INØ;
;
IN3     ASSIGN,ATRIB(2)=37,ATRIB(3)=37;          CHIPPING
         ACT,,,INØ;
;
IN4     ASSIGN,ATRIB(2)=41,ATRIB(3)=41;          FINAL TRANSPORT
         ACT,,,INØ;
;
INØ     GOON,1;                                  GO TO DESIRED ALTERING:
         ACT,,ATRIB(1).EQ.1,AL1;                 DECREASE REOURCES
         ACT,,ATRIB(1).EQ.2,AL2;                 MAKE RESOURCE AGAIN AVAILABLE
;
;
;
;
AL1     ASSIGN,XX(1)=-1;                         SET XX(1) NEGATIVE
        ASSIGN,XX(1)=XX(1)*ARRAY(6,ATRIB(2));    INDEXING HOW MANY RESC.AVAIL.
        ALTER,ATRIB(2),XX(1);                    DECREASE AVAILABLE RESOURCES
        ASSIGN,ATRIB(2)=ATRIB(2)+1,1;
         ACT,,ATRIB(2).LE.ATRIB(3),AL1;
         ACT;
;
        CLOSE,ATRIB(5);                          CLOSE GATE OF PROCESS
        TERMINATE;
;
;
;
AL2     ASSIGN,II=ATRIB(2),XX(1)=ARRAY(6,II);    MAKE RESOURCES BACK AVAILABLE
        ALTER,ATRIB(2),XX(1);
        ASSIGN,ATRIB(2)=ATRIB(2)+1,1;
         ACT,,ATRIB(2).LE.ATRIB(3),AL2;
         ACT;
;
        OPEN,ATRIB(5);                           OPEN GATE OF PROCESS
        TERMINATE;
;
;
;
;  ROUTINE FOR PROCESSING
;  ----------------------
;
;
PROC    GOON,1;
        ASSIGN,XX(5)=1Ø1
        ACT,USERF(2);                  SET BATCHSIZE & TREES
GO1Ø    GOON,1;
         ACT,,ATRIB(5).EQ.1,GO11;
         ACT;
;
```

```
        GOON,1;
         ACT,,ATRIB(2).EQ.0,WAIT;
         ACT;
        ASSIGN,II=LEVEL2+ATRIB(5),XX(II)=XX(II)-ATRIB(2),1; CALC.INFEED INVEN.
        ASSIGN,XX(5)=103;                       INDEXING USERF
        ACT,USERF(103);                         COLCT STATS
;
GO11    ASSIGN,ARRAY(26,ATRIB(5))=ARRAY(26,ATRIB(5))+ATRIB(2);INV.IN TRANSIT
        ASSIGN,II=LEVEL1+ATRIB(5),XX(II)=XX(II)+ATRIB(2),1; SUM.PROCESSED
        GOON,2;
         ACT,,,END;                             TEST CURRENT PROCESS END
         ACT;
;
        GOON,1;                                 TEST IF LAST ENTITY OF PROCES
         ACT,,ARRAY(15,ATRIB(5)).LT.2,GO12;
         ACT;
        ASSIGN,ATRIB(7)=1;                      MARK LAST ENTITY
;
GO12    GOON,1;                                 REROUTE IF FELLING ENTITIY
         ACT,,ATRIB(5).EQ.1,AS20;
         ACT;
;
        GOON,2;
         ACT,,,INLO;            TEST INFEED INVENTORY CURRENT PROC.TOO LOW
         ACT;
;
        GOON,2;
         ACT,,,OUTN;            TEST OUTFEED INVENTORY PREVIOUS PROC.NORMAL?
         ACT;
;
AS20    ASSIGN,ATRIB(4)=TNOW,1;
         ACT,,ARRAY(4,ATRIB(5)).GT.0,LOAD;    DIVERT IF LOAD FCTN
         ACT;
;
GO13    GOON,1;
         ACT,,ATRIB(5).EQ.13.AND.ARRAY(6,42).GT.0,AW50; SEIZE TRACTOR FTRAPO
         ACT;
;
GO14    GOON,1;
         ACT,,ARRAY(9,ATRIB(5)).EQ.0,GO15;      MODELLING CONSTANT TIMES
         ACT,,ARRAY(9,ATRIB(5)).EQ.1,GO16;      MODELLING WITH USERFUNCTIONS
;
;
GO15    ASSIGN,ATRIB(3)=ATRIB(3)*ARRAY(1,ATRIB(1)); CALC VARIABLE PROCESS TIME
        ACT,ATRIB(3);
        GOON,1;
         ACT,,,GO17;
;
GO16    ASSIGN,XX(5)=ATRIB(1);
        ACT,USERF(3);
;
;
```

```
G017    GOON,1;
        ACT,ARRAY(2,ATRIB(1));                   1ST CONSTANT TIME FACTOR
;
        GOON,1;
        ACT,ARRAY(3,ATRIB(1));                   2ND CONSTATNT TIME FACTOR
;
        GOON,1;
         ACT,,ATRIB(5).GE.12,G018;               REROUTE IF FTRAPO/CHIPPING
         ACT;
;
        ASSIGN,XX(1)=ARRAY(23,ATRIB(5));        INDEXING NEXT PROCESS
        ASSIGN,II=LEVEL2+XX(1),XX(II)=XX(II)+ATRIB(2); INV.CALC.INFEED NEXT
        ASSIGN,XX(5)=1Ø4;                        INDEXING USERF
        ACT,USERF(1Ø4);                          COLCT STATS
        ASSIGN,ARRAY(26,ATRIB(5))=ARRAY(26,ATRIB(5))-ATRIB(2);INV.IN TRANSIT
;
        GOON,2;
         ACT,,,OUTH;                            TEST OUTFEED INVENTORY TOO HIGH
         ACT;
;
        GOON,2;
         ACT,,,INN;                              TEST INFEED INVENTORY NORMAL?
         ACT;
;
G018    GOON,1;
        ACT,ARRAY(3,ATRIB(1));                   2ND CONSTANT TIME FACTOR
;
        ASSIGN,ATRIB(4)=TNOW-ATRIB(4);           CALCULATE MACHINE HOURS
        ASSIGN,ARRAY(1Ø,ATRIB(1))=ARRAY(1Ø,ATRIB(1))+ATRIB(4); ADD MACHINE HRS
        FREE,ATRIB(1)/1,1;                     FREE RESOURCE
         ACT,,ATRIB(5).EQ.13,FTST;             REROUTE IF FTRAPO
         ACT;
G021    OPEN,ATRIB(5),1;                       OPEN GATE CURRENT PROCESS
         ACT,,ATRIB(5).EQ.12,G063;             REROUTE IF CHIPPING
         ACT,,ATRIB(7).EQ.1,STAT;              GOTO STATS IF LAST ENTITY
         ACT;
;
G019    GOON,1;
         ACT,,ATRIB(5).EQ.12.AND.ATRIB(7).EQ.1,G065;    LAST ENTITY CHIPPING
         ACT,,ATRIB(5).EQ.13,FTRA;                      REROUTE IF FTRAPO
         ACT;
;
        ASSIGN,ATRIB(1)=ARRAY(23,ATRIB(5));    INDEXING NEXT PROCESS
        OPEN,ATRIB(1),1;                       OPEN GATE NEXT PROCESS
G02Ø    GOON,1;
         ACT,,ATRIB(5).EQ.1,FELL;
         ACT,,ATRIB(5).EQ.2,PRO1;
         ACT,,ATRIB(5).EQ.3,PRO2;
         ACT,,ATRIB(5).EQ.4,PRO3;
         ACT,,ATRIB(5).EQ.5,PRO4;
         ACT,,ATRIB(5).EQ.6,PRO5;
         ACT,,ATRIB(5).EQ.7,PRO6;
         ACT,,ATRIB(5).EQ.8,PRO7;
```

```
        ACT,,ATRIB(5).EQ.9,PRO8;
        ACT,,ATRIB(5).EQ.10,PRO9;
        ACT,,ATRIB(5).EQ.12,CHIP;
        ACT,,ATRIB(5).EQ.13,FTRA;
;
;
;
; ROUTINE, TESTING INFEED CURRENT PROCESS INVENTORY TOO LOW
; --------------------------------------------------------
;
;
INLO    ASSIGN,II=ATRIB(5)+LEVEL2,XX(1)=XX(II);  ACTUAL INFEED INV.CURRENT PRO
        ASSIGN,II=ATRIB(5)+LEVEL3,XX(2)=XX(II);  STOPPING LEVEL MIN.INFEED INV
        ASSIGN,II=ARRAY(24,ATRIB(5)),1;          INDEXING PREVIOUS PROCESS
         ACT,,XX(1).GE.XX(2),DEST;               INVENTORY TEST
         ACT,,ARRAY(15,II).GE.2,DEST;            PREVIOUS PROCESS ENDED
         ACT,,ARRAY(15,ATRIB(5)).GE.2,DEST;      CURRENT PROCESS ENDED
         ACT,,ARRAY(20,ATRIB(5)).GT.0,DEST;      OUTFEED ALREADY DOWN
         ACT,,ARRAY(19,ATRIB(5)).GT.0,DEST;      FLUX AROUND LIMIT
         ACT;
;
        ASSIGN,ARRAY(17,ATRIB(5))=TNOW;          STORE TIME
        ASSIGN,ARRAY(19,ATRIB(5))=1;             SET FLAG
        ASSIGN,ATRIB(1)=1,1;                     SET FLAG DECREASING RESOURCES
         ACT,,,AL;                               GOTO ALTERING
;
;
;
; ROUTINE, TEST INFEED INVENTORY NEXT PROCESS BACK TO NORMAL
; --------------------------------------------------------
;
;
INN     ASSIGN,ATRIB(6)=ARRAY(23,ATRIB(5));      INDEXING NEXT PROCESS
INN1    ASSIGN,II=ATRIB(6)+LEVEL2,XX(1)=XX(II);  INFEED INV.NEXT PROCESS
        ASSIGN,II=ATRIB(6)+LEVEL5,XX(2)=XX(II);  STARTUP INV.LEVEL
        ASSIGN,II=ATRIB(6),1;                    INDEXING NEXT PROCESS
         ACT,,ARRAY(15,II).EQ.0,DEST;            NEXT PROCESS NOT UP YET
         ACT,,XX(1).LT.XX(2),DEST;               INVENTORY TEST NEXT PROCESS
         ACT,,ARRAY(19,II).EQ.0,DEST;            FLUX AROUND LIMIT
         ACT,,ARRAY(20,II).GT.0,DEST;            OUTFEED ALREADY DOWN N.P.
          ACT;
;
        ASSIGN,XX(1)=TNOW-ARRAY(17,II);          CALC.DOWNTIME
        ASSIGN,ARRAY(19,II)=0,ARRAY(11,II)=ARRAY(11,II)+XX(1);ADD DOWNTIME
        ASSIGN,ATRIB(5)=II,ATRIB(1)=2;           SET FLAG INCREASING RESOURCES
        OPEN,ATRIB(5);                           OPEN GATE NEXT PROCESS
         ACT,,,AL;                               GOTO ALTERING RESOURCES
;
;
;
```

```
; ROUTINE, TEST OUTFEED INVENTORY CURRENT PROCESS IS TOO HIGH
; ----------------------------------------------------------
; (OUTFEED INVENTORY CURRENT PROCESS = INFEED INVENTORY NEXT PROCESS)
;
;
OUTH    ASSIGN,II=LEVEL2+ARRAY(23,ATRIB(5)),XX(1)=XX(II);INFEED INV.NEXT PROC
        ASSIGN,II=LEVEL4+ARRAY(23,ATRIB(5)),XX(2)=XX(II),1; MAX.INV.NEXT PROC
        ASSIGN,II=ARRAY(23,ATRIB(5)),1;           INDEXING NEXT PROCESS
         ACT,,ARRAY(15,II).EQ.Ø,DEST;             NEXT PROC.NOT UP YET
         ACT,,XX(1).LE.XX(2),DEST;                INVENTORY TEST NEXT PROC.
         ACT,,ARRAY(15,ATRIB(5)).GE.2,DEST;       CURRENT PROCESS ENDED
         ACT,,ARRAY(19,ATRIB(5)).GT.Ø,DEST;       INFEED ALREADY DOWN
         ACT,,ARRAY(2Ø,ATRIB(5)).GT.Ø,DEST;       FLUX AROUND LIMIT
         ACT;
;
        ASSIGN,ARRAY(17,II)=TNOW;                STORE TIME
        ASSIGN,ATRIB(1)=1,ARRAY(2Ø,ATRIB(5))=1;  SET FLAG DECREASE RESOURCES
         ACT,,,AL;
;
;
;
; ROUTINE, TEST OUTFEED INVENTORY PREVIOUS PROCESS BACK TO NORMAL
; --------------------------------------------------------------
; (INFEED INVENTORY CURRENT PROCESS = OUTFEED INVENTORY PREVIOUS PROCESS)
;
;
OUTN    ASSIGN,II=LEVEL2+ATRIB(5),XX(1)=XX(II); INV.CURRENT PROCESS
        ASSIGN,II=LEVEL6+ATRIB(5),XX(2)=XX(II); STARTUP LEVEL MAX.INV.
        ASSIGN,II=ARRAY(24,ATRIB(5)),1;          INDEXING PREVIOUS PROCESS
         ACT,,XX(1).GT.XX(2),DEST;               INVENTORY TEST MAX.INV.
         ACT,,ARRAY(15,II).GE.2,DEST;            PREV.PROCESS ALREADY ENDED
         ACT,,ARRAY(2Ø,II).NE.1,DEST;            FLUX AROUND LIMIT
         ACT,,ARRAY(19,II).GT.Ø,DEST;            INFEED INV.DOWN PREV.PROCESS
         ACT;
;
        ASSIGN,XX(1)=TNOW-ARRAY(17,ATRIB(5));    CALC.DOWNTIME
        ASSIGN,ARRAY(12,ATRIB(5))=ARRAY(12,ATRIB(5))+XX(1);  ADD DOWNTIME
        ASSIGN,ARRAY(2Ø,II)=Ø,ATRIB(5)=II;       SET FLAG,SET PROCESS
        ASSIGN,ATRIB(1)=2;                       SET FLAG INCREASING RESOURCES
        OPEN,ATRIB(5);                           OPEN GATE PREVIOUS PROCESS
         ACT,,,AL;                               GOTO ALTERING RESOURCES
;
;
;
; ROUTINE, TEST END OF PROCESS
; ---------------------------
;
;
```

```
END     GOON,1;
         ACT,,ATRIB(5).EQ.1,EFELL;              TEST IF FELLING
         ACT;
;
        ASSIGN,II=LEVEL2+ATRIB(5),XX(1)=XX(II); INVENTORY CURRENT PROCESS
        ASSIGN,II=ARRAY(24,ATRIB(5)),1;         INDEXING PREVIOUS PROCESS
         ACT,,XX(1).GT.Ø.ØØ1,DEST;              TEST INVENTORY CURRENT PROCESS
         ACT,,ARRAY(15,II).LT.2,DEST;           TEST PREVIOUS PROCESS FINISH
         ACT,,ARRAY(26,II).GT.Ø.ØØ1,DEST;       TEST INVENTORY IN TRANSIT
         ACT;
;
        ASSIGN,ARRAY(15,ATRIB(5))=2,ATRIB(7)=1; SET FLAG PROCESS FINISHED
         ACT,,,SEND;                            GOTO RESOURCE ADJUSTMENT
;
;
;
EFELL   ASSIGN,II=LEVEL1+1
        GOON,1;
         ACT,,XX(II).LT.XX(4),DEST;             TEST IF ALL TREES HARVESTED
         ACT;
;
        ASSIGN,ARRAY(15,1)=2,ATRIB(7)=1;        SET FLAG PROCESS FINISHED
;
;
;
SEND    ASSIGN,ATRIB(1)=1,2;                    SET FLAG DECREASING RESOURCE
         ACT,,,AL;
         ACT;
;
        GOON,1;
         ACT,,ATRIB(5).EQ.11,E4;                TESTING IF DISTRIBUTION
         ACT,,ATRIB(5).EQ.12,E2;                TESTING IF CHIPPING
         ACT,,ATRIB(5).EQ.13,E3;                TESTING IF FTRAPO
         ACT;
;
        ASSIGN,II=ARRAY(23,ATRIB(5)),ATRIB(5)=II,ATRIB(1)=2,1; INDEX NEXT PROC
         ACT,,,E5;
;
;
E2      ASSIGN,XX(8)=2;                         SET FLAG CHIPPING ENDED
         ACT,,,DEST;
E3      ASSIGN,XX(9)=2;                         SET FLAG FTRAPO ENDED
         ACT,,,DEST;
;
E4      ASSIGN,II=ARRAY(7,5),ATRIB(5)=II,ATRIB(1)=2,2;   INDEXING ROUTE 1
         ACT,,,E5;
         ACT;
        ASSIGN,II=ARRAY(7,6),ATRIB(5)=II;       INDEXING ROUTE 2
E5      GOON,1;
         ACT,,ARRAY(2Ø,ATRIB(5)).EQ.1,DEST;     NEXT PROCESS DOWN OUTFEED
         ACT,,ARRAY(19,ATRIB(5)).EQ.1,E6;       NEXT PROCESS DOWN INFEED
         ACT,,,DEST;                            DESTROY IF DONE
;
```

```
E6      ASSIGN,ARRAY(19,ATRIB(5))=Ø,2;          SET FLAG PROCESS UP INFEED
         ACT,,,AL;                              BRING BACK NEXT PROCESS
         ACT,,ATRIB(5).EQ.11,E7;                CONTINUE IF DISTRIBUTION
         ACT,,,DEST;                            OTHERWISE DESTROY
;
E7      OPEN,GATE11;                            OPEN GATE FOR DISTRIBUTION
         ACT,,,DEST;                            DESTROY IF DONE
;
;
;
;
;  ROUTINE TO CALCULATE THE REQUIRED STATISTICS
;  --------------------------------------------
;
;
STAT    ASSIGN,XX(5)=12Ø;
        ACT,USERF(12Ø);                         OUTPUT STATS TO PRINTER
        ASSIGN,ARRAY(15,ATRIB(5))=3,1;          SET FLAG PROC.COMPLETLY ENDED
        ASSIGN,ARRAY(26,ATRIB(5))=Ø,1;          SET INV.IN TRANSIT Ø
         ACT,,,GO19;                            RETURN TO MAIN PROCESSING
;
;
;
;  ROUTINE: MODELLING LOADING FUNCTION
;  ----------------------------------
;
;
LOAD    GOON,1;
         ACT,,ARRAY(9,ATRIB(5)).EQ.Ø,GO31;
         ACT,,ARRAY(9,ATRIB(5)).EQ.1,GO32;
;
;
GO31    ASSIGN,XX(6)=ARRAY(1,ATRIB(6))*ATRIB(3); CALC VARIABLE PROCESSING TIME
        ACT,XX(6);
        GOON,1;
         ACT,,,GO33;
;
GO32    ASSIGN,XX(5)=ATRIB(6);
        ACT,USERF(4);
;
;
GO33    GOON,1;
        ASSIGN,XX(1)=ATRIB(2)/ARRAY(8,ATRIB(6)); CALC.HOW MANY LOADS
        ASSIGN,XX(6)=XX(1)*ARRAY(2,ATRIB(6));     CALC.1ST CONST.TIME FACTOR
        ASSIGN,XX(6)=XX(6)+XX(1)*ARRAY(3,ATRIB(6)); CALC.2ND CONST TIME
        ACT,XX(6);                              1ST+2ND CONSTANT TIME FACTOR
;
        GOON,2;
         ACT,,,GO34;
         ACT,,ATRIB(5).LT.1Ø.OR.ATRIB(5).GE.12,GO13;RETURN TO MAIN PROCESSING
         ACT,,,DEST;
;
```

```
GO34    GOON,1;
        ASSIGN,XX(1)=ATRIB(2)/ARRAY(8,ATRIB(6)); CALC.HOW MANY LOADS
        ASSIGN,XX(1)=XX(1)*ARRAY(3,ATRIB(6));     CALC.2ND CONST.TIME FACTOR
        ACT,XX(1);                                2ND CONSTANT TIME FACTOR
        ASSIGN,ATRIB(4)=TNOW-ATRIB(4); MACH. TIME;
        ASSIGN,ARRAY(10,ATRIB(6))=ARRAY(10,ATRIB(6))+ATRIB(4); ADD.MACH.TIME
        FREE,ATRIB(6)/1,1;
         ACT,,,DEST;
;
;
; ROUTINE: TEST END OF SIMULATION
; ------------------------------
;
;
SEN1    ASSIGN,II=ARRAY(24,ATRIB(5)),ATRIB(1)=20,1;INDEXING PREVIOUS PROCESS
         ACT,,ARRAY(15,II).NE.3,WAI1;             TESTING IF PREV.PROC.ENDED
         ACT;
SEN2    GOON,1;
         ACT,,ARRAY(15,13).GT.0,BOTH;
         ACT,,,SEN5;
;
;
;
SEN3    ASSIGN,II=ARRAY(24,ATRIB(5)),ATRIB(1)=21,1;      INDEX PREV. PROCESS
         ACT,,ARRAY(15,II).NE.3,WAI1;                    TESTING IF PREV.PROC.ENDED
         ACT;
SEN4    GOON,1;
         ACT,,ARRAY(15,12).GT.0,BOTH;
         ACT,,,SEN5;
;
;
;
BOTH    ACCUMULATE,2,1,LAST,1;
SEN5    ASSIGN,ATRIB(1)=22,1;
        ASSIGN,ARRAY(15,14)=3;
         ACT,,NRUSE(32).GT.0,WAI1;              TEST IF ALL LOADERS ARE SHUT
         ACT,,NRUSE(33).GT.0,WAI1;              DOWN FOR STATS
         ACT,,NRUSE(34).GT.0,WAI1;
         ACT,,NRUSE(35).GT.0,WAI1;
         ACT,,NRUSE(36).GT.0,WAI1;
         ACT;
;
        ASSIGN,ATRIB(5)=14,XX(5)=120;           INDEXING FOR STATS
        ACT,USERF(120);
SEN6    TERMINATE,1;                            DEFINITIVE END OF SIMULATION
;
;
;
```

```
; ROUTINE: WAITING LOOPS
; ------------------------------------------------------
;
;
WAIT     FREE,ATRIB(1)/1,1;                    WAITING LOOP IF NO INFEED INV
          ACT,,ATRIB(6).EQ.Ø,WAØØ;             NO LOADER USED
          ACT;
         FREE,ATRIB(6)/1,1;                    FREE LOADER
WAØØ     GOON,1;
          ACT,,ATRIB(5).EQ.12,WAI3;            IS AVAILABLE
          ACT;
;
WAIØ     GOON,1;
         ACT,XX(1Ø);
         GOON,1;
          ACT,,,GO2Ø;                          RETURN TO MAIN PROCESS ROUT.
;
WAI3     FREE,CHIPTRA1/1,1;                    FREE CHIPTRA IF CHIP PROCESS
         ASSIGN,ARRAY(13,2)=1;                 RESET FLAG SEIZE PRIME TRAPO
          ACT,,,WAIØ;
;
;
;
WAI1     GOON,1;                               WAITING LOOP IF END OF SIMU,
         ACT,XX(1Ø);                           PROCESS IS NOT FINISHED
         GOON,1;
          ACT,,ATRIB(1).EQ.2Ø,SEN1;
          ACT,,ATRIB(1).EQ.21,SEN3;
          ACT,,ATRIB(1).EQ.22,SEN5;
;
;
;
;
;
; BEGINN OF FELLING PROCESS, SITE 1
; ================================================================
;
;
;
;
;
; ROUTINE OT ASSIGN AVAILABLE FELLER
; --------------------------------
;
;
FELL     ASSIGN,ATRIB(5)=1,ATRIB(6)=ARRAY(4,1),1;
          ACT,,NNRSC(1).GT.Ø,MA1;
          ACT,,NNRSC(2).GT.Ø,MA2;
          ACT,,NNRSC(3).GT.Ø,MA3;
          ACT,,NNRSC(4).GT.Ø,MA4;
          ACT,,,CL1;
;
;
```

```
CL1     CLOSE,GATE1;
A1      AWAIT(51),GATE1;
         ACT,,,FELL;
;
;
MA1     ASSIGN,ATRIB(1)=1;
         ACT,,,AW1;
MA2     ASSIGN,ATRIB(1)=2;
         ACT,,,AW1;
MA3     ASSIGN,ATRIB(1)=3;
         ACT,,,AW1;
MA4     ASSIGN,ATRIB(1)=4;
         ACT,,,AW1;
;
;
AW1     AWAIT(ATRIB(1)=1,50),ATRIB(1),1;
         ACT,,ATRIB(6).EQ.0,PROC;
         ACT;
;
F3      AWAIT(ATRIB(6)=1,50),ATRIB(6);   SEIZE LOADER
         ACT,,,PROC;
;
;
;
;
; BEGIN OF SECOND PROCESS
; ===============================================================
;
;
;  ROUTINE TO ASSIGN AVAILABLE RESOURCES AND GO TO PROCESSING
;  --------------------------------------------------------
;
;
PRO1    ASSIGN,ATRIB(5)=2,ATRIB(6)=ARRAY(4,2),1;
         ACT,,NNRSC(5).GT.0,MA5;
         ACT,,NNRSC(6).GT.0,MA6;
         ACT,,NNRSC(7).GT.0,MA7;
         ACT,,,CL2;
;
;
CL2     CLOSE,GATE2;
A2      AWAIT(52),GATE2;
         ACT,,,PRO1;
;
;
MA5     ASSIGN,ATRIB(1)=5;
         ACT,,,AW2;
MA6     ASSIGN,ATRIB(1)=6;
         ACT,,,AW2;
MA7     ASSIGN,ATRIB(1)=7;
         ACT,,,AW2;
;
;
```

```
AW2     AWAIT(ATRIB(1)=1,50),ATRIB(1),1;
         ACT,,ATRIB(6).EQ.0,PROC;
         ACT;
;
A3      AWAIT(ATRIB(6)=1,50),ATRIB(6);   SEIZE LOADER
         ACT,,,PROC;
;
;
;  SECOND PROCESS: DETECT, STARTUP-INVENTORY LEVEL REACHED
;  -------------------------------------------------------
;
;
        DETECT,XX(29),XP,XX(89),,1;
         ACT,,ARRAY(15,2).GT.0,DEST;
         ACT;
;
        ASSIGN,ATRIB(1)=2,ATRIB(5)=2,ATRIB(2)=5,ATRIB(3)=7;
        ASSIGN,ARRAY(15,2)=1,ARRAY(16,2)=TNOW,2;
         ACT,,,AL;
         ACT,,,SY13;
;
;
;
;
;
;
; BEGIN OF THIRD PROCESS
; *****************************************************************
;
;
;  ROUTINE TO ASSIGN AVAILABLE RESOURCES AND GO TO PROCESSING
;  ------------------------------------------------------------
;
;
PRO2    ASSIGN,ATRIB(5)=3,ATRIB(6)=ARRAY(4,3),1;
         ACT,,NNRSC(8).GT.0,MA8;
         ACT,,NNRSC(9).GT.0,MA9;
         ACT,,NNRSC(10).GT.0,MA10;
         ACT,,,CL3;
;
;
CL3     CLOSE,GATE3;
        AWAIT(53),GATE3;
         ACT,,,PRO2;
;
;
MA8     ASSIGN,ATRIB(1)=8;
         ACT,,,AW3;
MA9     ASSIGN,ATRIB(1)=9;
         ACT,,,AW3;
MA10    ASSIGN,ATRIB(1)=10;
         ACT,,,AW3;
```

```
;
;
AW3     AWAIT(ATRIB(1)=1,5Ø),ATRIB(1),1;
         ACT,,ATRIB(6).EQ.Ø,PROC;
         ACT;
;
        AWAIT(ATRIB(6)=1,5Ø),ATRIB(6);   SEIZE LOADER
         ACT,,,PROC;
;
;
;
;
;
;  THIRD PROCESS: DETECT, STARTUP-INVENTORY LEVEL REACHED
;  ---------------------------------------------------
;
;
        DETECT,XX(3Ø),XP,XX(9Ø),,1;
         ACT.,ARRAY(15,3).GT.Ø,DEST;
         ACT;
;
        ASSIGN,ATRIB(1)=2,ATRIB(5)=3,ATRIB(2)=8,ATRIB(3)=1Ø;
        ASSIGN,ARRAY(15,3)=1,ARRAY(16,3)=TNOW,2;
         ACT,,,AL;
         ACT,,,SY13;
;
;
;
;
;
;
;
;
; BEGIN OF FORTH PROCESS
; ================================================================
;
;
;  ROUTINE TO ASSIGN AVAILABLE RESOURCES AND GO TO PROCESSING
;  ---------------------------------------------------------
;
;
PRO3    ASSIGN,ATRIB(5)=4,ATRIB(6)=ARRAY(4,4),1;
         ACT,,NNRSC(11).GT.Ø,MA11;
         ACT,,NNRSC(12).GT.Ø,MA12;
         ACT,,NNRSC(13).GT.Ø,MA13;
         ACT,,,CL4;
;
;
CL4     CLOSE,GATE4;
        AWAIT(54),GATE4;
         ACT,,,PRO3;
;
;
```

```
MA11     ASSIGN,ATRIB(1)=11;
          ACT,,,AW4;
MA12     ASSIGN,ATRIB(1)=12;
          ACT,,,AW4;
MA13     ASSIGN,ATRIB(1)=13;
          ACT,,,AW4;
;
;
AW4      AWAIT(ATRIB(1)=1,5Ø),ATRIB(1),1;
          ACT,,ATRIB(6).EQ.Ø,PROC;
          ACT;
;
         AWAIT(ATRIB(6)=1,5Ø),ATRIB(6);   SEIZE LOADER
          ACT,,,PROC;
;
;
;
;
;
;  FORTH PROCESS: DETECT, STARTUP-INVENTORY LEVEL REACHED
;  -----------------------------------------------------
;
;
         DETECT,XX(31),XP,XX(91),,1;
          ACT,,ARRAY(15,4).GT.Ø,DEST;
          ACT;
;
         ASSIGN,ATRIB(1)=2,ATRIB(5)=4,ATRIB(2)=11,ATRIB(3)=13;
         ASSIGN,ARRAY(15,4)=1,ARRAY(16,4)=TNOW,2;
          ACT,,,AL;
          ACT,,,SY13;
;
;
;
;
;
;
;
;
; BEGIN OF FIFTH PROCESS
; ==============================================================
;
;
;  ROUTINE TO ASSIGN AVAILABLE RESOURCES AND GO TO PROCESSING
;  ---------------------------------------------------------
;
;
PRO4     ASSIGN,ATRIB(5)=5,ATRIB(6)=ARRAY(4,5),1;
          ACT,,NNRSC(14).GT.Ø,MA14;
          ACT,,NNRSC(15).GT.Ø,MA15;
          ACT,,NNRSC(16).GT.Ø,MA16;
          ACT,,,CL5;
```

```
;
;
CL5     CLOSE,GATE5;
        AWAIT(55),GATE5;
         ACT,,,PRO4;
;
;
MA14    ASSIGN,ATRIB(1)=14;
         ACT,,,AW5;
MA15    ASSIGN,ATRIB(1)=15;
         ACT,,,AW5;
MA16    ASSIGN,ATRIB(1)=16;
         ACT,,,AW5;
;
;
AW5     AWAIT(ATRIB(1)=1,50),ATRIB(1),1;
         ACT,,ATRIB(6).EQ.0,PROC;
         ACT;
;
        AWAIT(ATRIB(6)=1,50),ATRIB(6);   SEIZE LOADER
         ACT,,,PROC;
;
;
;
;
;
;  FITH PROCESS: DETECT, STARTUP-INVENTORY LEVEL REACHED
;  ----------------------------------------------------
;
;
        DETECT,XX(32),XP,XX(92),,1;
         ACT,,ARRAY(15,5).GT.0,DEST;
         ACT;
;
        ASSIGN,ATRIB(1)=2,ATRIB(5)=5,ATRIB(2)=14,ATRIB(3)=16;
        ASSIGN,ARRAY(15,5)=1,ARRAY(16,5)=TNOW,2;
         ACT,,,AL;
         ACT,,,SY13;
;
;
;
;
;
;
;
;
```

```
; BEGIN OF SIXT PROCESS
; ==============================================================
;
;
;  ROUTINE TO ASSIGN AVAILABLE RESOURCES AND GO TO PROCESSING
;  -------------------------------------------------------
;
;
PRO5     ASSIGN,ATRIB(5)=6,ATRIB(6)=ARRAY(4,6),1;
          ACT,,NNRSC(17).GT.Ø,MA17;
          ACT,,NNRSC(18).GT.Ø,MA18;
          ACT,,NNRSC(19).GT.Ø,MA19;
          ACT,,,CL6;
;
;
CL6      CLOSE,GATE6;
         AWAIT(56),GATE6;
          ACT,,,PRO5;
;
;
MA17     ASSIGN,ATRIB(1)=17;
          ACT,,,AW6;
MA18     ASSIGN,ATRIB(1)=18;
          ACT,,,AW6;
MA19     ASSIGN,ATRIB(1)=19;
          ACT,,,AW6;
;
;
AW6      AWAIT(ATRIB(1)=1,5Ø),ATRIB(1),1;
          ACT,,ATRIB(6).EQ.Ø,PROC;
          ACT;
;
         AWAIT(ATRIB(6)=1,5Ø),ATRIB(6);   SEIZE LOADER
          ACT,,,PROC;
;
;
;
;
;
;  SIXT PROCESS: DETECT, STARTUP-INVENTORY LEVEL REACHED
;  ----------------------------------------------------
;
;
         DETECT,XX(33),XP,XX(93),,1;
          ACT,,ARRAY(15,6).GT.Ø,DEST;
          ACT;
;
         ASSIGN,ATRIB(1)=2,ATRIB(5)=6,ATRIB(2)=17,ATRIB(3)=19;
         ASSIGN,ARRAY(15,6)=1,ARRAY(16,6)=TNOW,2;
          ACT,,,AL;
          ACT,,,SY13;
```

```
;
;
;
; BEGIN OF SEVENTH PROCESS
; ==============================================================
;
;
;  ROUTINE TO ASSIGN AVAILABLE RESOURCES AND GO TO PROCESSING
;  ----------------------------------------------------------
;
;
PRO6      ASSIGN,ATRIB(5)=7,ATRIB(6)=ARRAY(4,7),1;
           ACT,,NNRSC(20).GT.0,MA20;
           ACT,,NNRSC(21).GT.0,MA21;
           ACT,,NNRSC(22).GT.0,MA22;
           ACT,,,CL7;
;
;
CL7       CLOSE,GATE7;
          AWAIT(57),GATE7;
           ACT,,,PRO6;
;
;
MA20      ASSIGN,ATRIB(1)=20;
           ACT,,,AW7;
MA21      ASSIGN,ATRIB(1)=21;
           ACT,,,AW7;
MA22      ASSIGN,ATRIB(1)=22;
           ACT,,,AW7;
;
;
AW7       AWAIT(ATRIB(1)=1,50),ATRIB(1),1;
           ACT,,ATRIB(6).EQ.0,PROC;
           ACT;
;
          AWAIT(ATRIB(6)=1,50),ATRIB(6);   SEIZE LOADER
           ACT,,,PROC;
;
;
;
;
;
;  SEVENTH PROCESS: DETECT, STARTUP-INVENTORY LEVEL REACHED
;  -----------------------------------------------------
;
;
          DETECT,XX(34),XP,XX(94),,1;
           ACT,,ARRAY(15,7).GT.0,DEST;
           ACT;
;
```

```
        ASSIGN,ATRIB(1)=2,ATRIB(5)=7,ATRIB(2)=20,ATRIB(3)=22;
        ASSIGN,ARRAY(15,7)=1,ARRAY(16,7)=TNOW,2;
         ACT,,,AL;
         ACT,,,SY13;
;
;
;
;
; BEGIN OF EIGTH PROCESS
; ================================================================
;
;
;  ROUTINE TO ASSIGN AVAILABLE RESOURCES AND GO TO PROCESSING
;  ----------------------------------------------------------
;
;
PRO7    ASSIGN,ATRIB(5)=8,ATRIB(6)=ARRAY(4,8),1;
         ACT,,NNRSC(23).GT.0,MA23;
         ACT,,NNRSC(24).GT.0,MA24;
         ACT,,NNRSC(25).GT.0,MA25;
         ACT,,,CL8;
;
;
CL8     CLOSE,GATE8;
        AWAIT(58),GATE8;
         ACT,,,PRO7;
;
;
MA23    ASSIGN,ATRIB(1)=23;
         ACT,,,AW8;
MA24    ASSIGN,ATRIB(1)=24;
         ACT,,,AW8;
MA25    ASSIGN,ATRIB(1)=25;
         ACT,,,AW8;
;
;
AW8     AWAIT(ATRIB(1)=1,50),ATRIB(1),1;
         ACT,,ATRIB(6).EQ.0,PROC;
         ACT;
;
        AWAIT(ATRIB(6)=1,50),ATRIB(6);      SEIZE LOADER
         ACT,,,PROC;
;
;
;
;
;
```

```
; EIGTH PROCESS: DETECT, STARTUP-INVENTORY LEVEL REACHED
; -------------------------------------------------------
;
;
        DETECT,XX(35),XP,XX(95),,1;
         ACT,,ARRAY(15,8).GT.Ø,DEST;
         ACT;
;
        ASSIGN,ATRIB(1)=2,ATRIB(5)=8,ATRIB(2)=23,ATRIB(3)=25;
        ASSIGN,ARRAY(15,6)=8,ARRAY(16,8)=TNOW,2;
         ACT,,,AL;
         ACT,,,SY13;
;
;
;
; BEGIN OF NINETH PROCESS
; ====================================================================
;
;
; ROUTINE TO ASSIGN AVAILABLE RESOURCES AND GO TO PROCESSING
; ------------------------------------------------------------
;
;
PRO8    ASSIGN,ATRIB(5)=9,ATRIB(6)=ARRAY(4,9),1;
         ACT,,NNRSC(26).GT.Ø,MA26;
         ACT,,NNRSC(27).GT.Ø,MA27;
         ACT,,NNRSC(28).GT.Ø,MA28;
         ACT,,,CL9;
;
;
CL9     CLOSE,GATE9;
        AWAIT(59),GATE9;
         ACT,,,PRO8;
;
;
MA26    ASSIGN,ATRIB(1)=26;
         ACT,,,AW9;
MA27    ASSIGN,ATRIB(1)=27;
         ACT,,,AW9;
MA28    ASSIGN,ATRIB(1)=28;
         ACT,,,AW9;
;
;
AW9     AWAIT(ATRIB(1)=1,5Ø),ATRIB(1),1;
         ACT,,ATRIB(6).EQ.Ø,PROC;
         ACT;
;
        AWAIT(ATRIB(6)=1,5Ø),ATRIB(6);      SEIZE LOADER
         ACT,,,PROC;
```

```
;
;
;
;
;
;  NINETH PROCESS: DETECT, STARTUP-INVENTORY LEVEL REACHED
;  ---------------------------------------------------
;
;
        DETECT,XX(36),XP,XX(96),,1;
         ACT,,ARRAY(15,9).GT.Ø,DEST;
         ACT;
;
        ASSIGN,ATRIB(1)=2,ATRIB(5)=9,ATRIB(2)=26,ATRIB(3)=28;
        ASSIGN,ARRAY(15,9)=1,ARRAY(16,9)=TNOW,2;
         ACT,,,AL;
         ACT,,,SY13;
;
;
;
; BEGIN OF TENTH PROCESS
; ===============================================================
;
;
;  ROUTINE TO ASSIGN AVAILABLE RESOURCES AND GO TO PROCESSING
;  ---------------------------------------------------------
;
;
PRO9    ASSIGN,ATRIB(5)=1Ø,ATRIB(6)=ARRAY(4,1Ø),1;
         ACT,,NNRSC(29).GT.Ø,MA29;
         ACT,,NNRSC(3Ø).GT.Ø,MA3Ø;
         ACT,,NNRSC(31).GT.Ø,MA31;
         ACT,,,CL1Ø;
;
;
CL1Ø     CLOSE,GATE1Ø;
        AWAIT(6Ø),GATE1Ø;
         ACT,,,PRO9;
;
;
MA29    ASSIGN,ATRIB(1)=29;
         ACT,,,AW1Ø;
MA3Ø    ASSIGN,ATRIB(1)=3Ø;
         ACT,,,AW1Ø;
MA31    ASSIGN,ATRIB(1)=31;
         ACT,,,AW1Ø;
;
;
AW1Ø     AWAIT(ATRIB(1)=1,5Ø),ATRIB(1),1;
         ACT,,ATRIB(6).EQ.Ø,PROC;
         ACT;
;
```

```
        AWAIT(ATRIB(6)=1,50),ATRIB(6);       SEIZE LOADER
         ACT,,,PROC;
;
;
;
;
;
;  TENTH PROCESS: DETECT, STARTUP-INVENTORY LEVEL REACHED
;  -----------------------------------------------------
;
;
        DETECT,XX(37),XP,XX(97),,1;
         ACT,,ARRAY(15,10).GT.0,DEST;
         ACT;
;
        ASSIGN,ATRIB(1)=2,ATRIB(5)=10,ATRIB(2)=29,ATRIB(3)=31;
        ASSIGN,ARRAY(15,10)=1,ARRAY(16,10)=TNOW,2;
         ACT,,,AL;
         ACT,,,SY13;
;
;
;
;
;
;
;
;
;
;
;
;;
; BEGIN OF DISTRIBUTION FUNCTION
; ================================================================
;
;
;
SORT    ASSIGN,ATRIB(5)=11,1;
         ACT,,ATRIB(1).EQ.1,G047;            PROCESS ENDED,LAST ENTITY
         ACT;
;
;
        ASSIGN,XX(5)=103;                    INDEXING USERF
        ACT,USERF(103);                      COLECT INV.STATS
;
        ASSIGN,XX(5)=130;                    INDEXING USERF
        ACT,USERF(130);                      CALC INVENTORY
;
        GOON,1;
         ACT,,ARRAY(15,11).NE.1,SOR1;        PROCESS NOT UP YET
         ACT,,ARRAY(19,11).EQ.1,SOR1;        INFEED TOO LOW
         ACT,,ATRIB(2).EQ.0,SOR1;            NOTHING TO MOVE
         ACT;
```

```
;
;
AS47    ASSIGN,ATRIB(1)=ARRAY(4,11),ARRAY(26,11)=ARRAY(26,11)+ATRIB(2),1;
         ACT,,ATRIB(1).EQ.Ø,AS48;               TEST IF LOADER USED,NO=JUMP
         ACT;                                   YES=CONTINUE
        AWAIT(ATRIB(1)=1,5Ø),ATRIB(1),1;        SEIZE LOADER
;
AS48    ASSIGN,XX(5)=1Ø3;                         INDEXING USERFUNCTION
        ACT,USERF(1Ø3);                           COLCT INV.STATS
;
        GOON,2;
         ACT,,,END;                               TEST PROCESS ENDED
         ACT;
;
        GOON,1;
         ACT,,ARRAY(15,11).LT.2,GO46;
         ACT;
        ASSIGN,ATRIB(1)=1;                      INDEXING LAST ENTITY
;
GO46    GOON,2;
         ACT,,,INLO;                            TEST INFEED INV.CURRENT PROC
         ACT;
;
        GOON,2;
         ACT,,,OUTN;                            TEST OUTFEED INV.PREVIOUS P.
         ACT;
;
        GOON,1;                                 TEST IF LOADER IS USED
         ACT,,ARRAY(4,11).EQ.Ø,GO44;            NO LOADER,JUMP TO NEXT STEP
         ACT;                                   CONTINUE
;
        ASSIGN,ATRIB(4)=TNOW,1;                 MARK BEGIN PROCESSING
         ACT,,ARRAY(9,11).EQ.Ø,AS49;            MODELLING AVERAGE
         ACT,,ARRAY(9,11).EQ.1,GO42;            MODELLING USERF
;
AS49    ASSIGN,II=ARRAY(4,11),XX(6)=ARRAY(1,II)*ATRIB(3);CALC VAR.PROC.TIME
        ACT,XX(6);
        GOON,1;
         ACT,,,GO43;
;
GO42    GOON,1;
         ACT,USERF(11);
;
GO43    ASSIGN,II=ARRAY(4,11);                  INDEXING LOADER
        ASSIGN,XX(1)=ATRIB(2)/ARRAY(8,II); CALC # OF RUNS
        ASSIGN,XX(6)=XX(1)*ARRAY(2,II);         1ST CONSTANT TIME/RUN
        ASSIGN,XX(6)=XX(6)+XX(1)*ARRAY(3,II);   2ND CONSTANT TIME
        ACT,XX(6);
;
;
```

```
G044    ASSIGN,II=ARRAY(7,5)+LEVEL2,XX(II)=XX(II)+ATRIB(6);INVENTORIES INFEED
        ASSIGN,II=ARRAY(7,6)+LEVEL2,XX(II)=XX(II)+ATRIB(7);NEXT PROCESSES
        ASSIGN,ARRAY(26,11)=ARRAY(26,11)-ATRIB(2),XX(5)=105,1;USRF&TRANSITINV
        ACT,USERF(105);                         COLCT STATS INV.NEXT PROC
        ASSIGN,XX(5)=106;                       INDEXING USERF
        ACT,USERF(106);                         COLCT STATS INV.NEXT PROC
        ASSIGN,ATRIB(6)=ARRAY(7,5),2;           INDEXING ROUTE 1
         ACT,,,INN1;                            TEST INFEED INV.NEXT PROC1
         ACT;
;
        ASSIGN,ATRIB(6)=ARRAY(7,6),2;           INDEXING ROUTE 2
         ACT,,,INN1;                            TEST INFEED INV.NEXT PROC2
         ACT;
;
        GOON,1;                                 TEST IF LOADER USED
         ACT,,ARRAY(4,11).EQ.0,G045;            NO LOADER USED,JUMP STEP
         ACT;                                   CONTINUE
;
        ASSIGN,II=ARRAY(4,11),XX(1)=ATRIB(2)/ARRAY(8,II); INDEXING LOADER
        ASSIGN,XX(1)=XX(1)*ARRAY(3,II);         CALCULATE 2ND CONST.TIME
        ACT,XX(1);                              2ND CONSTANT TIME
;
        ASSIGN,ATRIB(4)=TNOW-ATRIB(4);          CALC MACHINE TIME
        ASSIGN,II=ARRAY(4,11);                  INDEXING LOADER
        ASSIGN,ARRAY(10,II)=ARRAY(10,II))+ATRIB(4);SUM MACHINE TIME
        ASSIGN,ARRAY(7,18)=ARRAY(7,18)+ATRIB(4); SUM MACH.TIME PROCESS
        ASSIGN,ATRIB(3)=II;                     INDEXING LOADER USED
        FREE,ATRIB(3)/1;                        FREE LOADER
;
G045    ASSIGN,II=LEVEL1+11;                    INDEXING XX-VAR
        ASSIGN,XX(II)=XX(II)+ARRAY(7,10);       SUM PROCESSED
        ASSIGN,ATRIB(2)=ARRAY(7,5),ATRIB(3)=ARRAY(7,6); INDEXING NEXT PROCS
        OPEN,ATRIB(2);                          OPEN GATE NEXT PROCESS #1
        OPEN,ATRIB(3);                          OPEN GATE NEXT PROCESS #2
         ACT,,,SORT;                            GOTO BEGIN SORT/DISTRIBUTION
;
;
;
G047    ASSIGN,ARRAY(15,11)=3;                  MARK PROCESS REALLY ENDED
        ASSIGN,XX(5)=120;                       INDEXING USERFUNCTION
        ACT,USERF(120);                         PRINT OUT STATS
        GOON,1;
         ACT,,,DEST;
;
SOR1    GOON,1;                                 THIS IS A WAITING LOOP FOR
        ACT,XX(10);                             DISTRIBUTION. EVERY XX(10)
        GOON,1;                                 WE CHECK IF WE CAN MAKE A
         ACT,,,SORT;                            DISTRIBUTION
;
;
;
```

```
;  DISTRIBUTION/SORTING: DETECT,STARTUP-INVENTORY LEVEL REACHED
;  ----------------------------------------------------------
;
;
        DETECT,XX(38),XP,XX(98),,1;
         ACT,,ARRAY(15,11).GT.Ø,DEST;           FLUX AROUND LIMIT
         ACT;
;
        ASSIGN,ARRAY(15,11)=1,ARRAY(16,11)=TNOW; SET VARIABLES
        OPEN,GATE11,1;                          OPEN GATE
         ACT,,,SORT;                            BEGIN SORTING
;
;
;
;  BEGIN OF CHIPPING OPERATION
;  ============================================================
;
;
CHIP    ASSIGN,ATRIB(5)=12,ATRIB(6)=ARRAY(4,12),1;
         ACT,,NNRSC(37).GT.Ø.AND.ARRAY(13,2).EQ.1.AND.NNRSC(39).GT.Ø,AW6Ø;
         ACT,,NNRSC(37).GT.Ø.AND.ARRAY(13,2).EQ.Ø,AW61;
         ACT;
;
CL62    CLOSE,GATE12;
        AWAIT(62),GATE12;
         ACT,,,CHIP;
;
;
AW6Ø    AWAIT(39),CHIPTRA1,1;                   SEIZE TRANSPORT UNIT
        ASSIGN,ARRAY(13,2)=Ø;                   SET FLAG PRIME TRAPO SEIZED
AW61    ASSIGN,ATRIB(1)=37;                     SET ATRIB
        AWAIT(37),ATRIB(1),1;                   SEIZE CHIPPER
         ACT,,ATRIB(6).EQ.Ø,PROC;
         ACT;
;
        AWAIT(ATRIB(6)=1,5Ø),ATRIB(6);
         ACT,,,PROC;
;
;
;
;
GO63    ASSIGN,ARRAY(1Ø,39)=ARRAY(1Ø,39)+ATRIB(4),2;ADD MACH.HRS.PRIME TRAPO
         ACT,,,GO67;                            FIRST CONTINUE, THEN GOTO
         ACT,,,CHIP;                            CHIPPING
;
GO67   ASSIGN,ARRAY(13,1)=ARRAY(13,1)+ATRIB(2),1;ADD TO INV.TRAILER
         ACT,,ATRIB(7).EQ.1,GO68;               CONTINUE IF LAST ENTITY
         ACT,,ARRAY(13,1).LT.ARRAY(8,39),DEST;  TRAILER NOT FULL YET
         ACT;                                   START ACTUAL SHIPPING
;
```

```
G068    ASSIGN,ATRIB(4)=TNOW,ARRAY(13,2)=1,1;    SET FLAGS & TEST COMBI
        ASSIGN,ATRIB(2)=ARRAY(13,1),ARRAY(13,1)=Ø,1; SET AMOUNT HAULED
         ACT,,ARRAY(6,4Ø).LE.Ø,FR6Ø;             TRACTOR COMBINATION Ø=NO
         ACT;
;
       AWAIT(4Ø),CHIPTRA2/1;                     SEIZE TRAKTOR
;
;
FR6Ø    ASSIGN,ATRIB(6)=TNOW;                    STORE TRACTOR TIME
        ACT,ARRAY(2,39);                         1.CONSTANT TIME
        GOON,1;
        ACT,ARRAY(3,39);                         2.CONSTANT TIME
        GOON,1;
        ACT,ARRAY(3,39);                         2.CONSTANT TIME
        FREE,CHIPTRA1/1,1;                       FREE TRAILER;
        ASSIGN,XX(1)=TNOW-ATRIB(4),1;   CALC.MACH.HRS.CHIPTRA1
        ASSIGN,ARRAY(1Ø,39)=ARRAY(1Ø,39)+XX(1),1;ADD MACH.HRS
         ACT,,ARRAY(6,4Ø).LE.Ø,G064;             TEST COMBINATION
         ACT;
        ASSIGN,ATRIB(6)=TNOW-ATRIB(6),ARRAY(1Ø,4Ø)=ARRAY(1Ø,4Ø)+ATRIB(6);ADD
        FREE,CHIPTRA2/1,1;                       FREE TRAKTOR          HRS
;
G064    GOON,1;
         ACT,,ATRIB(7).EQ.1,STAT;                LAST ENTITY,GOTO STATS
         ACT;
        OPEN,GATE12;
        ASSIGN,XX(5)=131,XX(15)=XX(15)+ATRIB(2); SUM SYSTEM HARVESTED
        ACT,USERF(131);                          DISPLAY AMOUNT HARVESTED
        GOON,1;
         ACT,,,DEST;
;
G065    GOON,1;
         ACT,,ATRIB(7).EQ.1,SEN1;                LAST ENTITY,GOTO END SIMU
         ACT,,,DEST;
;
;
;
;  CHIPPING: DETECT, STARTUP-INVENTORY LEVEL REACHED
;  ------------------------------------------------
;
;
        DETECT,XX(39),XP,XX(99),,1;
         ACT,,ARRAY(15,12).GT.Ø,DEST;
         ACT;
;
        ASSIGN,ATRIB(1)=2,ATRIB(5)=12,ATRIB(2)=37,ATRIB(3)=4Ø;
        ASSIGN,ARRAY(15,12)=1,ARRAY(16,12)=TNOW,ARRAY(13,2)=1;
        ALTER,CHIPTRA1,ARRAY(6,39);
        ALTER,CHIPTRA2,ARRAY(6,4Ø);
        OPEN,GATE12,2;
         ACT,,,AL;
         ACT,,,SY13;
```

```
;
;
;;
;
;
;  BEGIN OF FINAL TRANSPORT
;  ============================================================
;
;
;
FTRA     ASSIGN,ATRIB(1)=41,ATRIB(5)=13,ATRIB(6)=ARRAY(4,13),1;
          ACT,,ATRIB(7).EQ.1,SEN3;              LAST ENTITY
          ACT,,NNRSC(41).GT.Ø,AW13;             RESOURCE CAN BE SEIZED
          ACT;
;
;
CL13     CLOSE,GATE13;
A13      AWAIT(63),GATE13;
          ACT,,,FTRA;
;
;
AW13     AWAIT(41),FTRAPO1,1;
          ACT,,ATRIB(6).EQ.Ø,PROC;
          ACT;
;
A14      AWAIT(ATRIB(6)=1,5Ø),ATRIB(6),1;
          ACT,,,PROC;
;
;
AW5Ø     AWAIT(42),FTRAPO2,1;
         ASSIGN,ATRIB(6)=TNOW;
          ACT,,,GO14;
;
;
;
FTST     ASSIGN,XX(15)=XX(15)+ATRIB(2),XX(5)=131,1; CALC.AMOUNT HARVESTED
         ACT,USERF(131);                         DISPLAY AMOUNT HARVESTED
         GOON,1;
           ACT,,ARRAY(6,42).EQ.Ø,GO21;           NO FTRAPO2,RETURN MAIN ROUT.
           ACT;
         ASSIGN,ATRIB(6)=TNOW-ATRIB(6);          CALC.MACH.TIME FTRAPO2
         ASSIGN,ARRAY(1Ø,42)=ARRAY(1Ø,42)+ATRIB(6);ADD MACHINE TIME FTRAPO2
         FREE,FTRAPO2/1,1;                       FREE FTRAPO2
          ACT,,,GO21;                            RETURN TO MAIN ROUTINE
;
;
;
```

```
; FINAL TRANSPORT: DETECT, STARTUP-INVENTORY LEVEL REACHED
; --------------------------------------------------------
;
;
         DETECT,XX(40),XP,XX(100),,1;
          ACT,,ARRAY(15,13).GT.0,DEST;
          ACT;
;
         ASSIGN,ATRIB(1)=2,ATRIB(5)=13,ATRIB(2)=41,ATRIB(3)=42;
         ALTER,FTRAPO2,ARRAY(6,42);
         ASSIGN,ARRAY(15,13)=1,ARRAY(16,13)=TNOW,2;
          ACT,,,AL;
          ACT,,,SY13;
;
;
;
;
;
;
;
         END;
;
FIN;
```

APPENDIX A

2. Table 3: Contents of ATTRIBUTES

Attribute 1 = # of resource used

Attribute 2 = Load volume of current run

Attribute 3 = # of logs in current run & calculated processing time per tree

Attribute 4 = TNOW

Attribute 5 = # of current Process

Attribute 6 = # of loading resource used

Attribute 7 = Flag last entity

APPENDIX A

3. Table 4: Contents of XX(i) variables

XX-VARIABLES

XX (1) = HELP VARIABLE FOR CALCULATIONS
XX (2) = HELP VARIABLE FOR CALCULATIONS
XX (3) = HELP VARIABLE FOR CALCULATIONS
XX (4) = HOW MANY CU FT TO BE HARVESTED
XX (5) = INDEXING FORTAN USERFUNCTIONS
XX (6) = PROCESSING TIME FOR LOADING FUNCTION
XX (7) = INVENTORY MARK FOR CALCULATING INVENTORY
INCREASE DISTRIBUTION FTCT
XX (8) = FLAG FINAL TRAPO ENDED
XX (9) = FLAG CHIPPING PROCESS ENDED
XX (10) = TIME DELAY PARAMETER
XX (11) =
XX (12) =
XX (13) =
XX (14) =
XX (15) =
XX (16) = SUM CU FT PROCESSED PROCESS 1
XX (17) = SUM CU FT PROCESSED PROCESS 2
XX (18) = SUM CU FT PROCESSED PROCESS 3
XX (19) = SUM CU FT PROCESSED PROCESS 4
XX (20) = SUM CU FT PROCESSED PROCESS 5
XX (21) = SUM CU FT PROCESSED PROCESS 6
XX (22) = SUM CU FT PROCESSED PROCESS 7
XX (23) = SUM CU FT PROCESSED PROCESS 8
XX (24) = SUM CU FT PROCESSED PROCESS 9
XX (25) = SUM CU FT PROCESSED PROCESS 10
XX (26) = SUM CU FT PROCESSED SORTING
XX (27) = SUM CU FT PROCESSED CHIPPING
XX (28) = SUM CU FT PROCESSED FINAL TRANSPORT
XX (29) = INVENTORY PROCESS 2
XX (30) = INVENTORY PROCESS 3
XX (31) = INVENTORY PROCESS 4
XX (32) = INVENTORY PROCESS 5
XX (33) = INVENTORY PROCESS 6
XX (34) = INVENTORY PROCESS 7
XX (35) = INVENTORY PROCESS 8
XX (36) = INVENTORY PROCESS 9
XX (37) = INVENTORY PROCESS 10
XX (38) = INVENTORY SORTING
XX (39) = INVENTORY CHIPPING
XX (40) = INVENTORY FINAL TRANSPORT
XX (41) = MINIMUM INVENTORY LEVEL PROCESS 2
XX (42) = MINIMUM INVENTORY LEVEL PROCESS 3
XX (43) = MINIMUM INVENTORY LEVEL PROCESS 4
XX (44) = MINIMUM INVENTORY LEVEL PROCESS 5
XX (45) = MINIMUM INVENTORY LEVEL PROCESS 6
XX (46) = MINIMUM INVENTORY LEVEL PROCESS 7
XX (47) = MINIMUM INVENTORY LEVEL PROCESS 8
XX (48) = MINIMUM INVENTORY LEVEL PROCESS 9

XX (49) = MINIMUM INVENTORY LEVEL PROCESS 10
XX (50) = MINIMUM INVENTORY LEVEL SORTING
XX (51) = MINIMUM INVENTORY LEVEL CHIPPING
XX (52) = MINIMUM INVENTORY LEVEL FINAL TRANSPORT
XX (53) = MAXIMUM INVENTORY LEVEL PROCESS 2
XX (54) = MAXIMUM INVENTORY LEVEL PROCESS 3
XX (55) = MAXIMUM INVENTORY LEVEL PROCESS 4
XX (56) = MAXIMUM INVENTORY LEVEL PROCESS 5
XX (57) = MAXIMUM INVENTORY LEVEL PROCESS 6
XX (58) = MAXIMUM INVENTORY LEVEL PROCESS 7
XX (59) = MAXIMUM INVENTORY LEVEL PROCESS 8
XX (60) = MAXIMUM INVENTORY LEVEL PROCESS 9
XX (61) = MAXIMUM INVENTORY LEVEL PROCESS 10
XX (62) = MAXIMUM INVENTORY LEVEL SORTING
XX (63) = MAXIMUM INVENTORY LEVEL CHIPPING
XX (64) = MAXIMUM INVENTORY LEVEL FINAL TRANSPORT
XX (65) = STARTUP-INVENTORY LEVEL MINIMUM PROCESS 2
XX (66) = STARTUP-INVENTORY LEVEL MINIMUM PROCESS 3
XX (67) = STARTUP-INVENTORY LEVEL MINIMUM PROCESS 4
XX (68) = STARTUP-INVENTORY LEVEL MINIMUM PROCESS 5
XX (69) = STARTUP-INVENTORY LEVEL MINIMUM PROCESS 6
XX (70) = STARTUP-INVENTORY LEVEL MINIMUM PROCESS 7
XX (71) = STARTUP-INVENTORY LEVEL MINIMUM PROCESS 8
XX (72) = STARTUP-INVENTORY LEVEL MINIMUM PROCESS 9
XX (73) = STARTUP-INVENTORY LEVEL MINIMUM PROCESS 10
XX (74) = STARTUP-INVENTORY LEVEL MINIMUM SORTING
XX (75) = STARTUP-INVENTORY LEVEL MINIMUM CHIPPING
XX (76) = STARTUP-INVENTORY LEVEL MINIMUM FINAL TRANSPORT
XR (77) = STARTUP-INVENTORY LEVEL MAXIMUM PROCESS 2
XX (78) = STARTUP-INVENTORY LEVEL MAXIMUM PROCESS 3
XX (79) = STARTUP-INVENTORY LEVEL MAXIMUM PROCESS 4
XX (80) = STARTUP-INVENTORY LEVEL MAXIMUM PROCESS 5
XX (81) = STARTUP-INVENTORY LEVEL MAXIMUM PROCESS 6
XX (82) = STARTUP-INVENTORY LEVEL MAXIMUM PROCESS 7
XX (83) = STARTUP-INVENTORY LEVEL MAXIMUM PROCESS 8
XX (84) = STARTUP-INVENTORY LEVEL MAXIMUM PROCESS 9
XX (85) = STARTUP-INVENTORY LEVEL MAXIMUM PROCESS 10
XX (86) = STARTUP-INVENTORY LEVEL MAXIMUM SORTING
XX (87) = STARTUP-INVENTORY LEVEL MAXIMUM CHIPPING
XX (88) = STARTUP-INVENTORY LEVEL MAXIMUM FINAL TRANSPORT
XX (89) = STARTUP-INVENTORY LEVEL FOR PROCESS 2
XX (90) = STARTUP-INVENTORY LEVEL FOR PROCESS 3
XX (91) = STARTUP-INVENTORY LEVEL FOR PROCESS 4
XX (92) = STARTUP-INVENTORY LEVEL FOR PROCESS 5
XX (93) = STARTUP-INVENTORY LEVEL FOR PROCESS 6
XX (94) = STARTUP-INVENTORY LEVEL FOR PROCESS 7
XX (95) = STARTUP-INVENTORY LEVEL FOR PROCESS 8
XX (96) = STARTUP-INVENTORY LEVEL FOR PROCESS 9
XX (97) = STARTUP-INVENTORY LEVEL FOR PROCESS 10
XX (98) = STARTUP-INVENTORY LEVEL FOR SORTING
XX (99) = STARTUP-INVENTORY LEVEL FOR CHIPPING
XX (100)= STARTUP-INVENTORY LEVEL FOR FINAL TRANSPORT

APPENDIX A

4. Table 5: ARRAY description

ARRAY DESCRIPTION

ARRAY row #	Description
1	Average process. time per tree / machine types
2	Constant time per load / machine type
3	Constant time per one way haul / machine type
4	# of loader to use / process
5	# of material distribution / process
6	# of machines / machine type
7	Used for process #11, sorting (see next page)
8	machine load capacity / machine type
9	Time delays / process; Ø=build in, 1=USERF
1Ø	Cumulated productive machine hrs / machine type
11	Cum. inventory downtime hrs minimum / process
12	Cum. Inv. downtime hrs maximum / process
13	Used for process #12, chipping
14	
15	Flag process is activated Ø=no, 1=yes, 2= ended 3= ended & statistics / process
16	Start time process / process
17	Start invent. downtime (temp.buffer) / process
18	Cum. machine breakdown hrs / machine type
19	Flag process down because of minimum inv.
2Ø	Flag process down because of maximum inv.
21	Fixed costs per scheduled hr. / machine type
22	Variable cost per productive hr. / machine type
23	# of the proceeding process / process
24	# of the preceeding process / process
25	Cum. statistics for complete system stats
26	Inventory in transit / process

ARRAY line 7, Process #11, sorting

ARRAY line 7 column	Description
1	Sum route 1
2	Sum route 2
3	Percentage route 1
4	Percentage route 2
5	Proceeding process #, route 1
6	Proceeding process #, route 2
7	Internal inventory, route 1
8	Internal inventory, route 2
9	
10	Sum to move, temp. buffer
11	
12	
13	Flag inventory too high, route 1
14	Flag inventory too high, route 2
15	How much goes route 1, temp. buffer
16	How much goes route 2, temp. buffer
17	
18	Sum productive machine hrs., loader for sorting

APPENDIX A

5. Table 6: Machines & Processes

MACHINES & PROCESSES

Machine type	Process #
1	1, Felling
2	1, Felling
3	1, Felling
4	1, Felling
5	2
6	2
7	2
8	3
9	3
10	3
11	4
12	4
13	4
14	5
15	5
16	5
17	6
18	6
19	6
20	7
21	7
22	7
23	8
24	8
25	8
26	9
27	9
28	9
29	10
30	10
31	10
32	Loader
33	Loader
34	Loader
35	Loader
36	Loader
37	12, Chipper type 1
38	12, Chipper type 2
39	12, Primary transportation device
40	12, Secondary transportation device
41	13, Primary transportation device
42	13, Secondary transportation device

APPENDIX A

6. Initialization of the Network

INITIALIZATION OF THE NETWORK

At the beginning of each simulation run, the SLAM processor calls the FORTRAN written function INTLC.FOR (see Appendix B Listing 2). The User is prompted for the filename of the harvesting model created previously with the input front-end and that he wants to simulate (see also section V.B., output front-end). The file is then read into a variable array called USERARR within the COMMON Block USER3 (see Appendix B, Listing 1), that is identical to the local ARRAY block within the SLAM network. Care should be taken, if any additional FORTRAN inserts are made, to maintain the given COMMON Block and variable declarations given in Appendix B, Listing 1 (see also O'Reilly, 1984; Page 3-2 and 3-3). The values stored in USERARR are then copied to the ARRAY within the SLAM network and the XX(i) variables that contain the inventory information are initialized. For a description of the XX(i) variable content, see Appendix A, Listing 3; and for the ARRAY variables see Appendix A, Listing 4.

The model then performs some additional initializations, places the machine entities in their respective subnetworks and starts the first process by releasing the machine entities placed into the AWAIT-node attached to this process.

APPENDIX B

TABLE OF CONTENTS:

APPENDIX B

1. Listing, COMMON Block

```
C
C DEFINE COMMON BLOCK VARIABLE NAMES FOR ALL PROGRAMS:
C ==============================================
C
      COMMON/SCOM1/ATRIB(100),DD(100),DDL(100),DTNOW,II,MFA,MSTOP,NCLNR
     1,NCRDR,NPRNT,NNRUN,NNSET,NTAPE,SS(100),SSL(100),TNEXT,TNOW,XX(100)
      COMMON/USER1/DISARR(8,10),MCHARR(42,4,10),XXLEVEL(7),OUTFLAG
      COMMON/USER2/MCHNAMES(52),PROCNAMES(20),DISTRIBNAMES(4),FILENAME
     1,STSTR,SDSTR,ETSTR,EDSTR
      COMMON/USER3/USERARR(26,42),SIMRUN
C
C DEFINE COMMON VARIABLE TYPES FOR ALL PROGRAMS:
C ==========================================
C
      REAL*4 ATRIB,DD,DDL,DTNOW
      INTEGER*2 II,MFA,MSTOP,NCLNR,NPRNT,NNRUN,NNSET,NTAPE
      REAL*4 SS,SSL,TNEXT,TNOW,XX
      REAL*4 DISARR,MCHARR
      REAL*4 USERARR,SIMRUN
      INTEGER*2 XXLEVEL,OUTFLAG
      CHARACTER*20 MCHNAMES,PROCNAMES,DISTRIBNAMES,FILENAME
      CHARACTER*10 STSTR,SDSTR,ETSTR,EDSTR
C
C
```

APPENDIX B

2. Listing, INTCL.FOR

```
C*************************************************************
C*                                                           *
C*                 OREGON STATE UNIVERSITY                   *
C*                        JUNE 1986                          *
C*                                                           *
C*                 >>>  L O G S I M <<<                      *
C*                                                           *
C*     SIMULATION OF MECHANIZED LOG HARVESTING SYSTEMS       *
C*                                                           *
C*                                                           *
C*     DESIGNED BY : CHRISTOPH WIESE                         *
C*                   MASTERS CANDIDATE, DEP. OF INDUSTRIAL   *
C*                   ENGINEERING, OREGON STATE UNIVERSITY    *
C*                                                           *
C*     DESIGNED FOR: DEPARTMENT OF FOREST ENGINEERING        *
C*                   OREGON STATE UNIVERSITY                 *
C*                                                           *
C*                                                           *
C*     SUPERVISION : DR. ELDON OLSEN                         *
C*                   ASSOCIATE PROFESSOR, DEP. OF INDUSTRIAL *
C*                   ENGINEERING, OREGON STATE UNIVERSITY    *
C*                                                           *
C*                                                           *
C*************************************************************
C*                                                           *
C* FORTRAN USERFUNCTIONS: INICL.FOR  (INITIALIZE SIMULATION) *
C*                                                           *
C*                     31-MAY-87  18:55                      *
C*                                                           *
C*************************************************************
C
C
C
C
C
$INCLUDE:'PRCTL.FOR'
C
C
C
C
C INTERFACE TO DOS-SERVICES:
C ========================
C
      INTERFACE TO SUBROUTINE TIME (N,STR)
      CHARACTER*10 STR [NEAR,REFERENCE]
      INTEGER*2 N [VALUE]
      END
      INTERFACE TO SUBROUTINE DATE (N,STR)
      CHARACTER*10 STR [NEAR,REFERENCE]
      INTEGER*2 N [VALUE]
      END
```

```
C
C PROGRAM DECLARATION:
C --------------------
C
      SUBROUTINE INTLC
C
C
C COMMON BLOCK :
C --------------
C
$INCLUDE:'VARBLOCK.DOC'
C
C
C DEFINE LOCAL VARIABLES, NAMES & TYPE:
C -------------------------------------
C
      INTEGER*4 IANSWER,INDEX1,INDEX2,INDEX3,INDEX4,INDEX5
      INTEGER*4 INDEX6,INDEX7,INDEX8,INDEX9,IFN
      CHARACTER*20 CHRANSWER
      CHARACTER*10 TSTR
      REAL F1NUM,F2NUM,F3NUM,F4NUM,F5NUM,F6NUM,F7NUM
      LOGICAL*4 FILESTATUS
C
C
C BEGIN PROCESSING:
C -----------------
C
C
C CHECK WHICH SIMULATION RUN IS CURRENT:
C --------------------------------------
C
C
      IF (NNRUN.GT.1) GOTO 150
C
C
C
C OPENING SCREEN:
C ---------------
       WRITE(*,90)
90     FORMAT(////////////////////////////,
     1 10X,'*******************************************************'/,
     2 10X,'*                                                     *'/,
     3 10X,'*                  >>> L O G S I M <<<                *'/,
     4 10X,'*         MECHANIZED LOG HARVESTING SIMULATOR         *'/,
     5 10X,'*                  SIMULATION MODULE                  *'/,
     6 10X,'*******************************************************'/,
     7 ////)
```

```
C
C
100   WRITE(*,101)
101   FORMAT(5X,
     1'                  BEGIN OF SIMULATION'/5X,
     2'                  ===================' //5X,
     3'THIS FUNCTION READS A PREVIOSLY DEFINED MODEL INTO THE SLAM-'/5X,
     4'NETWORK AND SIMULATES IT.  ',///)
      WRITE(*,'(7X,A\)') 'DO YOU WISH TO CONTINUE (Y/N) ?  [Y]-----> '
      READ (*,'(A1)') CHRANSWER
      IF (CHRANSWER.EQ.'N') THEN
        XX(1)=0
        WRITE(*,102)
102     FORMAT(///10X'!!!!!   BYE-BYE, SEE YOU AGAIN  !!!!!'/
     1            10X'         >>> L O G S I M <<<'///)
        STOP ' '
      ELSE
        CONTINUE
      ENDIF
C
C
150     CALL INITREAD
C
C
        DO 420 INDEX1=1,3,1
         DO 410 INDEX2=1,42,1
          F1NUM=USERARR(INDEX1,INDEX2)
          CALL PUTARY(INDEX1,INDEX2,F1NUM)
410      CONTINUE
420     CONTINUE
        DO 440 INDEX1=4,5,1
         DO 430 INDEX2=1,13,1
          F1NUM=USERARR(INDEX1,INDEX2)
          CALL PUTARY(INDEX1,INDEX2,F1NUM)
430      CONTINUE
440     CONTINUE
        DO 450 INDEX1=1,42,1
          F1NUM=USERARR(6,INDEX1)
          CALL PUTARY(6,INDEX1,F1NUM)
450     CONTINUE
        DO 460 INDEX1=1,18,1
          F1NUM=USERARR(7,INDEX1)
          CALL PUTARY(7,INDEX1,F1NUM)
460     CONTINUE
        DO 480 INDEX1=8,10,1
         DO 470 INDEX2=1,42,1
          F1NUM=USERARR(INDEX1,INDEX2)
          CALL PUTARY(INDEX1,INDEX2,F1NUM)
470      CONTINUE
```

```
480     CONTINUE
        DO 483 INDEX1=11,12,1
         DO 482 INDEX2=1,13,1
          F1NUM=USERARR(INDEX1,INDEX2)
          CALL PUTARY(INDEX1,INDEX2,F1NUM)
482      CONTINUE
483     CONTINUE
        DO 485 INDEX1=13,14,1
         DO 484 INDEX2=1,2,1
          F1NUM=USERARR(INDEX1,INDEX2)
          CALL PUTARY(INDEX1,INDEX2,F1NUM)
484      CONTINUE
485     CONTINUE
        DO 487 INDEX1=15,17,1
         DO 486 INDEX2=1,13,1
          F1NUM=USERARR(INDEX1,INDEX2)
          CALL PUTARY(INDEX1,INDEX2,F1NUM)
486      CONTINUE
487     CONTINUE
        DO 488 INDEX2=1,42,1
         F1NUM=USERARR(18,INDEX2)
         CALL PUTARY(18,INDEX2,F1NUM)
488     CONTINUE
        DO 491 INDEX1=19,20,1
         DO 490 INDEX2=1,13,1
          F1NUM=USERARR(INDEX1,INDEX2)
          CALL PUTARY(INDEX1,INDEX2,F1NUM)
490      CONTINUE
491     CONTINUE
         DO 500 INDEX1=21,22,1
         DO 499 INDEX2=1,42,1
          F1NUM=USERARR(INDEX1,INDEX2)
          CALL PUTARY(INDEX1,INDEX2,F1NUM)
499      CONTINUE
500     CONTINUE
        DO 520 INDEX1=23,24,1
         DO 510 INDEX2=1,13,1
          F1NUM=USERARR(INDEX1,INDEX2)
          CALL PUTARY(INDEX1,INDEX2,F1NUM)
          F2NUM=GETARY(INDEX1,INDEX2)
510      CONTINUE
520     CONTINUE
        DO 524 INDEX2=1,6,1
          F1NUM=USERARR(25,INDEX2)
          CALL PUTARY(25,INDEX2,F1NUM)
524     CONTINUE
        DO 526 INDEX1=1,13,1
          F1NUM=USERARR(26,INDEX2)
          CALL PUTARY(26,INDEX2,F1NUM)
526     CONTINUE
C
C
C
```

```
      IF(NNRUN.GT.1) GOTO 1000
C
C
       WRITE(*,530)
530    FORMAT(///,10X'SIMULATION RESULTS SHOULD BE ROUTED TO:',/,
     1           10X'    SCREEN                = 1',/,
     2           10X'    SCREEN & PRINTER      = 2',//,
     3           10X'PLEASE ENTER CHOICE  -----> '\)
535    READ(*,'(BN,I2)')IANSWER
       IF (IANSWER.LT.1.OR.IANSWER.GT.2) THEN
         WRITE(*,540)
540      FORMAT(/,10X,'!!! CANNOT BE !!!'/,
     1           10X,'PLEASE ENTER AGAIN  -----> ',\)
         GOTO 535
       ELSE
         OUTFLAG=IANSWER
       ENDIF
C
C
      WRITE(*,541)
541   FORMAT(//,10X'HOW MANY SIMULATION RUNS DO YOU WANT?',/,
     1         10X'THE PRESET MAXIMUM IS 10.',/,
     2         10X'ENTER NUMBER OF RUNS       [1]-----> ',\)
544   READ(*,'(BN,I2)')IANSWER
      IF (IANSWER.LT.0.OR.IANSWER.GT.10) THEN
         WRITE(*,545)
545      FORMAT(/,10X,'!!! CANNOT BE !!!'/,
     1         10X,'PLEASE ENTER AGAIN        [1]-----> ',\)
         GOTO 535
      ELSEIF (IANSWER.EQ.0) THEN
        SIMRUN=1.0
      ELSE
        SIMRUN=IANSWER
      ENDIF
C
C
C
       WRITE(*,550)
550    FORMAT(///,10X,'PLEASE HIT  >RETURN<  TO START THE SIMULATION'/
     1           10X,'---------------------------------------------')
       READ(*,'(I2)')IANSWER
C
C
C
C
C
C DATTIM.FOR program - To access the date and time:
C ==============================================
C
C CALL DATE AND TIME (NOTE THAT THE STRING LENGTH IS PASSED
C AS THE FIRST ARGUMENT)
```

```
C
1000  CALL DATE (10,TSTR)
      SDSTR=TSTR
      CALL TIME (10,TSTR)
      STSTR=TSTR
      WRITE (*,*) 'TIME=',STSTR
      WRITE (*,*) 'DATE=',SDSTR
C
C
      XX(1)=1
C
C
C
9998  RETURN
      END
```

APPENDIX B

3. Listing, INITREAD.FOR

```
C************************************************************
C*                                                          *
C*                 OREGON STATE UNIVERSITY                  *
C*                        JUNE 1986                         *
C*                                                          *
C*                 >>> L O G S I M <<<                      *
C*                                                          *
C*     SIMULATION OF MECHANIZED LOG HARVESTING SYSTEMS      *
C*                                                          *
C*                                                          *
C*     DESIGNED BY : CHRISTOPH WIESE                        *
C*                   MASTERS CANDIDATE, DEP. OF INDUSTRIAL  *
C*                   ENGINEERING, OREGON STATE UNIVERSITY   *
C*                                                          *
C*     DESIGNED FOR: DEPARTMENT OF FOREST ENGINEERING       *
C*                   OREGON STATE UNIVERSITY                *
C*                                                          *
C*                                                          *
C*     SUPERVISION : DR. ELDON OLSEN                        *
C*                   ASSOCIATE PROFESSOR, DEP. OF INDUSTRIAL *
C*                   ENGINEERING, OREGON STATE UNIVERSITY   *
C*                                                          *
C*                                                          *
C************************************************************
C*                                                          *
C*   FORTRAN USERFUNCTIONS: INITREAD.FOR (READ IN VARIABLES) *
C*                                                          *
C*                      31-MAY-87  18:55                    *
C*                                                          *
C************************************************************
C
C
C
C
C
$INCLUDE:'PRCTL.FOR'
C
C PROGRAM DECLARATION:
C ===================
C
      SUBROUTINE INITREAD
C
C COMMON BLOCK :
C =============
C
$INCLUDE:'VARBLOCK.DOC'
C
C DEFINE LOCAL VARIABLES, NAMES & TYPE:
C ====================================
C
      INTEGER*4 IANSWER,INDEX1,INDEX2,INDEX3,INDEX4,INDEX5
      INTEGER*4 INDEX6,INDEX7,INDEX8,INDEX9
```

```
      CHARACTER*20 CHRANSWER
      REAL F1ANSWER,F2ANSWER,F3ANSWER
      LOGICAL*4 FILESTATUS
C
C
C  FUNCTION INITREAD.FOR, READ-IN THE SIMULATION MODEL AND INITIALIZE ALL VARS
C  ========================================================================
C
C
C
C INITIALIZATION OF ALL VARIABLES
C ==============================
C
C
C
      DO 110 INDEX1=1,100,1
        XX(INDEX1)=0.00
110   CONTINUE
C
      DO 114,INDEX1=1,26,1
      DO 112,INDEX2=1,42,1
        USERARR(INDEX1,INDEX2)=0.00
112   CONTINUE
114   CONTINUE
C
      DO 118 INDEX1=1,8,1
      DO 116 INDEX2=1,10,1
        DISARR(INDEX1,INDEX2)=0.00
116   CONTINUE
118   CONTINUE

C
      DO 124 INDEX1=1,42,1
      DO 122 INDEX2=1,4,1
      DO 120 INDEX3=1,10,1
        MCHARR(INDEX1,INDEX2,INDEX3)=0.00
120   CONTINUE
122   CONTINUE
124   CONTINUE
C
      DO 126 INDEX1=1,52,1
        MCHNAMES(INDEX1)=' '
126   CONTINUE
C
      DO 128 INDEX1=1,20,1
        PROCNAMES(INDEX1)=' '
128   CONTINUE
C
      DO 130 INDEX1=1,4,1
        DISTRIBNAMES(INDEX1)=' '
130   CONTINUE
C
C
```

```
C
C  CHECK WHICH CURRENT SIMULATION RUN
C  ---------------------------------
C
      IF (NNRUN.GT.1) GOTO 2070
C
C  RETRIVE THE MODEL FROM DISK DRIVE AND READ VARIABLE VALUES
C  ==========================================================
C
C
200   WRITE(*,202)
202   FORMAT(///,7X'FILENAME OF MODEL TO BE RETRIEVED?  -----> '\)
      READ(*,'(A20)')FILENAME
C
204   INQUIRE(FILE=FILENAME,EXIST=FILESTATUS)
      IF(.NOT.FILESTATUS) THEN
        WRITE(*,206)FILENAME
206     FORMAT(/,7X'!!!!  FILE: 'A' DOES NOT EXISTS !!!!!'/)
        WRITE(*,'(7X,A,\)')'INPUT NEW FILENAME                   -----> '
          READ(*,'(A20)')FILENAME
          GOTO 204
      ELSE
        CONTINUE
      ENDIF
C
C
      WRITE(*,207)
207   FORMAT(//10X,'!!!           PLEASE WAIT A MOMENT          !!!')
C
C
2070  OPEN(10,FILE=FILENAME,STATUS='OLD')
      REWIND 10
C
C
      READ(10,'(F8.1)') XX(1)
      READ(10,'(F8.1)') XX(2)
      READ(10,'(F8.1)') XX(3)
      READ(10,'(F8.0)') XX(4)
      READ(10,'(F8.1)') XX(5)
      READ(10,'(F8.1)') XX(6)
      READ(10,'(F8.1)') XX(7)
      READ(10,'(F8.1)') XX(8)
      READ(10,'(F8.1)') XX(9)
      READ(10,'(F8.4)') XX(10)
C
      DO 210 INDEX1=11,100,1
        READ(10,208) XX(INDEX1)
208     FORMAT(F8.1)
210   CONTINUE
```

```
C
C
      DO 213 INDEX1=1,3,1
      DO 212 INDEX2=1,42,1
        READ(10,211) USERARR(INDEX1,INDEX2)
211     FORMAT(F8.4)
212   CONTINUE
213   CONTINUE
C
      DO 2130 INDEX1=4,26,1
      DO 2120 INDEX2=1,42,1
        READ(10,2110) USERARR(INDEX1,INDEX2)
2110     FORMAT(F8.2)
2120  CONTINUE
2130  CONTINUE
C
C
C
      DO 222 INDEX1=1,8,1
       DO 220 INDEX2=1,10,1
        READ(10,219) DISARR(INDEX1,INDEX2)
219     FORMAT(F8.2)
220    CONTINUE
222   CONTINUE

C
      DO 230 INDEX1=1,42,1
       DO 228 INDEX2=1,4,1
        DO 226 INDEX3=1,10,1
          READ(10,224) MCHARR(INDEX1,INDEX2,INDEX3)
224       FORMAT(F8.2)
226     CONTINUE
228    CONTINUE
230   CONTINUE
C
      DO 234 INDEX1=1,52,1
        READ(10,232) MCHNAMES(INDEX1)
232     FORMAT(A)
234   CONTINUE
C
      DO 238 INDEX1=1,20,1
        READ(10,236) PROCNAMES(INDEX1)
236     FORMAT(A)
238   CONTINUE
C
      DO 242 INDEX1=1,4,1
        READ(10,240) DISTRIBNAMES(INDEX1)
240     FORMAT(A)
242   CONTINUE
C
      REWIND 10
      CLOSE(10,STATUS='KEEP')
```

```
C
C
C
      WRITE(*,300)
300   FORMAT(//,10X,
     1'!!!  MODEL HAS BEEN SUCCESSFULLY RETRIEVED  !!!')
C
C
9998  RETURN
      END
```

APPENDIX B

4. Listing, USERF.FOR

```
C***************************************************************
C*                                                             *
C*                  OREGON STATE UNIVERSITY                    *
C*                         JUNE 1986                           *
C*                                                             *
C*                  >>>  L O G S I M <<<                       *
C*                                                             *
C*     SIMULATION OF MECHANIZED LOG HARVESTING SYSTEMS         *
C*                                                             *
C*                                                             *
C*     DESIGNED BY : CHRISTOPH WIESE                           *
C*                   MASTERS CANDIDATE, DEP. OF INDUSTRIAL     *
C*                   ENGINEERING, OREGON STATE UNIVERSITY      *
C*                                                             *
C*     DESIGNED FOR: DEPARTMENT OF FOREST ENGINEERING          *
C*                   OREGON STATE UNIVERSITY                   *
C*                                                             *
C*                                                             *
C*     SUPERVISION : DR. ELDON OLSEN                           *
C*                   ASSOCIATE PROFESSOR, DEP. OF INDUSTRIAL   *
C*                   ENGINEERING, OREGON STATE UNIVERSITY      *
C*                                                             *
C*                                                             *
C***************************************************************
C*                                                             *
C*              FORTRAN USERFUNCTIONS:  USERF.FOR              *
C*                                                             *
C*                      31-MAY-87  18:55                       *
C*                                                             *
C***************************************************************
C
C
C
C
C
C
C
C COMPILER DIRECTIVES:
C ===================
C
$INCLUDE:'PRCTL.FOR'
C
C
C INTERFACE TO DOS SERVICES:
C ========================
C
      INTERFACE TO SUBROUTINE TIME (N,STR)
      CHARACTER*10 STR [NEAR,REFERENCE]
      INTEGER*2 N [VALUE]
      END
      INTERFACE TO SUBROUTINE DATE (N,STR)
      CHARACTER*10 STR [NEAR,REFERENCE]
```

```
      INTEGER*2 N [VALUE]
      END
C
C
C PROGRAM DECLARATION:
C -------------------
C
C
      FUNCTION USERF(IFN)
C
C COMMON BLOCK :
C -------------
C
$INCLUDE:'VARBLOCK.DOC'
C
C DEFINE LOCAL VARIABLES, NAMES & TYPE:
C -----------------------------------
C
      INTEGER*4 IANSWER,INDEX1,INDEX2,INDEX3,INDEX4,INDEX5
      INTEGER*4 INDEX6,INDEX7,INDEX8,INDEX9,IFN,MARKE
      CHARACTER*20 CHRANSWER
      CHARACTER*10 TSTR
      REAL F1NUM,F2NUM,F3NUM,F4NUM,F5NUM,F6NUM,F7NUM,F8MUM,F9NUM
      REAL F10NUM,F11NUM,F12NUM,F13NUM,F14NUM,F15NUM,F16NUM,F17NUM
      REAL F18NUM,F19NUM,F20NUM,F21NUM,F22NUM,F23NUM,F24NUM,F25NUM
      LOGICAL*4 FILESTATUS
C
C BEGIN PROCESSING:
C ----------------
C
      IF(XX(5).EQ.101) THEN
        GOTO 1000
      ELSEIF (XX(5).EQ.103) THEN
        GOTO 5000
      ELSEIF (XX(5).EQ.104) THEN
        GOTO 5100
      ELSEIF (XX(5).EQ.105) THEN
        GOTO 5200
      ELSEIF (XX(5).EQ.106) THEN
        GOTO 5300
      ELSEIF (XX(5).EQ.110) THEN
        GOTO 2000
      ELSEIF (XX(5).EQ.111) THEN
        GOTO 2000
      ELSEIF (XX(5).EQ.120) THEN
        GOTO 4000
      ELSEIF (XX(5).EQ.130) THEN
        GOTO 7000
      ELSEIF (XX(5).EQ.131) THEN
        GOTO 7500
      ELSE
        WRITE(*,20)IFN,XX(5)
```

```
20      FORMAT(' !!!!!!!!  USERFUNCTION 'I5' IS NOT DEFINED !!!!!',/
     1         '                  XX(5)='F8.3,/,
     2         '                PRESS RETURN TO CONTINUE')
        USERF=0
        GOTO 9998
      ENDIF
C
C
C
C
C  USERF(101), ASSIGN TREE/LOG/PAPER/PULP DISTRIBUTION PARAMETERS
C  ===========================================================
C
C
1000  USERF=0
C
C GENERAL PROCEDURES
C ------------------
C
      INDEX9=ATRIB(5)
      INDEX7=0
C
C
C
C
C TESTING & SET FLAG IF PREVIOUS PROCESS ENDED:
C
      IF(ATRIB(5).NE.1) THEN
        INDEX2=GETARY(24,INDEX9)
        INDEX3=GETARY(15,INDEX2)
        F1NUM=GETARY(26,INDEX2)
        IF(INDEX3.LT.2) THEN
          INDEX1=0
        ELSEIF(INDEX3.GE.2.AND.F1NUM.GT.0.001) THEN
          INDEX1=0
        ELSE
          INDEX1=1
        ENDIF
      ELSE
        INDEX1=0
      ENDIF
C
C     TESTING IF INVENTORY IS AVAILABLE:
      IF(INDEX9.NE.1) THEN
        INDEX3=XXLEVEL(2)+ATRIB(5)
        F1NUM=XX(INDEX3)
      ELSE
        INDEX3=XXLEVEL(1)+ATRIB(5)
        F1NUM=XX(4)-XX(INDEX3)
      ENDIF
      IF (F1NUM.LE.0) THEN
        ATRIB(2)=0
        ATRIB(3)=0
```

```
        RETURN
      ELSE
        CONTINUE
      ENDIF
C
C     GET BATCHSIZE & ROUTE ACCORDINGLY
      INDEX6=ATRIB(1)
      F2NUM=GETARY(8,INDEX6)
      IF(F2NUM.EQ.99999)THEN
        GOTO 1300
      ELSE
        CONTINUE
      ENDIF
C
C
C  SET BATCHSIZE FOR A NORMAL CAPACITY
C  ---------------------------------
C
C     TESTING IF BATCHSIZE GREATER INVENTORY:
      IF(ATRIB(5).EQ.11) THEN
        F2NUM=GETARY(7,10)
      ENDIF
      F3NUM=F1NUM-F2NUM
      IF (ATRIB(5).EQ.1.AND.F3NUM.LT.0) THEN
        F4NUM=F1NUM
        INDEX7=1
      ELSEIF (F3NUM.LT.0.AND.INDEX1.EQ.1)THEN
        F4NUM=F1NUM
        INDEX7=1
      ELSEIF (F3NUM.GE.0)THEN
        F4NUM=F2NUM
      ELSE
        ATRIB(2)=0
        ATRIB(3)=0
        RETURN
      ENDIF
C
C     INDEXING THE DISARR-COLUMS ACCORDING TO DISTRIBUTION SPECIFIED:
1025  INDEX1=GETARY(5,INDEX9)
      IF (INDEX1.EQ.1) THEN
        INDEX2=1
        INDEX3=2
      ELSEIF (INDEX1.EQ.2) THEN
        INDEX2=3
        INDEX3=4
      ELSEIF (INDEX1.EQ.3) THEN
        INDEX2=5
        INDEX3=6
      ELSEIF (INDEX1.EQ.4) THEN
        INDEX2=7
        INDEX3=8
```

```
      ELSE
        CONTINUE
      ENDIF
C
C     SET INITIAL VALUES:
      ATRIB(3)=0
      ATRIB(2)=0
      F5NUM=0
      F6NUM=0
      INDEX4=0
C
C     LOOP TO CREATE BATCHSIZE:
C
C     F6NUM=CURRENT BATCH SIZE CALCULATED
C     F5NUM=TESTING VALUE SUM CUFT
C     F4NUM=BATCHSIZE
C     F3NUM=SAMPLE
C     F2NUM=UPPERBOUND
C     F1NUM=LOWERBOUND
C     INDEX4=NUMBER OF TREES
C
1050  F3NUM=UNFRM(0.0,100.0,6)
      F1NUM=0
C
      DO 1100 INDEX8=1,10,1
        F2NUM=DISARR(INDEX2,INDEX8)
       IF (F3NUM.GE.F1NUM.AND.F3NUM.LE.F2NUM) THEN
          F5NUM=F5NUM+DISARR(INDEX3,INDEX8)
          INDEX4=INDEX4+1
          GOTO 1200
        ELSE
          F1NUM=F2NUM
        ENDIF
1100  CONTINUE
C
1200  IF (F5NUM.LE.F4NUM) THEN
        F6NUM=F5NUM
        GOTO 1050
      ELSEIF (F5NUM.GT.F4NUM) THEN
        ATRIB(2)=F6NUM
        ATRIB(3)=INDEX4-1
      ELSE
        CONTINUE
      ENDIF
C
      F1NUM=F4NUM-F6NUM
      IF (ATRIB(5).EQ.11) THEN
        CONTINUE
      ELSEIF (F1NUM.GT.0.AND.INDEX7.EQ.1) THEN
        ATRIB(2)=F4NUM
        ATRIB(3)=ATRIB(3)+1
```

```
      ELSE
        CONTINUE
      ENDIF
C
C
C
      RETURN
C
C
C
C
C SET BATCHSIZE FOR CAPACITY=99999. SINGLE TREE
C ------------------------------------------
C
C INDEXING THE DISARR-COLUMS ACCORDING TO DISTRIBUTION SPECIFIED:
C
1300  INDEX5=INDEX1
      INDEX1=GETARY(5,INDEX9)
      IF (INDEX1.EQ.1) THEN
        INDEX2=1
        INDEX3=2
      ELSEIF (INDEX1.EQ.2) THEN
        INDEX2=3
        INDEX3=4
      ELSEIF (INDEX1.EQ.3) THEN
        INDEX2=5
        INDEX3=6
      ELSEIF (INDEX1.EQ.4) THEN
        INDEX2=7
        INDEX3=8
      ELSE
        CONTINUE
      ENDIF
C
C     SET INITIAL VALUES:
      ATRIB(3)=0
      ATRIB(2)=0
      F5NUM=0
      F6NUM=0
C
C     LOOP TO CREATE BATCHSIZE:
C
C     F5NUM=CACLULATED BATCHSIZE
C     F3NUM=SAMPLE
C     F2NUM=UPPERBOUND
C     F1NUM=LOWERBOUND
C
1350  F3NUM=UNFRM(0.0,100.0,6)
      F1NUM=0
```

```
C
      DO 14ØØ INDEX8=1,1Ø,1
        F2NUM=DISARR(INDEX2,INDEX8)
        IF (F3NUM.GE.F1NUM.AND.F3NUM.LE.F2NUM) THEN
          F5NUM=DISARR(INDEX3,INDEX8)
          GOTO 145Ø
        ELSE
          F1NUM=F2NUM
        ENDIF
14ØØ  CONTINUE
C
C
C     TESTING IF BATCHSIZE GREATER INVENTORY:
145Ø  IF(INDEX9.NE.1) THEN
        INDEX3=XXLEVEL(2)+ATRIB(5)
        F1NUM=XX(INDEX3)
      ELSE
        INDEX3=XXLEVEL(1)+ATRIB(5)
        F1NUM=XX(4)-XX(INDEX3)
      ENDIF
C
      F3NUM=F1NUM-F5NUM
      IF (ATRIB(5).EQ.1.AND.F3NUM.LT.Ø) THEN
        ATRIB(2)=F1NUM
        ATRIB(3)=1
      ELSEIF (F3NUM.LT.Ø.AND.INDEX5.EQ.1)THEN
        ATRIB(2)=F1NUM
        ATRIB(3)=1
      ELSEIF (F3NUM.GE.Ø)THEN
        ATRIB(2)=F5NUM
        ATRIB(3)=1
      ELSE
        ATRIB(2)=Ø
        ATRIB(3)=Ø
      ENDIF
C
C
C
C
      RETURN
C
C
C  MACHINE BREAKDOWN, ASSIGNMENT OF THE PARAMETERS
C  ===============================================
C
C
2ØØØ  INDEX9=ATRIB(1)
      IF (MCHARR(INDEX9,1,1).EQ.Ø) THEN
        USERF=Ø
        RETURN
      ENDIF
```

```
C
      IF (XX(5).EQ.110) THEN
        INDEX8=2
        INDEX7=1
      ELSEIF (XX(5).EQ.111) THEN
        INDEX8=4
        INDEX7=3
      ELSE
        CONTINUE
      ENDIF
C
C
2050  F3NUM=UNFRM(0.0,100.0,6)
      F1NUM=0
C
      DO 2100 INDEX1=1,10,1
        F2NUM=MCHARR(INDEX9,INDEX7,INDEX1)
        IF (F3NUM.GE.F1NUM.AND.F3NUM.LE.F2NUM) THEN
          F5NUM=MCHARR(INDEX9,INDEX8,INDEX1)
          GOTO 2200
        ELSE
          F1NUM=F2NUM
        ENDIF
2100  CONTINUE
C
2200  USERF=F5NUM
      RETURN
C
C
C
C USERFUNCTION TO DISPLAY/PRINTOUT THE SIMULATION RESULTS
C ======================================================
C
C OPEN THE APPROPRIATE OUTPUT DEVICES:
C -----------------------------------
C
C
4000  USERF=0
C
      IF (ATRIB(5).EQ.1.AND.OUTFLAG.GT.1) OPEN(15,FILE='LPT1')
C
C
      IF (ATRIB(5).EQ.1) THEN
        WRITE(*,4050)FILENAME
        IF (OUTFLAG.GT.1) WRITE(15,4050)FILENAME
4050    FORMAT(////,
     1  10X,'**********************************************************'/,
     2  10X,'*                                                        *'/,
     3  10X,'*                  >>> L O G S I M <<<                   *'/,
     4  10X,'*                  SIMULATION  RESULTS                   *'/,
     5  10X,'*                                                        *'/,
     6  10X,'**********************************************************'/,
     7  //2X'SIMULATION MODEL USED: ',A,/,2X'**********************'/)
```

```
C
        CALL DATE (10,TSTR)
        WRITE(*,*) ' DATE= ',TSTR
        IF (OUTFLAG.GT.1) WRITE(15,*) ' COMPUTER DATE: ',TSTR
        CALL TIME (10,TSTR)
        WRITE(*,*) ' TIME= ',TSTR
        IF (OUTFLAG.GT.1) WRITE(15,*) ' COMPUTER TIME: ',TSTR
C
        WRITE(*,4052)NNRUN,SIMRUN
        IF(OUTFLAG.GT.1)WRITE(15,4052)NNRUN,SIMRUN
4052    FORMAT('  SIMULATION RUN 'I2' OF 'F3.0)
C
      ELSEIF (ATRIB(5).EQ.14) THEN
        GOTO 4500
      ELSE
        CONTINUE
      ENDIF
C
C
      INDEX9=ATRIB(5)

C
C
      WRITE(*,4100)INDEX9,PROCNAMES(INDEX9)
      IF (OUTFLAG.GT.1) WRITE(15,4100)INDEX9,PROCNAMES(INDEX9)
4100  FORMAT(/////,10X,'PROCESS NO.'I2' : ',A/,
     1             10X,'===============',//)
C
C
C
      F1NUM=GETARY(16,INDEX9)
      F2NUM=TNOW
      F3NUM=F2NUM-F1NUM
      F6NUM=GETARY(11,INDEX9)
      F7NUM=GETARY(12,INDEX9)
      F8NUM=((F6NUM+F7NUM)/F3NUM)*100
C
      IF(INDEX9.EQ.1) THEN
        INDEX1=USERARR(6,1)+USERARR(6,2)+USERARR(6,3)+USERARR(6,4)
        F9NUM=GETARY(18,1)+GETARY(18,2)+GETARY(18,3)+GETARY(18,4)
        F25NUM=GETARY(10,1)
        F24NUM=GETARY(10,2)
        F23NUM=GETARY(10,3)
        F22NUM=GETARY(10,4)
        F5NUM=F25NUM+F24NUM+F23NUM+F22NUM
        F16NUM=GETARY(10,1)*USERARR(22,1)+
     1          GETARY(10,2)*USERARR(22,2)+
     2          GETARY(10,3)*USERARR(22,3)+
     3          GETARY(10,4)*USERARR(22,4)+
     4          USERARR(6,1)*F3NUM*USERARR(21,1)+
     5          USERARR(6,2)*F3NUM*USERARR(21,2)+
     6          USERARR(6,3)*F3NUM*USERARR(21,3)+
     7          USERARR(6,4)*F3NUM*USERARR(21,4)
```

```
      F11NUM=0
      F12NUM=0
      F13NUM=0
      F14NUM=0
      F15NUM=0
     ELSEIF(INDEX9.GE.2.AND.INDEX9.LE.10) THEN
      INDEX5=INDEX9*3-1
      INDEX6=INDEX5+1
      INDEX7=INDEX5+2
      INDEX1=USERARR(6,INDEX5)+USERARR(6,INDEX6)+USERARR(6,INDEX7)
      F9NUM=GETARY(18,INDEX5)+GETARY(18,INDEX6)+GETARY(18,INDEX7)
      F5NUM=GETARY(10,INDEX5)+GETARY(10,INDEX6)+GETARY(10,INDEX7)
      F16NUM=GETARY(10,INDEX5)*USERARR(22,INDEX5)+
     1        GETARY(10,INDEX6)*USERARR(22,INDEX6)+
     2        GETARY(10,INDEX7)*USERARR(22,INDEX7)+
     3        USERARR(6,INDEX5)*F3NUM*USERARR(21,INDEX5)+
     4        USERARR(6,INDEX6)*F3NUM*USERARR(21,INDEX6)+
     5        USERARR(6,INDEX7)*F3NUM*USERARR(21,INDEX7)
      F11NUM=CCAVG(INDEX9)
      F12NUM=CCMAX(INDEX9)
      F13NUM=CCMIN(INDEX9)
      F14NUM=CCSTD(INDEX9)
      F15NUM=CCNUM(INDEX9)
     ELSEIF(INDEX9.EQ.11) THEN
      INDEX4=USERARR(4,11)
      IF(INDEX4.GT.0) THEN
        INDEX1=1
        F5NUM=GETARY(7,18)
        F16NUM=F9NUM*USERARR(21,INDEX4)+F9NUM*USERARR(22,INDEX4)
        F17NUM=F16NUM/F19NUM
      ELSE
        INDEX1=0
        F5NUM=0
        F16NUM=0
        F17NUM=0
      ENDIF
      F4NUM=0
      F9NUM=0
      F10NUM=0
      F20NUM=0
      F18NUM=0
      INDEX8=XXLEVEL(1)+INDEX9
      F19NUM=XX(INDEX8)
      F11NUM=CCAVG(INDEX9)
      F12NUM=CCMAX(INDEX9)
      F13NUM=CCMIN(INDEX9)
      F14NUM=CCSTD(INDEX9)
      F15NUM=CCNUM(INDEX9)
      GOTO 4105
```

```
      ELSEIF(INDEX9.EQ.12) THEN
        INDEX1=USERARR(6,37)+USERARR(6,39)+USERARR(6,40)
        F9NUM=GETARY(18,37)+GETARY(18,39)+GETARY(18,40)
        F5NUM=GETARY(10,37)+GETARY(10,39)+GETARY(10,40)
        F16NUM=GETARY(10,37)*USERARR(22,37)+
     1         GETARY(10,39)*USERARR(22,39)+
     2         GETARY(10,40)*USERARR(22,40)+
     3         USERARR(6,37)*F3NUM*USERARR(21,37)+
     4         USERARR(6,39)*F3NUM*USERARR(21,39)+
     5         USERARR(6,40)*F3NUM*USERARR(21,40)
        F11NUM=CCAVG(INDEX9)
        F12NUM=CCMAX(INDEX9)
        F13NUM=CCMIN(INDEX9)
        F14NUM=CCSTD(INDEX9)
        F15NUM=CCNUM(INDEX9)
      ELSEIF (INDEX9.EQ.13) THEN
        INDEX1=USERARR(6,41)+USERARR(6,42)
        F9NUM=GETARY(18,41)+GETARY(18,42)
        F5NUM=GETARY(10,41)+GETARY(10,42)
        F16NUM=GETARY(10,41)*USERARR(22,41)+
     1         GETARY(10,42)*USERARR(22,42)+
     2         USERARR(6,41)*F3NUM*USERARR(21,41)+
     3         USERARR(6,42)*F3NUM*USERARR(21,42)
        F11NUM=CCAVG(INDEX9)
        F12NUM=CCMAX(INDEX9)
        F13NUM=CCMIN(INDEX9)
        F14NUM=CCSTD(INDEX9)
        F15NUM=CCNUM(INDEX9)
      ELSE
        CONTINUE
      ENDIF
C
      F4NUM=INDEX1*F3NUM
      F10NUM=(F5NUM/F4NUM)*100
      F20NUM=((F9NUM+F5NUM)/F4NUM)*100
      INDEX8=XXLEVEL(1)+INDEX9
      F19NUM=XX(INDEX8)
      F17NUM=F16NUM/F19NUM
      F18NUM=F16NUM/F4NUM
      USERARR(25,1)=USERARR(25,1)+F4NUM
      USERARR(25,2)=USERARR(25,2)+F9NUM
      USERARR(25,3)=USERARR(25,3)+F5NUM
      USERARR(25,4)=USERARR(25,4)+F16NUM
C
C
4105  WRITE(*,4110)F1NUM,F11NUM,F2NUM,F12NUM,F3NUM,F13NUM,F6NUM,F14NUM
      IF(OUTFLAG.GT.1) WRITE(15,4110)F1NUM,F11NUM,F2NUM,F12NUM,F3NUM,
     1 F13NUM,F6NUM,F14NUM
4110  FORMAT(2X,'TIME BEGIN OF PROCESS   :',E13.7,
     1        2X,'AVERAGE INVENTORY       :',E13.7,/,
     2        2X,'TIME END OF PROCESS     :',E13.7,
     3        2X,'MAXIMUM INVENTORY       :',E13.7,/,
     4        2X,'DURATION OF PROCESS     :',E13.7,
```

```
     5        2X,'MINIMUM INVENTORY       :',E13.7,/,
     6        2X,'TIME INVENTORY TOO LOW  :',E13.7,
     7        2X,'STD.DEV.INVENTORY       :',E13.7)
      WRITE(*,4150)F7NUM,F15NUM,F8NUM,F19NUM,INDEX1,F16NUM,F4NUM,F17NUM
      IF(OUTFLAG.GT.1) WRITE(15,4150)F7NUM,F15NUM,F8NUM,F19NUM,INDEX1,
     1 F16NUM,F4NUM,F17NUM
4150  FORMAT(2X,'TIME INVENTORY TOO HIGH :',E13.7,
     1        2X,'# OF OBSERVATIONS INV.  :',E13.7,/,
     2        2X,'% INVENTORY DOWNTIME    :',E13.7,
     3        2X,'SUM UNITS PROCESSED     :',E13.7,/,
     4        2X,'TOTAL # OF MACHINES     :',I13,
     5        2X,'SUM COST OF PROCESS     :',E13.7,/,
     6        2X,'SUM SCHEDULED HOURS     :',E13.7,
     7        2X,'COST PER UNIT           :',E13.7)
      WRITE(*,4200)F9NUM,F18NUM,F5NUM,F10NUM,F20NUM
      IF(OUTFLAG.GT.1) WRITE(15,4200)F9NUM,F18NUM,F5NUM,F10NUM,F20NUM
4200  FORMAT(2X,'SUM MACH.BREAKDOWN HOURS:',E13.7,
     1        2X,'COST PER SCHEDULED HOUR :',E13.7,/,
     2        2X,'SUM PRODUCTIVE HOURS    :',E13.7,/,
     3        2X,'% NET UTILIZATION MACH. :',E13.7,/,
     4        2X,'% GROSS UTILIZATION MACH:',E13.7,//)
C
C
      IF(INDEX9.EQ.1.) THEN
        INDEX4=1
        INDEX3=4
      ELSEIF(INDEX9.GT.1.AND.INDEX9.LE.10) THEN
        INDEX4=INDEX9*3-1
        INDEX3=INDEX9*3+1
      ELSEIF(INDEX9.EQ.11) THEN
        INDEX4=0
        INDEX3=0
        GOTO 4455
      ELSEIF(INDEX9.EQ.12) THEN
        INDEX4=37
        INDEX3=40
      ELSEIF(INDEX9.EQ.13) THEN
        INDEX4=41
        INDEX3=42
      ELSE
        CONTINUE
      ENDIF
C
      DO 4450 INDEX2=INDEX4,INDEX3,1
        INDEX1=USERARR(6,INDEX2)
        IF(INDEX1.EQ.0)GOTO 4445
        F4NUM=F3NUM*INDEX1
        F9NUM=GETARY(18,INDEX2)
        F5NUM=GETARY(10,INDEX2)
        F10NUM=(F5NUM/F4NUM)*100
        F20NUM=((F5NUM+F9NUM)/F4NUM)*100
        F16NUM=F5NUM*USERARR(22,INDEX2)+F4NUM*USERARR(21,INDEX2)
        F18NUM=F16NUM/F4NUM
```

```
      F16NUM=F16NUM/INDEX1
      WRITE(*,4300)INDEX2,MCHNAMES(INDEX2),INDEX1,F16NUM,F4NUM,F18NUM,
     1           F9NUM,F10NUM
      IF(OUTFLAG.GT.1)WRITE(15,4300)INDEX2,MCHNAMES(INDEX2),INDEX1,
     1           F16NUM,F4NUM,F18NUM,F9NUM,F10NUM
4300  FORMAT(//,2X,'MACHINE TYPE 'I2' : ',A,/,
     1       2X,'------------------',//,
     2       2X,'TOTAL # OF MACHINES     :',I13,
     3       2X,'COST PER MACHINE        :',E13.7,/,
     4       2X,'SUM SCHEDULED HOURS     :',E13.7,
     5       2X,'COST PER SCHEDULED HOUR :',E13.7,/
     6       2X,'SUM MACH.BREAKDOWN HOURS:',E13.7,
     7       2X,'% NET UTILIZATION MACH. :',E13.7)
      WRITE(*,4350)F5NUM,F20NUM
      IF(OUTFLAG.GT.1) WRITE(15,4350)F5NUM,F20NUM
4350  FORMAT(
     1       2X,'SUM PRODUCTIVE HOURS    :',E13.7,
     2       2X,'% GROSS UTILIZATION MACH:',E13.7)
4445  CONTINUE
4450 CONTINUE
C
C
C
4455 RETURN
C
C
C
4500 INDEX1=USERARR(6,32)+USERARR(6,33)+USERARR(6,34)+USERARR(6,35)
     1      +USERARR(6,36)
     IF(INDEX1.EQ.0)GOTO 4900
     WRITE (*,4550)
     IF (OUTFLAG.GT.1) WRITE(15,4550)
4550 FORMAT(/////,10X,'LOADING DEVICES'/,
     1           10X,'===============',//)
C
     INDEX1=USERARR(6,32)+USERARR(6,33)+USERARR(6,34)+USERARR(6,35)
     1      +USERARR(6,36)
     F4NUM=TNOW*INDEX1
     F5NUM=GETARY(10,32)+GETARY(10,33)+GETARY(10,34)+GETARY(10,35)
     1      +GETARY(10,36)
     F9NUM=GETARY(18,32)+GETARY(18,33)+GETARY(18,34)+GETARY(18,35)
     1      +GETARY(18,36)
     F10NUM=(F5NUM/F4NUM)*100
     F20NUM=((F9NUM+F5NUM)/F4NUM)*100
     F19NUM=XX(4)
     F16NUM=GETARY(10,32)*USERARR(22,32)+GETARY(10,33)*USERARR(22,33)+
     1       GETARY(10,34)*USERARR(22,34)+GETARY(10,35)*USERARR(22,35)+
     2       GETARY(10,36)*USERARR(22,36)+
     3       USERARR(6,32)*F3NUM*USERARR(21,32)+
     4       USERARR(6,33)*F3NUM*USERARR(21,33)+
     5       USERARR(6,34)*F3NUM*USERARR(21,34)+
     6       USERARR(6,35)*F3NUM*USERARR(21,35)+
     7       USERARR(6,36)*F3NUM*USERARR(21,36)
```

```
      F17NUM=F16NUM/F19NUM
      F18NUM=F16NUM/TNOW
C
      WRITE(*,4600)INDEX1,F19NUM,F4NUM,F16NUM,F9NUM,F17NUM,F5NUM,F18NUM
      IF(OUTFLAG.GT.1) WRITE(15,4600)INDEX1,F19NUM,F4NUM,F16NUM,F9NUM,
     1         F17NUM,F5NUM,F18NUM
4600  FORMAT(2X,'TOTAL # OF MACHINES      :',I13,
     1       2X,'SUM OF UNITS HARVESTED  :',E13.7,/,
     2       2X,'SUM SCHEDULED HOURS     :',E13.7,
     3       2X,'SUM COST LOADER DEVICES :',E13.7,/,
     4       2X,'SUM MACH.BREAKDOWN HOURS:',E13.7,
     5       2X,'COST PER UNIT           :',E13.7,/,
     6       2X,'SUM PRODUCTIVE HOURS    :',E13.7,
     7       2X,'COST PER SCHEDULED HOUR :',E13.7)
      WRITE(*,4650)F10NUM,F20NUM
      IF(OUTFLAG.GT.1) WRITE(15,4650)F10NUM,F20NUM
4650  FORMAT(2X,'% NET UTILIZATION MACH. :',E13.7,/,
     1       2X,'% GROSS UTILIZATION MACH:',E13.7,/)
      USERARR(25,1)=USERARR(25,1)+F4NUM
      USERARR(25,2)=USERARR(25,2)+F9NUM
      USERARR(25,3)=USERARR(25,3)+F5NUM
      USERARR(25,4)=USERARR(25,4)+F16NUM
C
      INDEX4=32
      INDEX3=36
      DO 4800 INDEX2=INDEX4,INDEX3,1
        INDEX1=USERARR(6,INDEX2)
        IF (INDEX1.EQ.0)GOTO 4790
        F4NUM=TNOW*INDEX1
        F9NUM=GETARY(18,INDEX2)
        F5NUM=GETARY(10,INDEX2)
        F10NUM=(F5NUM/F4NUM)*100
        F20NUM=((F5NUM+F9NUM)/F4NUM)*100
        F16NUM=F5NUM*USERARR(22,INDEX2)+F4NUM*USERARR(21,INDEX2)
        F18NUM=F16NUM/F4NUM
        F16NUM=F16NUM/INDEX1
        WRITE(*,4700)INDEX2,MCHNAMES(INDEX2),INDEX1,F16NUM,F4NUM,F18NUM,
     1            F9NUM,F10NUM
        IF(OUTFLAG.GT.1)WRITE(15,4700)INDEX2,MCHNAMES(INDEX2),INDEX1,
     1            F16NUM,F4NUM,F18NUM,F9NUM,F10NUM
4700    FORMAT(//,2X,'MACHINE TYPE 'I2' : ',A,/,
     1          2X,'-----------------',//,
     2       2X,'TOTAL # OF MACHINES      :',I13,
     3       2X,'COST PER MACHINE         :',E13.7,/,
     4       2X,'SUM SCHEDULED HOURS      :',E13.7,
     5       2X,'COST PER SCHEDULED HOUR :',E13.7,/
     6       2X,'SUM MACH.BREAKDOWN HOURS:',E13.7,
     7       2X,'% NET UTILIZATION MACH. :',E13.7)
        WRITE(*,4750)F5NUM,F20NUM
        IF(OUTFLAG.GT.1) WRITE(15,4750)F5NUM,F20NUM
4750    FORMAT(2X,'SUM PRODUCTIVE HOURS     :',E13.7,
     1         2X,'% GROSS UTILIZATION MACH:',E13.7)
4790    CONTINUE
```

```
4800  CONTINUE
C
C
C
C
C
4900  WRITE (*,4910)
      IF (OUTFLAG.GT.1) WRITE(15,4910)
4910  FORMAT(/////,10X,'COMPLETE HARVESTING SYSTEM STATISTICS'/,
     1             10X,'-------------------------------------',//)
C
      CALL DATE (10,TSTR)
      EDSTR=TSTR
      CALL TIME (10,TSTR)
      ETSTR=TSTR
      WRITE(*,4920) SDSTR,STSTR,EDSTR,ETSTR
      IF (OUTFLAG.GT.1) WRITE(15,4920) SDSTR,STSTR,EDSTR,ETSTR
4920  FORMAT(2X,'COMPUTER TIME START SIMULATION   DATE: ',A,
     1 3X,'TIME: 'A,/,
     2       2X,'COMPUTER TIME END SIMULATION     DATE: ',A,
     3 3X,'TIME: 'A,/)
       WRITE(*,4921)NNRUN,SIMRUN
       IF(OUTFLAG.GT.1)WRITE(15,4921)NNRUN,SIMRUN
4921    FORMAT('  SIMULATION RUN 'I2' OF 'F3.0)
C
      F1NUM=0
      F2NUM=TNOW
      INDEX1=0
      DO 4930 INDEX8=1,42,1
        INDEX1=INDEX1+USERARR(6,INDEX8)
4930  CONTINUE
      F4NUM=USERARR(25,1)
      F9NUM=USERARR(25,2)
      F5NUM=USERARR(25,3)
      F10NUM=(F5NUM/F4NUM)*100
      F20NUM=((F5NUM+F9NUM)/F4NUM)*100
      F19NUM=XX(4)
      F16NUM=USERARR(25,4)
      F17NUM=F16NUM/F19NUM
      F18NUM=F16NUM/F2NUM
C
      WRITE(*,4935)F1NUM,F2NUM
      IF (OUTFLAG.GT.1) WRITE(15,4935)F1NUM,F2NUM
4935  FORMAT(2X,'BEGIN OF HARVESTING     :',E13.7,
     1       2X,'END OF HARVESTING       :',E13.7)
      WRITE(*,4940)INDEX1,F19NUM,F4NUM,F16NUM,F9NUM,F17NUM,F5NUM,F18NUM
      IF(OUTFLAG.GT.1) WRITE(15,4940)INDEX1,F19NUM,F4NUM,F16NUM,F9NUM,
     1          F17NUM,F5NUM,F18NUM
4940  FORMAT(2X,'TOTAL # OF MACHINES     :',I13,
     1       2X,'SUM OF UNITS HARVESTED  :',E13.7,/,
     2       2X,'SUM SCHEDULED HOURS     :',E13.7,
     3       2X,'SUM COST OF SYSTEM      :',E13.7,/,
     4       2X,'SUM MACH.BREAKDOWN HOURS:',E13.7,
```

```
     5        2X,'COST PER UNIT          :',E13.7,/,
     6        2X,'SUM PRODUCTIVE HOURS   :',E13.7,
     7        2X,'COST PER SYSTEM HOUR   :',E13.7)
C
C
C
      WRITE(*,4950)F10NUM,F20NUM,NNRUN,SIMRUN
      IF(OUTFLAG.GT.1) WRITE(15,4950)F10NUM,F20NUM,NNRUN,SIMRUN
4950  FORMAT(2X,'% NET UTILIZATION MACH. :',E13.7,/,
     1        2X,'% GROSS UTILIZATION MACH:',E13.7,/////,
     2        2X,'-------------------------- END OF RUN #'I2' OF 'F3.0,
     3          '--------------------------',///)
C
      IF(OUTFLAG.GT.1)CLOSE(15)
C
C
C
CC
C
C
C
C
      IF (NNRUN.EQ.SIMRUN) THEN
        STOP ' '
      ELSE
        CONTINUE
      ENDIF
C
C
C
C
      RETURN
C
C
C
C  USERFUNCTION TO RECORD OBSERVATIONS ON THE INVENTORIES
C  ====================================================
C
C
5000  USERF=0
      IF (ATRIB(5).EQ.1) RETURN
C
      INDEX1=ATRIB(5)
      INDEX2=XXLEVEL(2)+ATRIB(5)
      CALL COLCT(XX(INDEX2),INDEX1)
      RETURN
C
5100  USERF=0
      INDEX3=ATRIB(5)
      INDEX1=GETARY(23,INDEX3)
      INDEX2=XXLEVEL(2)+INDEX1
      CALL COLCT(XX(INDEX2),INDEX1)
      RETURN
```

```
C
5200  USERF=0
      INDEX1=USERARR(7,5)
      INDEX2=XXLEVEL(2)+INDEX1
      CALL COLCT(XX(INDEX2),INDEX1)
      RETURN
C
5300  USERF=0
      INDEX1=USERARR(7,6)
      INDEX2=XXLEVEL(2)+INDEX1
      CALL COLCT(XX(INDEX2),INDEX1)
      RETURN
C
C
C
C
C  USERFUNCTION 160: CALUCALTE INVENTORY TO MOVE ROUTE 1 & ROUTE 2 DISTRIBUTION
C  ===========================================================================
C
C
7000  USERF=0
C
C
C   CALCULATE THE CURRENT INVENTORY
C   -------------------------------
C
      INDEX1=XXLEVEL(2)+11
      F1NUM=XX(INDEX1)-XX(7)
      XX(7)=XX(INDEX1)
      IF (F1NUM.GT.0) THEN
        F2NUM=F1NUM*USERARR(7,3)/100
        F3NUM=GETARY(7,7)+F2NUM
        CALL PUTARY(7,7,F3NUM)
        F4NUM=F1NUM*USERARR(7,4)/100
        F5NUM=GETARY(7,8)+F4NUM
        CALL PUTARY(7,8,F5NUM)
      ELSE
        CONTINUE
      ENDIF
C
C
C   CALCULATING THE INVENTORY TO MOVE ROUTE 1
C   -----------------------------------------
C
      INDEX1=USERARR(7,5)
      INDEX2=USERARR(7,5)+XXLEVEL(2)
      F1NUM=XX(INDEX2)
      INDEX3=USERARR(7,5)+XXLEVEL(4)
      F2NUM=XX(INDEX3)
      F3NUM=F2NUM-F1NUM
      F4NUM=GETARY(7,7)
```

```
C
      IF(F1NUM.GT.F2NUM) THEN
        USERARR(7,15)=0
      ELSEIF (F3NUM.GT.F4NUM) THEN
        USERARR(7,15)=F4NUM
      ELSE
        USERARR(7,15)=F3NUM
      ENDIF
C
C
C   CALCULATING THE INVENTORY TO MOVE ROUTE 2
C   -------------------------------------------
C
      INDEX1=USERARR(7,6)
      INDEX2=USERARR(7,6)+XXLEVEL(2)
      F1NUM=XX(INDEX2)
      INDEX3=USERARR(7,6)+XXLEVEL(4)
      F2NUM=XX(INDEX3)
      F3NUM=F2NUM-F1NUM
      F4NUM=GETARY(7,8)
C
      IF(F1NUM.GT.F2NUM) THEN
        USERARR(7,16)=0
      ELSEIF (F3NUM.GT.F4NUM) THEN
        USERARR(7,16)=F4NUM
      ELSE
        USERARR(7,16)=F3NUM
      ENDIF
C
C
C CALCULATE SUM TO MOVE
C ---------------------
C
C  TEST IF PREVIOUS PROCESS ENDED
C
      INDEX2=USERARR(24,11)
      INDEX3=GETARY(15,INDEX2)
      F1NUM=GETARY(26,INDEX2)
      IF (INDEX3.LT.2) THEN
        INDEX4=0
      ELSEIF (INDEX3.GE.2.AND.F1NUM.GT.0.001) THEN
        INDEX4=0
      ELSE
        INDEX4=1
      ENDIF
C
      F1NUM=USERARR(7,15)
      F2NUM=USERARR(7,16)
      F3NUM=F1NUM+F2NUM
      INDEX1=USERARR(4,11)
      IF(INDEX1.GT.0)F4NUM=USERARR(8,INDEX1)
```

```
C
      IF(F3NUM.EQ.Ø) THEN
        ATRIB(2)=Ø
        ATRIB(6)=Ø
        ATRIB(7)=Ø
        CALL PUTARY(7,1Ø,Ø)
        CALL PUTARY(7,15,Ø)
        CALL PUTARY(7,16,Ø)
      ELSEIF(INDEX1.GT.Ø.AND.F4NUM.GT.F3NUM.AND.INDEX4.EQ.Ø)THEN
        ATRIB(2)=Ø
        ATRIB(6)=Ø
        ATRIB(7)=Ø
        CALL PUTARY(7,1Ø,Ø)
        CALL PUTARY(7,15,Ø)
        CALL PUTARY(7,16,Ø)
      ELSE
        ATRIB(2)=F3NUM
        ATRIB(6)=F1NUM
        ATRIB(7)=F2NUM
        CALL PUTARY(7,1Ø,F3NUM)
        CALL PUTARY(7,15,F1NUM)
        CALL PUTARY(7,16,F2NUM)
        F15NUM=GETARY(7,7)
        F16NUM=GETARY(7,8)
        F15NUM=F15NUM-F1NUM
        F16NUM=F16NUM-F2NUM
        CALL PUTARY(7,7,F15NUM)
        CALL PUTARY(7,8,F16NUM)
        INDEX1=XXLEVEL(2)+11
        XX(INDEX1)=XX(INDEX1)-F3NUM
        XX(7)=XX(INDEX1)
      ENDIF
C
C
C
      INDEX1=USERARR(4,11)
      IF (INDEX1.EQ.Ø) THEN
        RETURN
      ELSE
        CONTINUE
      ENDIF
C
C
C  SET AMOUNT OF TREES ACCORDING TO DESIRED DISTRIBUTION
C  ---------------------------------------------------------
C
C
      F4NUM=F3NUM
      F2ØNUM=F3NUM
```

```
C
C     INDEXING THE DISARR-COLUMS ACCORDING TO DISTRIBUTION SPECIFIED:
7025  INDEX1=GETARY(5,11)
      IF (INDEX1.EQ.1) THEN
        INDEX2=1
        INDEX3=2
      ELSEIF (INDEX1.EQ.2) THEN
        INDEX2=3
        INDEX3=4
      ELSEIF (INDEX1.EQ.3) THEN
        INDEX2=5
        INDEX3=6
      ELSEIF (INDEX1.EQ.4) THEN
        INDEX2=7
        INDEX3=8
      ELSE
        CONTINUE
      ENDIF
C
C     SET INITIAL VALUES:
      ATRIB(3)=0
      ATRIB(2)=0
      F5NUM=0
      F6NUM=0
      INDEX4=0
C
C     LOOP TO CREATE BATCHSIZE:
C
C     F6NUM=CURRENT BATCH SIZE CALCULATED
C     F5NUM=TESTING VALUE SUM CUFT
C     F4NUM=BATCHSIZE
C     F3NUM=SAMPLE
C     F2NUM=UPPERBOUND
C     F1NUM=LOWERBOUND
C     INDEX4=NUMBER OF TREES
C
7050  F3NUM=UNFRM(0.0,100.0,6)
      F1NUM=0
C
      DO 7100 INDEX8=1,10,1
        F2NUM=DISARR(INDEX2,INDEX8)
       IF (F3NUM.GE.F1NUM.AND.F3NUM.LE.F2NUM) THEN
          F5NUM=F5NUM+DISARR(INDEX3,INDEX8)
          INDEX4=INDEX4+1
          GOTO 7200
        ELSE
          F1NUM=F2NUM
        ENDIF
7100  CONTINUE
C
7200  IF (F5NUM.LE.F4NUM) THEN
        F6NUM=F5NUM
        GOTO 7050
```

```
      ELSE
        ATRIB(2)=F20NUM
        ATRIB(3)=INDEX4-1
      ENDIF
C
C
      RETURN
C
C
C
C
C
C  USER FUNCTION TO DISPLAY THE AMOUNT HARVESTED
C  *******************************************
C
C
C
7500  USERF=0
      CALL TIME (10,TSTR)
      WRITE(*,7510)NNRUN,XX(15),TSTR,TNOW
7510  FORMAT(2X,'  RUN #'I2,'  AMOUNT HARVESTED:'E13.7' TIME:'A,
     1           ' TNOW:'E13.7)
C
C
      RETURN
C
C
C
C
C
9998  RETURN
C
C
      END
```

APPENDIX C

TABLE OF CONTENTS:

APPENDIX C

1. Example session, LOGSIM

```
                    S L A M   I I

                  V e r s i o n   3.0

    This software is proprietary to and a trade secret of Pritsker
    & Associates, INC.  Access to and use of the software is granted
    under the terms and conditions of the software license agreement
    between Pritsker & Associates, INC., and licensee.

    The terms and conditions of the agreement shall be strictly
    enforced.  Any violations of the agreement may void licensees
    right to use the software.


        P r i t s k e r  &  A s s o c i a t e s ,  I N C .

              P . O .  B o x    2 4 1 3

        W e s t  L a f a y e t t e ,  I N  4 7 9 0 6



    Enter file name of translated model:HARVEST.TRA



    **********************************************************
    *                                                        *
    *                  >>> L O G S I M <<<                   *
    *          MECHANIZED LOG HARVESTING SIMULATOR           *
    *                   SIMULATION MODULE                    *
    **********************************************************



              BEGIN OF SIMULATION
              ===================

THIS FUNCTION READS A PREVIOSLY DEFINED MODEL INTO THE SLAM-
NETWORK AND SIMULATES IT.


  DO YOU WISH TO CONTINUE (Y/N) ?  [Y]-----> Y
```

```
FILENAME OF MODEL TO BE RETRIEVED?  -----> PTEST10.MOD

   !!!           PLEASE WAIT A MOMENT          !!!


   !!!   MODEL HAS BEEN SUCCESSFULLY RETRIEVED  !!!



   SIMULATION RESULTS SHOULD BE ROUTED TO:
      SCREEN                = 1
      SCREEN & PRINTER      = 2

   PLEASE ENTER CHOICE  -----> 1


   HOW MANY SIMULATION RUNS DO YOU WANT?
   THE PRESET MAXIMUM IS 10.
   ENTER NUMBER OF RUNS        [1]-----> 1


   PLEASE HIT  >RETURN<  TO START THE SIMULATION
   ---------------------------------------------
TIME=22:28:44
DATE=06-01-87
```

INTERMEDIATE RESULTS

```
RUN # 1  AMOUNT HARVESTED: .1000000E+01 TIME:22:28:44   TNOW: .2000000E+01
RUN # 1  AMOUNT HARVESTED: .2000000E+01 TIME:22:28:44   TNOW: .3000000E+01
RUN # 1  AMOUNT HARVESTED: .3000000E+01 TIME:22:28:45   TNOW: .4000000E+01
RUN # 1  AMOUNT HARVESTED: .4000000E+01 TIME:22:28:45   TNOW: .5000000E+01
RUN # 1  AMOUNT HARVESTED: .5000000E+01 TIME:22:28:45   TNOW: .6000000E+01
RUN # 1  AMOUNT HARVESTED: .6000000E+01 TIME:22:28:45   TNOW: .7000000E+01
RUN # 1  AMOUNT HARVESTED: .7000000E+01 TIME:22:28:45   TNOW: .8000000E+01
RUN # 1  AMOUNT HARVESTED: .8000000E+01 TIME:22:28:45   TNOW: .9000000E+01
RUN # 1  AMOUNT HARVESTED: .9000000E+01 TIME:22:28:46   TNOW: .1000000E+02
RUN # 1  AMOUNT HARVESTED: .1000000E+02 TIME:22:28:46   TNOW: .1100000E+02
RUN # 1  AMOUNT HARVESTED: .1100000E+02 TIME:22:28:46   TNOW: .1200000E+02
RUN # 1  AMOUNT HARVESTED: .1200000E+02 TIME:22:28:46   TNOW: .1300000E+02
RUN # 1  AMOUNT HARVESTED: .1300000E+02 TIME:22:28:46   TNOW: .1400000E+02
RUN # 1  AMOUNT HARVESTED: .1400000E+02 TIME:22:28:46   TNOW: .1500000E+02
```

```
*******************************************************
*                                                     *
*                 >>> L O G S I M <<<                 *
*                 SIMULATION  RESULTS                 *
*                                                     *
*******************************************************
```

SIMULATION MODEL USED: PTEST10.MOD
****

DATE= 06-01-87
TIME= 22:28:52
SIMULATION RUN 1 OF 1.

PROCESS NO. 1 : felling

```
TIME BEGIN OF PROCESS    : .0000000E+00  AVERAGE INVENTORY        : .0000000E+00
TIME END OF PROCESS      : .5000000E+02  MAXIMUM INVENTORY        : .0000000E+00
DURATION OF PROCESS      : .5000000E+02  MINIMUM INVENTORY        : .0000000E+00
TIME INVENTORY TOO LOW   : .0000000E+00  STD.DEV.INVENTORY        : .0000000E+00
TIME INVENTORY TOO HIGH  : .0000000E+00  # OF OBSERVATIONS INV.   : .0000000E+00
% INVENTORY DOWNTIME     : .0000000E+00  SUM UNITS PROCESSED      : .5000000E+02
TOTAL # OF MACHINES      :            1  SUM COST OF PROCESS      : .0000000E+00
SUM SCHEDULED HOURS      : .5000000E+02  COST PER UNIT            : .0000000E+00
SUM MACH.BREAKDOWN HOURS : .0000000E+00  COST PER SCHEDULED HOUR  : .0000000E+00
```

```
SUM PRODUCTIVE HOURS    : .5000000E+02
% NET UTILIZATION MACH. : .1000000E+03
% GROSS UTILIZATION MACH: .1000000E+03



MACHINE TYPE  1 :
-----------------

TOTAL # OF MACHINES     :            1  COST PER MACHINE         : .0000000E+00
SUM SCHEDULED HOURS     : .5000000E+02  COST PER SCHEDULED HOUR  : .0000000E+00
SUM MACH.BREAKDOWN HOURS: .0000000E+00  % NET UTILIZATION MACH.  : .1000000E+03
SUM PRODUCTIVE HOURS    : .5000000E+02  % GROSS UTILIZATION MACH: .1000000E+03

  RUN # 1  AMOUNT HARVESTED: .4900000E+02 TIME:22:28:52   TNOW: .5000000E+02
  RUN # 1  AMOUNT HARVESTED: .5000000E+02 TIME:22:28:52   TNOW: .5100000E+02



        PROCESS NO.13 : ftrapo
        ===============


TIME BEGIN OF PROCESS   : .1000000E+01  AVERAGE INVENTORY        : .5000000E+00
TIME END OF PROCESS     : .5100000E+02  MAXIMUM INVENTORY        : .1000000E+01
DURATION OF PROCESS     : .5000000E+02  MINIMUM INVENTORY        : .0000000E+00
TIME INVENTORY TOO LOW  : .0000000E+00  STD.DEV.INVENTORY        : .5025189E+00
TIME INVENTORY TOO HIGH : .0000000E+00  # OF OBSERVATIONS INV.   : .1000000E+03
% INVENTORY DOWNTIME    : .0000000E+00  SUM UNITS PROCESSED      : .5000000E+02
TOTAL # OF MACHINES     :            1  SUM COST OF PROCESS      : .0000000E+00
SUM SCHEDULED HOURS     : .5000000E+02  COST PER UNIT            : .0000000E+00
SUM MACH.BREAKDOWN HOURS: .0000000E+00  COST PER SCHEDULED HOUR  : .0000000E+00
SUM PRODUCTIVE HOURS    : .5000000E+02
% NET UTILIZATION MACH. : .1000000E+03
% GROSS UTILIZATION MACH: .1000000E+03



MACHINE TYPE 41 : truck
-----------------

TOTAL # OF MACHINES     :            1  COST PER MACHINE         : .0000000E+00
SUM SCHEDULED HOURS     : .5000000E+02  COST PER SCHEDULED HOUR  : .0000000E+00
SUM MACH.BREAKDOWN HOURS: .0000000E+00  % NET UTILIZATION MACH.  : .1000000E+03
SUM PRODUCTIVE HOURS    : .5000000E+02  % GROSS UTILIZATION MACH: .1000000E+03
```

COMPLETE HARVESTING SYSTEM STATISTICS

```
COMPUTER TIME START SIMULATION   DATE: 06-01-87    TIME: 22:28:44
COMPUTER TIME END SIMULATION     DATE: 06-01-87    TIME: 22:28:52

SIMULATION RUN  1 OF  1.
BEGIN OF HARVESTING      : .0000000E+00  END OF HARVESTING        : .5100000E+02
TOTAL # OF MACHINES      :            2  SUM OF UNITS HARVESTED   : .5000000E+02
SUM SCHEDULED HOURS      : .1000000E+03  SUM COST OF SYSTEM       : .0000000E+00
SUM MACH.BREAKDOWN HOURS: .0000000E+00   COST PER UNIT            : .0000000E+00
SUM PRODUCTIVE HOURS     : .1000000E+03  COST PER SYSTEM HOUR     : .0000000E+00
% NET UTILIZATION MACH. : .1000000E+03
% GROSS UTILIZATION MACH: .1000000E+03


-------------------------- END OF RUN # 1 OF  1.--------------------------
```

APPENDIX C

2. Example session, FRONTEND.FOR, Readin

```
****************************************************
*                                                  *
*              >>> L O G S I M <<<                 *
*             INPUT USER-INTERFACE                 *
*                                                  *
****************************************************
```

```
              M A I N - M E N U
     --------------------------------------------

          DEFINING A MODEL              = 1
          PRINT OUT A MODEL             = 2
          EDIT A MODEL                  = 3
          EXIT THE PROGRAM              = 0


          ENTER CHOICE PLEASE       -------> 1
```

```
              SUBROUTINE READIN
              ================
```

THIS IS THE SUBROUTINE TO DEFINE THE SIMULATION MODEL, LATER USED BY THE SLAM PROCESSOR. THE FOLLOWING DATA IS NECCESSARY TO DEFINE A HARVESTING MODEL:

- HARVESTING CONFIGURATION, GENERAL PARAMETERS
- DISTRIBUTION DATA OF THE TREES/LOGS/PULPLOGS/SAWLOGS
- MAXIMUM/MINIMUM DATA OF MATERIAL BUFFERS PER PROCESS
- HOW MANY MACHINES USED PER PROCESS
- MACHINE DATA: TIMES, BREAKDOWN FREQUENCIES, COSTS ETC

DEFAULT VALUES WILL BE GIVEN IN SQUARE BRAKETS [ ]. IF YOU WANT TO USE THEM, HIT SPACE BAR AND PRESS RETURN.

DO YOU WISH TO CONTINUE (Y/N) ?   [Y]-----> Y

```
                 FIRST PHASE
                 ============

 SPECIFICATION OF GENERAL SYSTEM PARAMETERS


WHAT IS THE FILENAME OF THE MODEL?  --->TEST.MOD

HOW MANY CU.FT SHOULD BE HARVESTED? --->2ØØØØ.

VALUE OF THE TIME DELAY PARAMETER?  --->.Ø1



                 SECOND PHASE
                 =============

WE NOW ARE GOING TO DEFINE THE MATERIAL FLOW THROUGH
THE HARVESTING SYSTEM. FOR EACH PROCESS PLEASE STATE
THE PROCESS FROM WHICH THE INCOMING MATERIALS STREAM
ORIGINATES AND THE DESTINATION OF THE OUTGOING MATERIAL
STREAM. A VALUE OF Ø FOR BOTH QUESTIONS MEANS THAT THE
PROCESS IS NOT USED.

            (PLEASE HIT RETURN TO CONTINUE)



PROCESS 1:
----------
OUTGOING DESTINATION?  --->2

PROCESS  2:
----------

INCOMING ORIGIN?       --->1
OUTGOING DESINATION?   --->13

PROCESS  3:
----------
```

```
INCOMING ORIGIN?       --->
OUTGOING DESINATION?   --->

PROCESS  4:
----------

INCOMING ORIGIN?       --->
OUTGOING DESINATION?   --->

PROCESS  5:
----------

INCOMING ORIGIN?       --->
OUTGOING DESINATION?   --->

PROCESS  6:
----------

INCOMING ORIGIN?       --->
OUTGOING DESINATION?   --->

PROCESS  7:
----------

INCOMING ORIGIN?       --->
OUTGOING DESINATION?   --->

PROCESS  8:
----------

INCOMING ORIGIN?       --->
OUTGOING DESINATION?   --->

PROCESS  9:
----------

INCOMING ORIGIN?       --->
OUTGOING DESINATION?   --->

PROCESS 10:
----------

INCOMING ORIGIN?       --->
OUTGOING DESINATION?   --->

PROCESS 11:
----------

INCOMING ORIGIN?       --->
OUTGOING DESINATION, ROUTE 1 ?   --->
OUTGOING DESINATION, ROUTE 2 ?   --->
```

```
PROCESS 12:
----------

INCOMING ORIGIN?        --->

PROCESS 13:
----------

INCOMING ORIGIN?        --->2




PROCESS #      IN ORIGIN    OUT DESTINATION
--------------------------------------------
    1                         2.
    2             1.         13.
    3
    4
    5
    6
    7
    8
    9
   10
   11
   12
   13             2.

IS THIS CORRECT? (Y/N)      [Y]----->



                  THIRD PHASE
                  ===========

NOW WE DEFINE THE CUMULATIVE FREQUENCY DISTRIBUTIONS
USED TO DESCRIBE TREES, LOGS ETC..
YOU CAN SPECIFY UP TO FOUR DIFFERENT FREQUENCY DISTRIBUTIONS,
WITH 10 FREQUENCY CLASSES EACH. YOU HAVE TO SPECIFY AT LEAST
ONE CLASS IN ONE  DISTRIBUTION.
!!! DONT FORGET THE DECIMAL POINT FOR INPUT !!!


          (PLEASE HIT RETURN TO CONTINUE)
```

```
FREQUENCY DISTRIBUTION NO. 1:
----------------------------

 NAME OF THIS DISTRIBUTION ?   -----> WHOLE TREES

CLASS  1: CUM.REL.FREQENCY?  [0]--->11.6
CLASS  1: VOLUME CU.FT?      [0]--->4.4

CLASS  2: CUM.REL.FREQENCY?  [0]--->29.7
CLASS  2: VOLUME CU.FT?      [0]--->9.4

CLASS  3: CUM.REL.FREQENCY?  [0]--->50.4
CLASS  3: VOLUME CU.FT?      [0]--->18.

CLASS  4: CUM.REL.FREQENCY?  [0]--->69.9
CLASS  4: VOLUME CU.FT?      [0]--->28.3

CLASS  5: CUM.REL.FREQENCY?  [0]--->84.4
CLASS  5: VOLUME CU.FT?      [0]--->40.9

CLASS  6: CUM.REL.FREQENCY?  [0]--->93.
CLASS  6: VOLUME CU.FT?      [0]--->54.6

CLASS  7: CUM.REL.FREQENCY?  [0]--->97.7
CLASS  7: VOLUME CU.FT?      [0]--->70.2

CLASS  8: CUM.REL.FREQENCY?  [0]--->100.
CLASS  8: VOLUME CU.FT?      [0]--->92.1
```

DISTRIBUTION NO. 1 : WHOLE TREES

CLASS	CUM.REL.FREQ.%	CU.FT
1	11.60	4.40
2	29.70	9.40
3	50.40	18.00
4	69.90	28.30
5	84.40	40.90
6	93.00	54.60
7	97.70	70.20
8	100.00	92.10

```
DISTRIBUTION OK (Y/N) ?  [Y]---->Y




FREQUENCY DISTRIBUTION NO. 2:
----------------------------

 NAME OF THIS DISTRIBUTION ?   ----->

CLASS  1: CUM.REL.FREQENCY?  [Ø]--->
CLASS  1: VOLUME CU.FT?      [Ø]--->


        DISTRIBUTION NO. 2 :
        ====================

  CLASS      CUM.REL.FREQ.%         CU.FT
 -----------------------------------------

  *******  DISTRIBUTION NOT USED  *******


 DISTRIBUTION OK (Y/N) ?  [Y]---->Y




FREQUENCY DISTRIBUTION NO. 3:
----------------------------

 NAME OF THIS DISTRIBUTION ?   ----->

CLASS  1: CUM.REL.FREQENCY?  [Ø]--->
CLASS  1: VOLUME CU.FT?      [Ø]--->


        DISTRIBUTION NO. 3 :
        ====================

  CLASS      CUM.REL.FREQ.%         CU.FT
 -----------------------------------------

  *******  DISTRIBUTION NOT USED  *******
```

```
 DISTRIBUTION OK (Y/N) ?  [Y]---->Y



FREQUENCY DISTRIBUTION NO. 4:
----------------------------

 NAME OF THIS DISTRIBUTION ?   ----->

CLASS   1: CUM.REL.FREQENCY?  [0]--->
CLASS   1: VOLUME CU.FT?      [0]--->



        DISTRIBUTION NO. 4 :
        ====================

  CLASS      CUM.REL.FREQ.%        CU.FT
 ------------------------------------------

  *******  DISTRIBUTION NOT USED  *******


 DISTRIBUTION OK (Y/N) ?  [Y]---->Y
```

```
                    FORTH PHASE
                    ===========

IN THIS PHASE WE WILL DESCRIBE THE PROCESSES USED A LITTLE
BIT MORE IN DETAIL. YOU WILL BE ASKED FOR :
   - AN OPTIONAL NAME FOR THE PROCESS
   - THE DISTRIBUTION TO BE USED FOR THIS PROCESS
   - STARTUP-INVENTORY LEVEL FOR THE PROCESS
   - MINIMUM INPUT BUFFER SIZE
   - STARTUP-INV. LEVEL AFTER MINIMUM HAS BEEN REACHED
   - MAXIMUM INPUT BUFFER SIZE
   - STARTUP-INV. LEVEL AFTER MAXIMUM HAS BEEN REACHED

PLEASE REMEMBER: THE INPUT BUFFER OF A PROCESS IS THE OUT-
PUT BUFFER OF HIS PREVIOUS PROCESS. THE MINIMUM BUFFER SIZE
 EFFECTS THE CURRENT PROCESS, THE MAXIMUM EFFECTS THE PREVIOUS
ONE.
```

```
    !!! DONT FORGET THE DECIMAL POINT FOR INPUT !!!


            (PLEASE HIT RETURN TO CONTINUE)


PROCESSS NO. 1
-------------
NAME OF PROCESS?    -----> FELLING
NO. OF DISTRIBUTION TO USE?            -----> 1
WHAT LOADER DO YOU WANT TO USE (32-36) ?      [Ø]----->
TIME DELAYS HANDELD BY
BUILD-IN MODEL=Ø  OR  USERFUNCTION=1 ?        [Ø]----->



 PROCESS NO. 1: FELLING
-------------

 DISTRIBUTION USED:  1
 STARTUP INVENTORY:       .Ø
 MINIMUM INVENTORY:       .Ø
 STARTUP MINIMUM  :       .Ø
 MAXIMUM INVENTORY:       .Ø
 STARTUP MAXIMUM  :       .Ø
 LOADER TYPE USED : NONE
 TIME DELAYS BY   : BUILD-IN FUNCTIONS


     INPUT DATA OK (Y/N)?    [Y]---->Y




PROCESS NO. 2
-------------
NAME OF PROCESS?    -----> SKIDDING
NO. OF DISTRIBUTION TO USE?            -----> 1
STARTUP-INVENTORY LEVEL?             [1]----->
MINNIMUM INFEED INVENTORY LEVEL?     [Ø]----->
STARTUP-INV.LEVEL AFTER MINIMUM?     [Ø]----->
MAXIMUM INFEED INV. LEVEL?    [999999.9]----->
STARTUP-INV.LEVEL AFTER MAXIMUM? [999999.9]-->
WHAT LOADER DO YOU WANT TO USE (32-36) ?      [Ø]----->
TIME DELAYS HANDELD BY
BUILD-IN MODEL=Ø  OR  USERFUNCTION=1 ?        [Ø]----->
```

```
PROCESS NO. 2: SKIDDING
-------------

DISTRIBUTION USED:  1
STARTUP INVENTORY:   1000.0
MINIMUM INVENTORY:       .0
STARTUP MINIMUM  :       .0
MAXIMUM INVENTORY: 999999.9
STARTUP MAXIMUM  : 999999.9
LOADER TYPE USED : NONE
TIME DELAYS BY   : BUILD-IN FUNCTIONS


     INPUT DATA OK (Y/N)?    [Y]---->Y




PROCESS NO.13
-------------
NAME OF PROCESS?    -----> FINAL TRANSPORT
NO. OF DISTRIBUTION TO USE?                -----> 1
STARTUP-INVENTORY LEVEL?             [1]-----> 2000.
MINNIMUM INFEED INVENTORY LEVEL?     [0]-----> 1320.
STARTUP-INV.LEVEL AFTER MINIMUM?     [0]-----> 1320.
MAXIMUM INFEED INV. LEVEL?    [999999.9]-----> 5000.
STARTUP-INV.LEVEL AFTER MAXIMUM? [999999.9]--> 5000.
WHAT LOADER DO YOU WANT TO USE (32-36) ?      [0]----->
TIME DELAYS HANDELD BY
BUILD-IN MODEL=0  OR  USERFUNCTION=1 ?        [0]----->



PROCESS NO.13: FINAL TRANSPORT
-------------

DISTRIBUTION USED:  1
STARTUP INVENTORY:   2000.0
MINIMUM INVENTORY:   1320.0
STARTUP MINIMUM  :   1320.0
MAXIMUM INVENTORY:   5000.0
STARTUP MAXIMUM  :   5000.0
LOADER TYPE USED : NONE
TIME DELAYS BY   : BUILD-IN FUNCTIONS
```

```
    INPUT DATA OK (Y/N)?    [Y]---->Y




                    FIFTH PHASE
                    ===========

WE NOW SPECIFY THE RESOURCES E.G. MACHINES WE WANT TO USE IN
EACH PROCESS. FOR EACH ACTIVE PROCESS THE PROGRAM WILL GIVE
A CHOICE OF DIFFERENT MACHINE TYPES. YOU WILL HAVE TO SPECIFY
THE INITIAL NUMBER OF MACHINES FOR EACH TYPE. MULTIPLE TYPES
OF MACHINES WITH DIFFERENT INITIAL NUMBERS OF MACHINES PER
PROCESS ARE POSSIBLE.
HOWEVER, IF YOU HAVE SPECIFIED ANY PROCESSES USING LOADERS
THE PROGRAM WILL PROMPT YOU FIRST TO ENTER HOW MANY
MACHINES FOR EACH LOADER TYPE USED YOU WANT TO EMPLOY.

THE MAXIMUM NUMBER OF MACHINES WHICH THE NETWORK WILL
HANDEL IS APPROXIMATLY 90 MACHINES IN TOTAL.


        (PLEASE HIT RETURN TO CONTINUE)




  PROCESS NO. 1:  FELLING
  ---------------
THERE ARE FOUR (4) DIFFERENT MACHINE TYPES POSSIBLE:

MACHINE TYPE  1: INITIAL # OF MACHINES ?    [0]-----> 2
MACHINE TYPE  2: INITIAL # OF MACHINES ?    [0]----->
MACHINE TYPE  3: INITIAL # OF MACHINES ?    [0]----->
MACHINE TYPE  4: INITIAL # OF MACHINES ?    [0]----->


 PROCESS NO.  1 : FELLING
 ----------------

 MACHINE TYPE  1, # OF INITIAL MACHINES :  2.
 MACHINE TYPE  2, # OF INITIAL MACHINES :  0.
```

```
MACHINE TYPE  3, # OF INITIAL MACHINES :  Ø.
MACHINE TYPE  4, # OF INITIAL MACHINES :  Ø.


     INPUT DATA OK (Y/N)?    [Y]---->Y




 PROCESS NO. 2:  SKIDDING
 ---------------
THERE ARE THREE (3) DIFFERENT MACHINE TYPES POSSIBLE:

MACHINE TYPE  5: INITIAL # OF MACHINES ?    [Ø]-----> 1
MACHINE TYPE  6: INITIAL # OF MACHINES ?    [Ø]----->
MACHINE TYPE  7: INITIAL # OF MACHINES ?    [Ø]----->


 PROCESS NO.  2 : SKIDDING
 ----------------

 MACHINE TYPE  5, # OF INITIAL MACHINES :  1.
 MACHINE TYPE  6, # OF INITIAL MACHINES :  Ø.
 MACHINE TYPE  7, # OF INITIAL MACHINES :  Ø.


     INPUT DATA OK (Y/N)?    [Y]---->Y




  PROCESS NO.13:  FINAL TRANSPORT
  --------------
FOR THIS PROCESS YOU CAN SPECIFY A PRIMARY AND A SECONDARY
TRANSPORTING DEVICE.

HOW MANY PRIMARY TRANSPORTERS DO YOU WANT TO USE ?        -----> 1
HOW MANY SECONDARY TRANSPORTERS DO YOU WANT TO USE ?  [Ø]-----> Ø


 PROCESS NO. 13 : FINAL TRANSPORT
 ----------------

 NUMBER OF PRIMARY TRANSP.DEVICES USED    :  1.
 NUMBER OF SECONDARY TRANSP.DEVICES USED  :  Ø.
```

```
     INPUT DATA OK (Y/N)?    [Y]---->Y




                 SIXTH PHASE
                 ===========

HERE WE SPECIFY ALL THE PARAMETERS RELATED TO THE MACHINE
TYPES YOU HAVE SET ACTIVE EARLIER:
- NAME OF MACHINE
- AVERAGE PROCESSING TIME PER TREE
- FIXED CONSTANT TIME PER LOAD
- FIXED CONSTANT TIME PER ONE WAY HAUL
- MACHINE CAPACITY IN CU.FT.
- FIXED COST PER SCHEDULED HOUR
- VARIABLE COST PER MACHINE HOUR


      (PLEASE HIT RETURN TO CONTINUE)



PROCESS NO. 1:  FELLING
==============


MACHINE TYPE  1 :
---------------
NAME OF MACHINE TYPE ?              ------> CAT 227 FELLER-BUNCH
AVERAGE PROCESSING TIME / TREE?  [Ø]------>
FIXED CONSTANT TIME / LOAD?      [Ø]------> .Ø4
FIXED CONST. TIME / ONE WAY HAUL? [Ø]------>
MACHINE CAPACITY IN CU.FT?       [1]------> 82.56
FIXED COST / SCHEDULED HOUR      [Ø]------> 41.99
VARIABLE COST/ MACHINE HOUR      [Ø]------> 41.35


PROCESS NO.  1 : FELLING
================
MACHINE TYPE  1
---------------

NAME OF MACHINE TYPE             : CAT 227 FELLER-BUNCH
AVERAGE PROCESSING TIME / TREE   :      .ØØØØ
FIXED CONSTANT TIME / LOAD       :      .Ø4ØØ
FIXED CONST. TIME / ONE WAY HAUL :      .ØØØØ
MACHINE CAPACITY IN CU.FT        :    82.56
FIXED COST / SCHEDULED HOUR      :    41.99
VARIABLE COST/ MACHINE HOUR      :    41.35
```

```
     INPUT DATA OK (Y/N)?    [Y]---->Y



PROCESS NO. 2:  SKIDDING
==============


MACHINE TYPE  5 :
---------------
NAME OF MACHINE TYPE ?              ------> CAT 528 GRAB-SKIDDER
AVERAGE PROCESSING TIME / TREE?  [Ø]------>
FIXED CONSTANT TIME / LOAD?      [Ø]------> .1
FIXED CONST. TIME / ONE WAY HAUL? [Ø]------>
MACHINE CAPACITY IN CU.FT?       [1]------> 22Ø.
FIXED COST / SCHEDULED HOUR      [Ø]------> 36.72
VARIABLE COST/ MACHINE HOUR      [Ø]------> 22.


PROCESS NO.  2 : SKIDDING
================
MACHINE TYPE  5
---------------

NAME OF MACHINE TYPE             : CAT 528 GRAB-SKIDDER
AVERAGE PROCESSING TIME / TREE   :      .ØØØØ
FIXED CONSTANT TIME / LOAD       :      .1ØØØ
FIXED CONST. TIME / ONE WAY HAUL :      .ØØØØ
MACHINE CAPACITY IN CU.FT        :    29.44
FIXED COST / SCHEDULED HOUR      :    36.72
VARIABLE COST/ MACHINE HOUR      :    22.ØØ

     INPUT DATA OK (Y/N)?    [Y]---->Y



PROCESS NO.13:  FINAL TRANSPORT
==============


MACHINE TYPE 41 :
---------------
NAME OF MACHINE TYPE ?              ------> LOG TRUCK
AVERAGE PROCESSING TIME / TREE?  [Ø]------>
```

```
FIXED CONSTANT TIME / LOAD?       [Ø]------> .5
FIXED CONST. TIME / ONE WAY HAUL? [Ø]------> 1.
MACHINE CAPACITY IN CU.FT?        [1]------> 1312.
FIXED COST / SCHEDULED HOUR       [Ø]------> 15.Ø4
VARIABLE COST/ MACHINE HOUR       [Ø]------> 36.48


PROCESS NO. 13 : FINAL TRANSPORT
================
MACHINE TYPE 41
---------------

NAME OF MACHINE TYPE              : LOG TRUCK
AVERAGE PROCESSING TIME / TREE    :     .ØØØØ
FIXED CONSTANT TIME / LOAD        :     .5ØØØ
FIXED CONST. TIME / ONE WAY HAUL  :    1.ØØØØ
MACHINE CAPACITY IN CU.FT         :  1312.ØØ
FIXED COST / SCHEDULED HOUR       :    15.Ø4
VARIABLE COST/ MACHINE HOUR       :    36.48

     INPUT DATA OK (Y/N)?    [Y]---->Y




               SEVENTH PHASE
               ================

IN THIS LAST PHASE YOU ARE ABLE TO SPECIFY THE MACHINE
BREAKDOWN PARAMETERS FOR EACH ACTIVE MACHINE. IN ORDER
TO DO SO YOU WILL HAVE TO INPUT THE CUMULATIVE FREQUENCY
DISTRIBUTION FOR THE TIME BETWEEN FAILURES AND THE
ACTUAL REPAIR TIME. EACH OF THESE TWO DISTRIBUTIONS CAN
HAVE UP TO TEN CLASSES.
 !!!  DONT FORGET THE DECIMAL POINT FOR INPUT  !!!


      (PLEASE HIT RETURN TO CONTINUE)
```

```
MACHINE TYPE  1 : CAT 227 FELLER-BUNCH
================

FREQUENCY DISTRIBUTION FOR TIMES BETWEEN FAILURES:
--------------------------------------------------


CLASS  1: CUM.REL.FREQENCY?       [0]----->20.
CLASS  1: TIME BETWEEN FAILURES?  [0]----->6.

CLASS  2: CUM.REL.FREQENCY?       [0]----->40.
CLASS  2: TIME BETWEEN FAILURES?  [0]----->12.

CLASS  3: CUM.REL.FREQENCY?       [0]----->60.
CLASS  3: TIME BETWEEN FAILURES?  [0]----->20.

CLASS  4: CUM.REL.FREQENCY?       [0]----->80.
CLASS  4: TIME BETWEEN FAILURES?  [0]----->36.

CLASS  5: CUM.REL.FREQENCY?       [0]----->100.
CLASS  5: TIME BETWEEN FAILURES?  [0]----->64.



MACHINE TYPE  1 : CAT 227 FELLER-BUNCH
=================

FREQUENCY DISTRIBUTION FOR MACHINE REPAIR TIMES:
------------------------------------------------


CLASS  1: CUM.REL.FREQENCY?       [0]----->50.
CLASS  1: REPAIR TIME?            [0]----->.5

CLASS  2: CUM.REL.FREQENCY?       [0]----->70.
CLASS  2: REPAIR TIME?            [0]----->1.

CLASS  3: CUM.REL.FREQENCY?       [0]----->80.
CLASS  3: REPAIR TIME?            [0]----->2.

CLASS  4: CUM.REL.FREQENCY?       [0]----->90.
CLASS  4: REPAIR TIME?            [0]----->5.

CLASS  5: CUM.REL.FREQENCY?       [0]----->100.
CLASS  5: REPAIR TIME?            [0]----->10.
```

FREQUENCY DISTRIBUTIONS MACHINE TYPE 1 : CAT 227 FELLER-BUNCH
===

CLASS	CUM FREQ.%	TIME BETW.FAILURE	CUM.FREQ.%	REPAIR TIME
1	20.00	6.00	50.00	.50
2	40.00	12.00	70.00	1.00
3	60.00	20.00	80.00	2.00
4	80.00	36.00	90.00	5.00
5	100.00	64.00	100.00	10.00

DISTRIBUTION OK (Y/N) ? [Y]---->Y

MACHINE TYPE 5 : CAT 528 GRAB-SKIDDER
================

FREQUENCY DISTRIBUTION FOR TIMES BETWEEN FAILURES:

CLASS 1: CUM.REL.FREQENCY? [0]----->
CLASS 1: TIME BETWEEN FAILURES? [0]----->

FREQUENCY DISTRIBUTIONS MACHINE TYPE 5 : CAT 528 GRAB-SKIDDER
===

CLASS	CUM FREQ.%	TIME BETW.FAILURE	CUM.FREQ.%	REPAIR TIME

****** DISTRIBUTION NOT USED ********

DISTRIBUTION OK (Y/N) ? [Y]---->Y

```
MACHINE TYPE 41 : LOG TRUCK
================

FREQUENCY DISTRIBUTION FOR TIMES BETWEEN FAILURES:
---------------------------------------------------


CLASS  1: CUM.REL.FREQENCY?      [Ø]----->
CLASS  1: TIME BETWEEN FAILURES?  [Ø]----->


FREQUENCY DISTRIBUTIONS MACHINE TYPE 41 : LOG TRUCK
==================================================

 CLASS     CUM FREQ.%      TIME BETW.FAILURE      CUM.FREQ.%     REPAIR TIME
-------------------------------------------------------------------------------

               ******  DISTRIBUTION NOT USED  ********

DISTRIBUTION OK (Y/N) ?  [Y]---->Y




               END OF SUBROUTINE READIN
               ========================

YOU HAVE NOW DEFINED A MODEL FOR THE MECHANIZED LOG
HARVESTING SIMULATOR. DO YOU WANT TO SAVE THIS MODEL ON
DISK? IF YOU DONT DO SO ALL YOUR WORK WILL BE LOST !!


SAVE MODEL ON DISK Y/N ?      [Y]------> Y

               !!!!  MODEL HAS BEEN SAVED  !!!!
                  PRESS RETURN TO CONTINUE
```

```
***************************************************
*                                                 *
*               >>> L O G S I M <<<               *
*             INPUT USER-INTERFACE                *
*                                                 *
***************************************************



               M A I N - M E N U
     -------------------------------------------

        DEFINING A MODEL                   = 1
        PRINT OUT A MODEL                  = 2
        EDIT A MODEL                       = 3
        EXIT THE PROGRAM                   = Ø


        ENTER CHOICE PLEASE         ------->
```

APPENDIX C

3. Example session, FRONTEND.FOR, Printout

```
*****************************************************
*                                                   *
*               >>> L O G S I M <<<                 *
*              INPUT USER-INTERFACE                 *
*                                                   *
*****************************************************


                M A I N - M E N U
       ------------------------------------------

         DEFINING A MODEL                  = 1
         PRINT OUT A MODEL                 = 2
         EDIT A MODEL                      = 3
         EXIT THE PROGRAM                  = 0


         ENTER CHOICE PLEASE         -------> 3


                SUBROUTINE PRINT
                ================

WITH THIS SUBROUTINE YOU CAN PRINTOUT THE DATA OF A
SIMULATION MODEL PREVIOUSLY DEFINED WITH SUBROUTINE
READIN.


  DO YOU WISH TO CONTINUE (Y/N) ?  [Y]-----> Y


  FILENAME OF MODEL TO BE RETRIEVED?  -----> TEST1.MOD

    !!!           PLEASE WAIT A MOMENT          !!!


    !!!  MODEL HAS BEEN SUCCESSFULLY RETRIEVED  !!!


  OUTPUT SHOULD BE ROUTED TO:

   SCREEN                   = 1
   SCREEN & PRINTER         = 2

  PLEASE ENTER CHOICE -----> 1
```

```
>>>>   FOR EXAMPLE OUTPUTS SEE APPENDIX E   <<<


     **************************************************
     *                                                *
     *              >>> L O G S I M <<<               *
     *             INPUT USER-INTERFACE               *
     *                                                *
     **************************************************



                 M A I N - M E N U
        ---------------------------------------------

            DEFINING A MODEL                  = 1
            PRINT OUT A MODEL                 = 2
            EDIT A MODEL                      = 3
            EXIT THE PROGRAM                  = 0


            ENTER CHOICE PLEASE        -------> 0
```

APPENDIX C

4. Example session, FRONTEND.FOR, Modify

```
**************************************************
*                                                *
*            >>> L O G S I M <<<                 *
*            INPUT USER-INTERFACE                *
*                                                *
**************************************************




            M A I N - M E N U
    ------------------------------------------

       DEFINING A MODEL                 = 1
       PRINT OUT A MODEL                = 2
       EDIT A MODEL                     = 3
       EXIT THE PROGRAM                 = 0


       ENTER CHOICE PLEASE       -------> 3





            SUBROUTINE MODIFY
            =================

THIS SUBROUTINE ALLOWS YOU TO MODIFY THE DATA OF A
SIMULATION MODEL PREVIOUSLY DEFINED WITH SUBROUTINE
READIN.



  DO YOU WISH TO CONTINUE (Y/N) ?  [Y]-----> Y


  FILENAME OF MODEL TO BE RETRIEVED?  -----> TEST1.MOD

     !!!          PLEASE WAIT A MOMENT          !!!


     !!!   MODEL HAS BEEN SUCCESSFULLY RETRIEVED  !!!
```

```
   SUBRROUTINE MODIFY CHOICES:
   ==========================

EDIT SYSTEM PARAMETERS                  = 1
EDIT MATERIAL FREQUENCY DISTRIBUTIONS   = 2
EDIT PROCESS PARAMETERS                 = 3
EDIT MACHINE PARAMETERS                 = 4
EDIT MACHINE DISTRIBUTIONS              = 5
SAVE MODIFYIED MODEL                    = 6
RETURN TO MAIN MENU                     = 0

      PLEASE ENTER CHOICE        -----> 1
```

```
EDITING SYSTEM PARAMETERS:
--------------------------

      NAME OF SIMULATION MODEL         : TEST1.MOD
1 =   AMOUNT TO BE HARVESTED (CU.FT.)  :   25640.
2 =   TIME DELAY PARAMETER             :        .0100
0 =   RETURN TO MODIFY MENU

PLEASE ENTER CHOICE   ----> 1

HOW MANY CU.FT SHOULD BE HARVESTED? --->23450.
```

```
EDITING SYSTEM PARAMETERS:
--------------------------

      NAME OF SIMULATION MODEL         : TEST1.MOD
1 =   AMOUNT TO BE HARVESTED (CU.FT.)  :   23450.
2 =   TIME DELAY PARAMETER             :        .0100
0 =   RETURN TO MODIFY MENU

PLEASE ENTER CHOICE   ----> 0
```

```
   SUBRROUTINE MODIFY CHOICES:
   ==========================

EDIT SYSTEM PARAMETERS                  = 1
EDIT MATERIAL FREQUENCY DISTRIBUTIONS   = 2
EDIT PROCESS PARAMETERS                 = 3
```

```
EDIT MACHINE PARAMETERS            = 4
EDIT MACHINE DISTRIBUTIONS         = 5
SAVE MODIFYIED MODEL               = 6
RETURN TO MAIN MENU                = 0

       PLEASE ENTER CHOICE       -----> 3
```

```
EDITING PROCESS PARAMETERS:
---------------------------

 1  =   PROCESS NO. 1 :FELLING
 2  =   PROCESS NO. 2 :SKIDDING
13  =   PROCESS NO.13 :FINAL TRANSPORT
 0  = RETURN TO MODIFY MENU

PLEASE ENTER CHOICE   ----> 2
```

```
          PROCESS NO. 2: SKIDDING
          ==============

INCOMING ORIGIN          :  PROCESS NO. 1.  FELLING
OUTGOING DESTINATION     :  PROCESS NO.13.  FINAL TRANSPORT
DISTRIBUTION USED        :       1.          WHOLE TREES
STARTUP-INVENTORY LEVEL  :     880.0
MINIMUM INVENTORY LEVEL  :     220.0
STARTUP LEVEL MINIMUM    :     220.0
MAXIMUM INVENTORY LEVEL  :    1600.0
STARTUP LEVEL MAXIMUM    :    1600.0
LOADER USED              :  NONE
TIME DELAYS HANDELED BY  :  STANDARD BUILD-IN FUNCTIONS

CONTINUE EDITING? Y/N   [N]----> Y
```

```
EDITING PROCESS NO. 2
---------------------
NAME OF PROCESS?    -----> SKIDDING
NO. OF DISTRIBUTION TO USE?               -----> 1
STARTUP-INVENTORY LEVEL?             [1]-----> 1000.
MINNIMUM INFEED INVENTORY LEVEL?     [0]-----> 220.
STARTUP-INV.LEVEL AFTER MINIMUM?     [0]-----> 220.
MAXIMUM INFEED INV. LEVEL?    [999999.9]-----> 800.
STARTUP-INV.LEVEL AFTER MAXIMUM? [999999.9]--> 800.
```

```
WHAT LOADER DO YOU WANT TO USE (32-36) ?    [Ø]----->
BUILD-IN MODEL=Ø  OR  USERFUNCTION=1 ?      [Ø]----->



         PROCESS NO. 2: SKIDDING
         ==============

 INCOMING ORIGIN          :  PROCESS NO. 1.  FELLING
 OUTGOING DESTINATION     :  PROCESS NO.13.  FINAL TRANSPORT
 DISTRIBUTION USED        :       1.          WHOLE TREES
 STARTUP-INVENTORY LEVEL  :    1ØØØ.Ø
 MINIMUM INVENTORY LEVEL  :     22Ø.Ø
 STARTUP LEVEL MINIMUM    :     22Ø.Ø
 MAXIMUM INVENTORY LEVEL  :     8ØØ.Ø
 STARTUP LEVEL MAXIMUM    :     8ØØ.Ø
 LOADER USED              :  NONE
 TIME DELAYS HANDELED BY  :  STANDARD BUILD-IN FUNCTIONS

 CONTINUE EDITING? Y/N   [N]----> N




 EDITING PROCESS PARAMETERS:
 ---------------------------

  1  =   PROCESS NO. 1 :FELLING
  2  =   PROCESS NO. 2 :SKIDDING
 13  =   PROCESS NO.13 :FINAL TRANSPORT
  Ø  = RETURN TO MODIFY MENU

 PLEASE ENTER CHOICE   ----> Ø




    SUBRROUTINE MODIFY CHOICES:
    ===========================

 EDIT SYSTEM PARAMETERS                  = 1
 EDIT MATERIAL FREQUENCY DISTRIBUTIONS   = 2
 EDIT PROCESS PARAMETERS                 = 3
 EDIT MACHINE PARAMETERS                 = 4
 EDIT MACHINE DISTRIBUTIONS              = 5
 SAVE MODIFYIED MODEL                    = 6
 RETURN TO MAIN MENU                     = Ø
```

```
PLEASE ENTER CHOICE       -----> 4




EDITING MACHINE PARAMETERS:
---------------------------

PLEASE ENTER THE NUMBER OF THE MACHINE YOU
WANT TO EDIT. IF THE MACHINE HAS NOT BEEN
SET ACTIVE PREVIOUSLY  YOU CAN ACTIVATE
IT NOW BY SPECIFYING THE INITIAL NUMBER OF
MACHINES GREATER THAN 0.
1-42 = MACHINE NUMBER
 0   = RETURN TO MODIFY MENU

PLEASE ENTER CHOICE   ----> 5



PROCESS NO. 2: SKIDDING
**************
MACHINE TYPE  5: CAT 528 GRAB-SKIDDER
----------------

INITIAL NUMBER OF MACHINES        :    2.
AVERAGE PROCESSING TIME / TREE    :     .0000
FIXED CONSTANT TIME / LOAD        :     .0800
FIXED CONST. TIME / ONE WAY HAUL  :     .0000
MACHINE CAPACITY IN CU.FT         :    29.44
FIXED COST / SCHEDULED HOUR       :    36.72
VARIABLE COST/ MACHINE HOUR       :    22.00

CONTINUE EDITING? Y/N   [N]----> Y


MACHINE TYPE  5 :
---------------
NAME OF MACHINE TYPE ?              ------> CAT 528 GRAB-SKIDDER
INITIAL NUMBER OF MACHINES ?        ------> 2
AVERAGE PROCESSING TIME / TREE?  [0]------> .095
FIXED CONSTANT TIME / LOAD?      [0]------>
FIXED CONST. TIME / ONE WAY HAUL? [0]------>
MACHINE CAPACITY IN CU.FT?       [1]------> 29.44
FIXED COST / SCHEDULED HOUR      [0]------> 36.72
VARIABLE COST/ MACHINE HOUR      [0]------> 28.52
```

```
MACHINE  NO. 2: SKIDDING
=============
MACHINE TYPE  5: CAT 528 GRAB-SKIDDER
----------------

INITIAL NUMBER OF MACHINES        :    2.
AVERAGE PROCESSING TIME / TREE    :      .0000
FIXED CONSTANT TIME / LOAD        :      .0950
FIXED CONST. TIME / ONE WAY HAUL  :      .0000
MACHINE CAPACITY IN CU.FT         :    29.44
FIXED COST / SCHEDULED HOUR       :    36.72
VARIABLE COST/ MACHINE HOUR       :    28.52

CONTINUE EDITING? Y/N   [N]----> N



EDITING MACHINE PARAMETERS:
---------------------------

PLEASE ENTER THE NUMBER OF THE MACHINE YOU
WANT TO EDIT. IF THE MACHINE HAS NOT BEEN
SET ACTIVE PREVIOUSLY  YOU CAN ACTIVATE
IT NOW BY SPECIFYING THE INITIAL NUMBER OF
MACHINES GREATER THAN 0.
1-42 = MACHINE NUMBER
 0   = RETURN TO MODIFY MENU

PLEASE ENTER CHOICE   ----> 0



   SUBRROUTINE MODIFY CHOICES:
   ==========================

EDIT SYSTEM PARAMETERS                  = 1
EDIT MATERIAL FREQUENCY DISTRIBUTIONS   = 2
EDIT PROCESS PARAMETERS                 = 3
EDIT MACHINE PARAMETERS                 = 4
EDIT MACHINE DISTRIBUTIONS              = 5
SAVE MODIFYIED MODEL                    = 6
RETURN TO MAIN MENU                     = 0

       PLEASE ENTER CHOICE       -----> 0
```

```
                END OF SUBROUTINE MODIFY
                ========================

YOU HAVE TO SAVE THE EDITED MODEL ON DISK,
OTHERWISE ALL YOUR WORK WILL BE LOST !!


SAVE MODEL ON DISK Y/N ?        [Y]------> Y
!!!!  FILE: TEST1.MOD            ALREADY EXISTS !!!!

OVERWRITE OLD FILE?             [N]------> Y


                !!!!  MODEL HAS BEEN SAVED  !!!!
                    PRESS RETURN TO CONTINUE




   SUBRROUTINE MODIFY CHOICES:
   ===========================

EDIT SYSTEM PARAMETERS                  = 1
EDIT MATERIAL FREQUENCY DISTRIBUTIONS   = 2
EDIT PROCESS PARAMETERS                 = 3
EDIT MACHINE PARAMETERS                 = 4
EDIT MACHINE DISTRIBUTIONS              = 5
SAVE MODIFYIED MODEL                    = 6
RETURN TO MAIN MENU                     = Ø

       PLEASE ENTER CHOICE        -----> Ø




        ************************************************
        *                                              *
        *             >>> L O G S I M <<<              *
        *             INPUT USER-INTERFACE             *
        *                                              *
        ************************************************
```

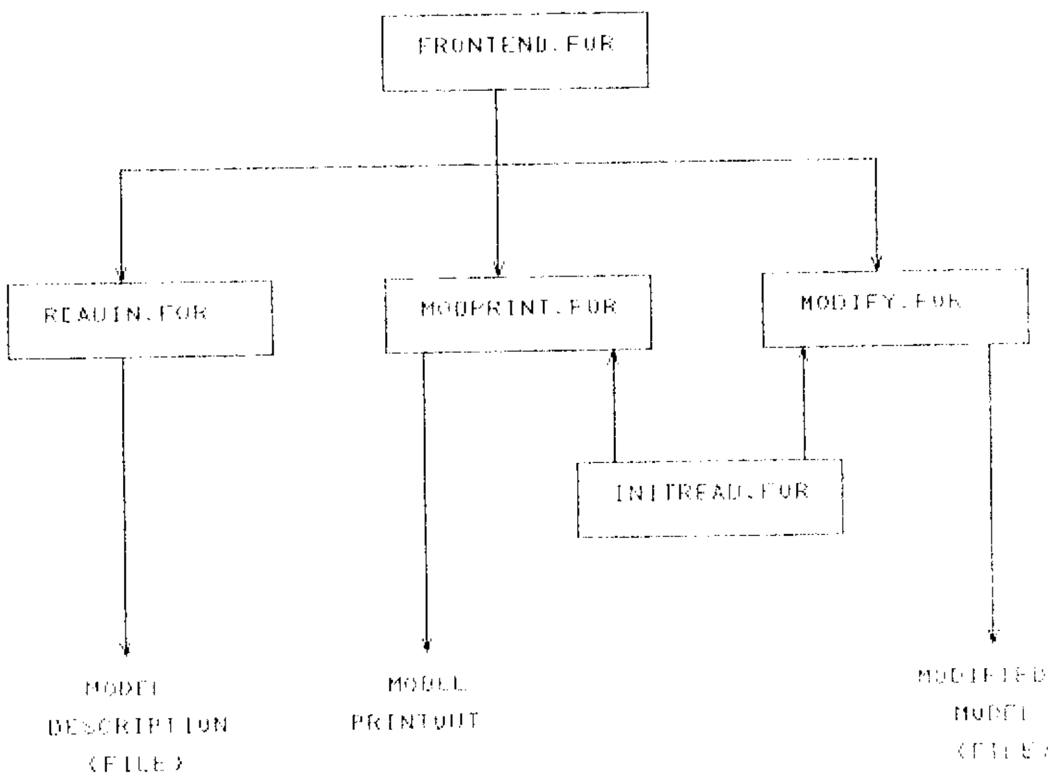
```
          M A I N - M E N U
--------------------------------------------

   DEFINING A MODEL                  = 1
   PRINT OUT A MODEL                 = 2
   EDIT A MODEL                      = 3
   EXIT THE PROGRAM                  = Ø


     PLEASE ENTER CHOICE        -----> Ø
```

APPENDIX D

TABLE OF CONTENTS:

APPENDIX D

1. Figure, FORTRAN filestructure FRONTEND.EXE



APPENDIX D

2. Listing, FRONTEND.FOR

```
C***************************************************************
C*                                                             *
C*                OREGON STATE UNIVERSITY                      *
C*                      JUNE 1986                              *
C*                                                             *
C*                >>>  L O G S I M <<<                         *
C*                                                             *
C*    SIMULATION OF MECHANIZED LOG HARVESTING SYSTEMS          *
C*                                                             *
C*                                                             *
C*    DESIGNED BY : CHRISTOPH WIESE                            *
C*                  MASTERS CANDIDATE, DEP. OF INDUSTRIAL      *
C*                  ENGINEERING, OREGON STATE UNIVERSITY       *
C*                                                             *
C*    DESIGNED FOR: DEPARTMENT OF FOREST ENGINEERING           *
C*                  OREGON STATE UNIVERSITY                    *
C*                                                             *
C*                                                             *
C*    SUPERVISION : DR. ELDON OLSEN                            *
C*                  ASSOCIATE PROFESSOR, DEP. OF INDUSTRIAL    *
C*                  ENGINEERING, OREGON STATE UNIVERSITY       *
C*                                                             *
C*                                                             *
C***************************************************************
C*                                                             *
C* MAIN PROGRAM OF THE CUTOMIZED SLAM II PROCESSOR: LOGSIM.EXE *
C*                                                             *
C*                    31-MAY-87  18:55                         *
C*                                                             *
C***************************************************************
C
C
C
C METACOMMANDS:
C ============
C
$INCLUDE:'PRCTL.FOR'
C
C PROGRAM DECLARATION:
C ===================
C
      PROGRAMM HARVEST
C
C
C
C COMMON BLOCK:
C ============
C
$INCLUDE:'VARBLOCK.DOC'
C
C
```

```
C INITIALIZE SYSTEM PARAMETERS:
C ---------------------------
C
      NCRDR=5
      NPRNT=Ø
      NTAPE=7
      XXLEVEL(1)=15
      XXLEVEL(2)=27
      XXLEVEL(3)=39
      XXLEVEL(4)=51
      XXLEVEL(5)=63
      XXLEVEL(6)=75
      XXLEVEL(7)=87
C
C
C CALL SLAM SIMULATION PROCESSOR
C ----------------------------
C
C
      CALL SLAM
C
C
C
C FORMAL END OF PROGRAM
C ---------------------
C
C
C
      STOP ' '
9998  END
```

APPENDIX D

3. Listing, READIN.FOR

```
C***************************************************************
C*                                                              *
C*                 OREGON STATE UNIVERSITY                      *
C*                        JUNE 1986                             *
C*                                                              *
C*                  >>> L O G S I M <<<                         *
C*                                                              *
C*     SIMULATION OF MECHANIZED LOG HARVESTING SYSTEMS          *
C*                                                              *
C*                                                              *
C*     DESIGNED BY : CHRISTOPH WIESE                            *
C*                   MASTERS CANDIDATE, DEP. OF INDUSTRIAL      *
C*                   ENGINEERING, OREGON STATE UNIVERSITY       *
C*                                                              *
C*     DESIGNED FOR: DEPARTMENT OF FOREST ENGINEERING           *
C*                   OREGON STATE UNIVERSITY                    *
C*                                                              *
C*                                                              *
C*     SUPERVISION : DR. ELDON OLSEN                            *
C*                   ASSOCIATE PROFESSOR, DEP. OF INDUSTRIAL    *
C*                   ENGINEERING, OREGON STATE UNIVERSITY       *
C*                                                              *
C*                                                              *
C***************************************************************
C*                                                              *
C*         FORTRAN INPUT USER-INTERFACE: READIN.FDR             *
C*                                                              *
C*                    31-MAY-87  18:55                          *
C*                                                              *
C***************************************************************
C
C
C
C
C
$INCLUDE:'PRCTL.FOR'
C
C PROGRAM DECLARATION:
C ====================
C
      SUBROUTINE READIN
C
C COMMON BLOCK :
C --------------
C
$INCLUDE:'VARBLOCK.DOC'
C
C DEFINE LOCAL VARIABLES, NAMES & TYPE:
C ------------------------------------
C
```

```
      INTEGER*4 IANSWER,INDEX1,INDEX2,INDEX3,INDEX4,INDEX5
      INTEGER*4 INDEX6,INDEX7,INDEX8,INDEX9
      CHARACTER*20 CHRANSWER,CHR1ANSWER
      REAL F1ANSWER,F2ANSWER,F3ANSWER,F4ANSWER,F5ANSWER,F6ANSWER
      REAL F7ANSWER,F8ANSWER,F9ANSWER,F10ANSWER,F11ANSWER,F12ANSWER
      LOGICAL*4 FILESTATUS
C
C BEGIN PROCESSING:
C ================
C
C OPENING SCREEN:
C ---------------
C
      WRITE(*,100)
100   FORMAT('1'////5X,
     1'                    SUBROUTINE READIN'/5X,
     2'                    ================='//5X,
     3'THIS IS THE SUBROUTINE TO DEFINE THE SIMULATION MODEL, LATER'/5X,
     4'USED BY THE SLAM PROCESSOR. THE FOLLOWING DATA IS NECCESSARY'/5X,
     5'TO DEFINE A HARVESTING MODEL:'//5X,
     6'   - HARVESTING CONFIGURATION, GENERAL PARAMETERS'/5X,
     7'   - DISTRIBUTION DATA OF THE TREES/LOGS/PULPLOGS/SAWLOGS')
      WRITE(*,101)
101   FORMAT(5X,
     1'   - MAXIMUM/MINIMUM DATA OF MATERIAL BUFFERS PER PROCESS'/5X,
     2'   - HOW MANY MACHINES USED PER PROCESS'/5X,
     3'   - MACHINE DATA: TIMES, BREAKDOWN FREQUENCIES, COSTS ETC'///5X,
     4'DEFAULT VALUES WILL BE GIVEN IN SQUARE BRAKETS [  ]. IF YOU'/5X,
     5'WANT TO USE THEM, HIT SPACE BAR AND PRESS RETURN.'////5X)
      WRITE(*,'(A\)') '   DO YOU WISH TO CONTINUE (Y/N) ?    [Y]-----> '
      READ (*,'(BZ,A1)') CHRANSWER
      IF (CHRANSWER.EQ.'N') THEN
        GOTO 9998
      ELSE
        CONTINUE
      ENDIF
C
C
C
C INITIALIZATION OF ALL VARIABLES
C ===============================
C
C
      DO 110 INDEX1=1,100,1
        XX(INDEX1)=0
110   CONTINUE
C
      DO 114 INDEX1=1,26,1
      DO 112 INDEX2=1,42,1
        USERARR(INDEX1,INDEX2)=0
112   CONTINUE
114   CONTINUE
C
```

```
      DO 118 INDEX1=1,8,1
      DO 116 INDEX2=1,10,1
        DISARR(INDEX1,INDEX2)=0
116   CONTINUE
118   CONTINUE

C
      DO 124 INDEX1=1,42,1
      DO 122 INDEX2=1,4,1
      DO 120 INDEX3=1,10,1
        MCHARR(INDEX1,INDEX2,INDEX3)=0
120   CONTINUE
122   CONTINUE
124   CONTINUE
C
      DO 126 INDEX1=1,52,1
        MCHNAMES(INDEX1)=' '
126   CONTINUE
C
      DO 128 INDEX1=1,20,1
        PROCNAMES(INDEX1)=' '
128   CONTINUE
C
      DO 130 INDEX1=1,4,1
        DISTRIBNAMES(INDEX1)=' '
130   CONTINUE
C
C
C
C
C BEGIN OF DEFINING THE MODEL
C ==========================
C
C HARVESTING CONFIGURATION AND GENERAL PARAMETERS:
C -----------------------------------------------
C
      WRITE(*,1000)
1000  FORMAT('1'////////////////////////////////,
     1 20X' FIRST PHASE',/,
     2 20X'=============',//,
     3'  SPECIFICATION OF GENERAL SYSTEM PARAMETERS'/)
C
C
1010  WRITE(*,'(//,A,\)')' WHAT IS THE FILENAME OF THE MODEL?  --->'
      READ(*,'(A20)')FILENAME
      IF(FILENAME.EQ.' ') THEN
       WRITE (*,'(/,A)')'  !!! CANNOT BE, PLEASE TRY AGAIN !!!'
       GOTO 1010
      ENDIF
C
C
```

```
1020  WRITE(*,'(//,A,\)')' HOW MANY CU.FT SHOULD BE HARVESTED? --->'
      READ(*,'(F8.0)')F1ANSWER
      IF(F1ANSWER.EQ.0) THEN
       WRITE (*,'(/,A)')'  !!! CANNOT BE, PLEASE TRY AGAIN !!!'
       GOTO 1020
      ELSEIF(F1ANSWER.GE.9999998) THEN
       WRITE (*,'(/,A)')'  !!! CANNOT BE, PLEASE TRY AGAIN !!!'
       GOTO 1020
      ELSE
       XX(4)=F1ANSWER
      ENDIF
C
C
1030  WRITE(*,'(//,A,\)')' VALUE OF THE TIME DELAY PARAMETER?  --->'
      READ(*,'(F8.4)')F1ANSWER
      IF(F1ANSWER.EQ.0) THEN
       WRITE (*,'(/,A)')'  !!! CANNOT BE, PLEASE TRY AGAIN !!!'
       GOTO 1030
      ELSEIF(F1ANSWER.GE.999.OR.F1ANSWER.LT.0.0001) THEN
       WRITE (*,'(/,A)')'  !!! CANNOT BE, PLEASE TRY AGAIN !!!'
       GOTO 1030
      ELSE
       XX(10)=F1ANSWER
      ENDIF
C
C
1100  WRITE(*,1110)
1110  FORMAT(/////////////////////////////,20X,
     1'SECOND PHASE'/19X'============='///,
     2' WE NOW ARE GOING TO DEFINE THE MATERIAL FLOW THROUGH'/,
     3' THE HARVESTING SYSTEM. FOR EACH PROCESS PLEASE STATE'/,
     4' THE PROCESS FROM WHICH THE INCOMING MATERIALS STREAM'/,
     5' ORIGINATES AND THE DESTINATION OF THE OUTGOING MATERIAL'/,
     6' STREAM. A VALUE OF 0 FOR BOTH QUESTIONS MEANS THAT THE'/,
     7' PROCESS IS NOT USED.'//)
      WRITE(*,'(A\)') '              (PLEASE HIT RETURN TO CONTINUE)'
      READ (*,'(BZ,I6)') IANSWER
      WRITE(*,'(/////////////////////////////)')
C
C
C
      DO 1350 INDEX1=1,13,1
C
1200  IF(INDEX1.EQ.1)THEN
        WRITE(*,'(A)')' PROCESS 1:'
        WRITE(*,'(A)')' ----------'
        WRITE(*,'(A\)')' OUTGOING DESTINATION?  --->'
        READ(*,'(BN,F8.0)')F1ANSWER
          IF(F1ANSWER.LE.1.OR.F1ANSWER.GT.13.)THEN
            WRITE(*,'(/,A,/)')' !! OUT CANNOT BE, PLEASE TRY AGAIN !!'
            GOTO 1200
          ELSE
```

```
          USERARR(23,1)=F1ANSWER
          USERARR(24,1)=0
        ENDIF
C
      ELSEIF(INDEX1.GE.2.AND.INDEX1.LE.10)THEN
        WRITE(*,1210)INDEX1
1210    FORMAT(//,' PROCESS ',I2,':',/,' ----------',/)
        WRITE(*,'(A\)')' INCOMING ORIGIN?       --->'
        READ(*,'(BN,F8.0)') F1ANSWER
        WRITE(*,'(A\)') ' OUTGOING DESINATION?   --->'
        READ(*,'(BN,F8.0)') F2ANSWER
          IF(F1ANSWER.LT.0.OR.F1ANSWER.GT.11.OR.F1ANSWER.EQ.INDEX1)THEN
            WRITE(*,1220)
1220        FORMAT(/,'  !! IN CANNOT BE, PLEASE TRY AGAIN !!'/)
            GOTO 1200
          ELSEIF(F2ANSWER.LT.0.OR.F2ANSWER.GT.13.OR.F2ANSWER.EQ.INDEX1)
     1    THEN
            WRITE(*,1230)
1230        FORMAT(/,'  !! OUT CANNOT BE, PLEASE TRY AGAIN !!'/)
            GOTO 1200
          ELSE
            USERARR(23,INDEX1)=F2ANSWER
            USERARR(24,INDEX1)=F1ANSWER
          ENDIF
C
      ELSEIF(INDEX1.EQ.11)THEN
        WRITE(*,1240)INDEX1
1240    FORMAT(//,' PROCESS ',I2,':',/,' ----------',/)
        WRITE(*,'(A\)')' INCOMING ORIGIN?       --->'
        READ(*,'(BN,F8.0)') F1ANSWER
        WRITE(*,'(A\)') ' OUTGOING DESINATION, ROUTE 1 ?   --->'
        READ(*,'(BN,F8.0)') F2ANSWER
        WRITE(*,'(A\)') ' OUTGOING DESINATION, ROUTE 2 ?   --->'
        READ(*,'(BN,F8.0)') F3ANSWER
          IF(F1ANSWER.LT.0.OR.F1ANSWER.GT.11.OR.F1NSWER.EQ.INDEX1)THEN
            WRITE(*,1250)
1250        FORMAT(/,'  !! IN CANNOT BE, PLEASE TRY AGAIN !!'/)
            GOTO 1200
          ELSEIF(F2ANSWER.LT.0.OR.F2ANSWER.GT.13.OR.F2ANSWER.EQ.INDEX1)
     1    THEN
            WRITE(*,1260)
1260        FORMAT(/,'  !!  OUT 1 CANNOT BE, PLEASE TRY AGAIN !!'/)
            GOTO 1200
          ELSEIF(F3ANSWER.LT.0.OR.F3ANSWER.GT.13.OR.F3ANSWER.EQ.INDEX1)
     1    THEN
            WRITE(*,1270)
1270        FORMAT(/,'  !!  OUT 2 CANNOT BE, PLEASE TRY AGAIN !!'/)
            GOTO 1200
          ELSE
            USERARR(24,INDEX1)=F1ANSWER
            USERARR(7,5)=F2ANSWER
            USERARR(7,6)=F3ANSWER
          ENDIF
```

```
C
      ELSEIF(INDEX1.EQ.12)THEN
        WRITE(*,1280)INDEX1
1280    FORMAT(//,' PROCESS ',I2,':',/,' ----------',/)
        WRITE(*,'(A\)')' INCOMING ORIGIN?       --->'
        READ(*,'(BN,F8.0)') F1ANSWER
          IF(F1ANSWER.LT.0.OR.F1ANSWER.GT.11.OR.F1ANSWER.EQ.INDEX1)THEN
            WRITE(*,1290)
1290        FORMAT(/,'  !! IN CANNOT BE, PLEASE TRY AGAIN !!'/)
            GOTO 1200
          ELSE
            USERARR(24,INDEX1)=F1ANSWER
            USERARR(23,INDEX1)=0
          ENDIF
C
      ELSE
        WRITE(*,1300)INDEX1
1300    FORMAT(//,' PROCESS ',I2,':',/,' ----------',/)
        WRITE(*,'(A\)')' INCOMING ORIGIN?       --->'
        READ(*,'(BN,F8.0)') F1ANSWER
          IF(F1ANSWER.LT.0.OR.F1ANSWER.GT.11.OR.F1ANSWER.EQ.INDEX1)THEN
            WRITE(*,1310)
1310        FORMAT(/,'  !! IN CANNOT BE, PLEASE TRY AGAIN !!'/)
            GOTO 1200
          ELSE
            USERARR(24,INDEX1)=F1ANSWER
            USERARR(23,INDEX1)=0
          ENDIF
        ENDIF
       CONTINUE
C
1350  CONTINUE
C
C
      WRITE(*,1400)
1400  FORMAT(/////////,' PROCESS #     IN ORIGIN    OUT DESTINATION'/,
     1' ---------------------------------------------')
C
      DO 1500 INDEX1=1,13,1
C
       IF(USERARR(23,INDEX1).EQ.0.AND.USERARR(24,INDEX1).EQ.0)THEN
         WRITE(*,'(4X,I2)')INDEX1
         GOTO 1450
       ENDIF
       IF(INDEX1.EQ.1) THEN
         WRITE (*,1410)INDEX1,USERARR(23,INDEX1)
1410     FORMAT (4X,I2,23X,F3.0)
       ELSEIF(INDEX1.GE.2.AND.INDEX1.LE.10) THEN
         WRITE (*,1420)INDEX1,USERARR(24,INDEX1),USERARR(23,INDEX1)
1420     FORMAT (4X,I2,10X,F3.0,10X,F3.0)
       ELSEIF(INDEX1.EQ.11) THEN
         WRITE (*,1430)INDEX1,USERARR(24,INDEX1),USERARR(7,5),
     1   USERARR(7,6)
```

```
1430     FORMAT (4X,I2,10X,F3.0,10X,F3.0,2X,F3.0)
        ELSE
         WRITE (*,1440)INDEX1,USERARR(24,INDEX1)
1440     FORMAT (4X,I2,10X,F3.0)
        ENDIF
1450    CONTINUE
1500   CONTINUE
C
       WRITE (*,1550)
1550   FORMAT(/,' IS THIS CORRECT? (Y/N)      [Y]----->',\)
       READ (*,'(A1)')CHRANSWER
       IF(CHRANSWER.EQ.'N') THEN
         GOTO 1100
       ELSE
         CONTINUE
       ENDIF
C
C
       DO 1610 INDEX1=1,13,1
C
       IF(INDEX1.EQ.1) THEN
         INDEX2=USERARR(23,1)
         IF(USERARR(24,INDEX2).NE.INDEX1) GOTO 1650
       ELSEIF(INDEX1.GE.2.AND.INDEX1.LE.10) THEN
         IF(USERARR(23,INDEX1).EQ.0.AND.USERARR(24,INDEX1).EQ.0)GOTO 1600
         INDEX2=USERARR(23,INDEX1)
         IF(USERARR(24,INDEX2).NE.INDEX1) GOTO 1650
         INDEX2=USERARR(24,INDEX1)
         IF(INDEX2.EQ.11) GOTO 1600
         IF(USERARR(23,INDEX2).NE.INDEX1) GOTO 1650
       ELSEIF(INDEX1.EQ.11) THEN
         IF(USERARR(24,INDEX1).EQ.0.AND.USERARR(7,5).EQ.0.AND.
      1    USERARR(7,6).EQ.0) GOTO 1600
         INDEX2=USERARR(7,5)
         IF(USERARR(24,INDEX2).NE.INDEX1) GOTO 1650
         INDEX2=USERARR(7,6)
         IF(USERARR(24,INDEX2).NE.INDEX1) GOTO 1650
       ELSEIF(INDEX1.EQ.12) THEN
         IF(USERARR(23,INDEX1).EQ.0.AND.USERARR(24,INDEX1).EQ.0)GOTO 1600
         INDEX2=USERARR(24,INDEX1)
         IF(INDEX2.EQ.11) GOTO 1600
         IF(USERARR(23,INDEX2).NE.INDEX1) GOTO 1650
       ELSEIF(INDEX1.EQ.13) THEN
         INDEX2=USERARR(24,INDEX1)
         IF(USERARR(23,INDEX1).EQ.0.AND.USERARR(24,INDEX1).EQ.0)GOTO 1600
         IF(INDEX2.EQ.11) GOTO 1600
         IF(USERARR(23,INDEX2).NE.INDEX1) GOTO 1650
       ELSEIF(USERARR(23,12).EQ.0.AND.USERARR(24,12).EQ.0.AND.
      1  USERARR(23,13).EQ.0.AND.USERARR(24,13).EQ.0) THEN
         GOTO 1650
       ELSE
         CONTINUE
       ENDIF
```

```
1600  CONTINUE
1610  CONTINUE
C
      GOTO 2000
C
C
1650   WRITE(*,1660) INDEX1
1660   FORMAT(//,
     1'  !!!!   LOGICAL ERROR IN PROCESS NO.'I2'    !!!!'/
     2'  !!!!   PLEASE ENTER FROM THE BEGINNING   !!!!'///)
      WRITE(*,'(A\)') '     (PLEASE HIT RETURN TO CONTINUE)'
      READ (*,'(BZ,I6)') IANSWER
      GOTO 1100
C
C
C
C DISTRIBUTION DATA OF THE TREES/LOGS/PULPLOGS/SAWLOGS
C ---------------------------------------------------
C
2000  WRITE(*,2010)
2010   FORMAT(/////////////////////////////,20X,
     1'THIRD PHASE'/20X'==========='//,
     2' NOW WE DEFINE THE CUMULATIVE FREQUENCY DISTRIBUTIONS '/,
     3' USED TO DESCRIBE TREES, LOGS ETC..'/,
     4' YOU CAN SPECIFY UP TO FOUR DIFFERENT FREQUENCY DISTRIBUTIONS,'/,
     5' WITH 10 FREQUENCY CLASSES EACH. YOU HAVE TO SPECIFY AT LEAST'/,
     6' ONE CLASS IN ONE  DISTRIBUTION.'/,
     7' !!! DONT FORGET THE DECIMAL POINT FOR INPUT !!!',//)
      WRITE(*,'(A\)') '             (PLEASE HIT RETURN TO CONTINUE)'
      READ (*,'(BN,I6)') IANSWER
C
C
C
      DO 2300 INDEX1=1,4,1
C
        IF(INDEX1.EQ.1)THEN
          INDEX8=1
          INDEX9=2
        ELSEIF(INDEX1.EQ.2)THEN
          INDEX8=3
          INDEX9=4
        ELSEIF(INDEX1.EQ.3)THEN
          INDEX8=5
          INDEX9=6
        ELSEIF(INDEX1.EQ.4)THEN
          INDEX8=7
          INDEX9=8
        ELSE
          CONTINUE
        ENDIF
C
```

```
2100    WRITE(*,2110)INDEX1
2110    FORMAT(////////////////////////////////////////,
     1  ' FREQUENCY DISTRIBUTION NO.'I2,':'/,
     2  ' ----------------------------')
C
        WRITE(*,2120)
2120    FORMAT(/,'  NAME OF THIS DISTRIBUTION ?   -----> ',\)
        READ(*,'(A)')CHRANSWER
        DISTRIBNAMES(INDEX1)=CHRANSWER
C
2130     DO 2170 INDEX2=1,10,1
2140      WRITE(*,2150)INDEX2
2150      FORMAT(/,' CLASS ',I2,': CUM.REL.FREQENCY?  [0]--->'\)
          READ(*,'(BN,F8.2)')F1ANSWER
          WRITE(*,2160)INDEX2
2160      FORMAT(' CLASS ',I2,': VOLUME CU.FT?      [0]--->'\)
          READ(*,'(BN,F8.2)')F2ANSWER
          IF(INDEX2.EQ.1.AND.F1ANSWER.EQ.0)THEN
             GOTO 2180
          ELSEIF(F2ANSWER.LE.0)THEN
             WRITE(*,'(/,A)')' !!  CANNOT BE, PLEASE TRY AGAIN  !!'
             GOTO 2140
          ELSEIF(F1ANSWER.GT.100)THEN
             WRITE(*,'(/,A)')' !!  CANNOT BE, PLEASE TRY AGAIN  !!'
             GOTO 2140
          ELSEIF(INDEX2.GT.1.AND.F1ANSWER.LE.F3ANSWER)THEN
             WRITE(*,'(/,A)')' !!  CANNOT BE, PLEASE TRY AGAIN  !!'
             GOTO 2140
          ELSEIF(INDEX2.EQ.10.AND.F1ANSWER.NE.100)THEN
             WRITE(*,'(/,A)')' !!  CANNOT BE, PLEASE TRY AGAIN  !!'
             GOTO 2140
          ELSE
             F3ANSWER=F1ANSWER
             DISARR(INDEX8,INDEX2)=F1ANSWER
             DISARR(INDEX9,INDEX2)=F2ANSWER
             IF(F1ANSWER.EQ.100)GOTO 2180
          ENDIF
2170     CONTINUE
C
2180    CONTINUE
C
C
C
        WRITE(*,2200)INDEX1,DISTRIBNAMES(INDEX1)
2200    FORMAT(//////////////////////////////////////////////,
     1  10X,'DISTRIBUTION NO.',
     2  I2' :  ',A,/,
     3  10X,'====================='//,
     4  '   CLASS      CUM.REL.FREQ.%         CU.FT'/,
     5  ' ------------------------------------------',)
        IF(DISARR(INDEX8,1).EQ.0)THEN
          WRITE(*,'(/,3X,A)')'*******  DISTRIBUTION NOT USED  *******'
        ELSE
```

```
          DO 2230 INDEX2=1,10,1
            IF(DISARR(INDEX8,INDEX2).EQ.0)GOTO 2230
            WRITE(*,2210)INDEX2,DISARR(INDEX8,INDEX2),
     1                   DISARRR(INDEX9,INDEX2)
2210        FORMAT(4X,I2,10X,F8.2,10X,F8.2)
2230      CONTINUE
        ENDIF
        WRITE(*,'(//,A,\)')'  DISTRIBUTION OK (Y/N) ?  [Y]---->'
        READ(*,'(A1)')CHRANSWER
        IF(CHRANSWER.EQ.'N')THEN
          DO 2240 INDEX2=1,10,1
            DISARR(INDEX8,INDEX2)=0
            DISARR(INDEX9,INDEX2)=0
2240      CONTINUE
          GOTO 2100
        ENDIF
C
C
2300  CONTINUE
C
C
C
C
      IF(DISARR(1,1).EQ.0.AND.DISARR(3,1).EQ.0.AND.DISARR(5,1).EQ.0
     1.AND.DISARR(7,1).EQ.0) THEN
        WRITE (*,2350)
2350    FORMAT(////,
     1  ' !!!!       YOU DID NOT SPECIFY ANY DISTRIBUTION      !!!!'/,
     2  ' !!!!  THIS IS NOT ALLOWED, ENTER FROM THE BEGINNING  !!!!'///)
        WRITE(*,'(A\)') '               (PLEASE HIT RETURN TO CONTINUE)'
        READ (*,'(BZ,I6)') IANSWER
        GOTO 2000
      ELSE
        CONTINUE
      ENDIF
C
C
C
C
C
C
C DESCRIPTION OF THE PROCESSES USED
C ---------------------------------
C
3000  WRITE(*,3010)
3010  FORMAT(////////////////////////////////,20X,
     1'FORTH PHASE'/20X'==========='//,
     2' IN THIS PHASE WE WILL DESCRIBE THE PROCESSES USED A LITTLE'/,
     3' BIT MORE IN DETAIL. YOU WILL BE ASKED FOR :'/,
     4'    - AN OPTIONAL NAME FOR THE PROCESS'/,
     5'    - THE DISTRIBUTION TO BE USED FOR THIS PROCESS'/,
     6'    - STARTUP-INVENTORY LEVEL FOR THE PROCESS'/,
     7'    - MINIMUM INPUT BUFFER SIZE')
```

```
      WRITE(*,3020)
3020  FORMAT(
     1'     - STARTUP-INV. LEVEL AFTER MINIMUM HAS BEEN REACHED'/,
     2'     - MAXIMUM INPUT BUFFER SIZE'/,
     3'     - STARTUP-INV. LEVEL AFTER MAXIMUM HAS BEEN REACHED'//,
     4' PLEASE REMEMBER: THE INPUT BUFFER OF A PROCESS IS THE OUT-'/,
     5' PUT BUFFER OF HIS PREVIOUS PROCESS. THE MINIMUM BUFFER SIZE')
      WRITE(*,3030)
3030  FORMAT(' ',
     1' EFFECTS THE CURRENT PROCESS, THE MAXIMUM EFFECTS THE PREVIOUS'/,
     2' ONE.'//,
     3'       !!! DONT FORGET THE DECIMAL POINT FOR INPUT !!!',//)
      WRITE(*,'(A\)') '                (PLEASE HIT RETURN TO CONTINUE)'
      READ (*,'(BZ,I6)') IANSWER
C
      F1ANSWER=0
      F2ANSWER=0
      F3ANSWER=0
      F4ANSWER=0
      F5ANSWER=0
      F6ANSWER=0
      F7ANSWER=0
C
C
C
3100  DO 3500 INDEX1=1,13,1
C
C
3105  IF(USERARR(23,INDEX1).EQ.0.AND.USERARR(24,INDEX1).EQ.0) GOTO 3490
C
      WRITE(*,3110)INDEX1
3110  FORMAT(////////////////////////' PROCESS NO.'I2,/' -------------')
      WRITE(*,'(A,\)')' NAME OF PROCESS?    -----> '
      READ(*,'(A20)')CHRANSWER
C
3120  WRITE(*,'(A,\)')' NO. OF DISTRIBUTION TO USE?              -----> '
      READ(*,'(I2)')IANSWER
      INDEX2=IANSWER*2
      INDEX2=INDEX2-1
      IF(IANSWER.LT.1.OR.IANSWER.GT.4)THEN
        WRITE(*,3130)
3130    FORMAT(/,' !!!  CANNOT BE, PLEASE TRY AGAIN  !!!')
        GOTO 3120
      ELSEIF(DISARR(INDEX2,1).EQ.0.) THEN
        WRITE(*,3135)
3135    FORMAT(/,' !!!  DISTRIBUTION NOT ACTIVE, PLEASE TRY AGAIN  !!!')
        GOTO 3120
      ELSE
        INDEX9=IANSWER
      ENDIF
C
      IF(INDEX1.EQ.1) GOTO 3260
```

```
C
3140 WRITE(*,'(A,\)')' STARTUP-INVENTORY LEVEL?            [1]-----> '
     READ(*,'(F8.1)')F1ANSWER
     IF(F1ANSWER.LT.0.OR.F1ANSWER.GE.XX(4)) THEN
       WRITE(*,3150)
3150   FORMAT(/,' !!!  CANNOT BE, PLEASE TRY AGAIN  !!!')
       GOTO 3140
     ELSEIF(F1ANSWER.EQ.0)THEN
       F1ANSWER=1
     ELSE
       CONTINUE
     ENDIF
C
3160 WRITE(*,'(A,\)')' MINNIMUM INFEED INVENTORY LEVEL?     [0]-----> '
     READ(*,'(F8.1)')F2ANSWER
     IF(F2ANSWER.LT.0.OR.F2ANSWER.GT.F1ANSWER) THEN
       WRITE(*,3170)
3170   FORMAT(/,' !!!  CANNOT BE, PLEASE TRY AGAIN  !!!')
       GOTO 3160
     ELSE
       CONTINUE
     ENDIF
C
3180 WRITE(*,'(A,\)')' STARTUP-INV.LEVEL AFTER MINIMUM?     [0]-----> '
     READ(*,'(F8.1)')F3ANSWER
     IF(F3ANSWER.LT.0.OR.F2ANSWER.GT.F3ANSWER) THEN
       WRITE(*,3190)
3190   FORMAT(/,' !!!  CANNOT BE, PLEASE TRY AGAIN  !!!')
       GOTO 3180
     ELSE
       CONTINUE
     ENDIF
C
3200 WRITE(*,'(A,\)')' MAXIMUM INFEED INV. LEVEL?    [999999.9]-----> '
     READ(*,'(F8.1)')F4ANSWER
     IF(F4ANSWER.LT.0) THEN
       WRITE(*,3210)
3210   FORMAT(/,' !!!  CANNOT BE, PLEASE TRY AGAIN  !!!')
       GOTO 3200
     ELSEIF(F4ANSWER.GT.0.AND.F3ANSWER.GE.F4ANSWER) THEN
       WRITE(*,3220)
3220   FORMAT(/,' !!!  CANNOT BE, PLEASE TRY AGAIN  !!!')
       GOTO 3200
     ELSEIF(F4ANSWER.EQ.0)THEN
       F4ANSWER=999999.9
     ELSE
       CONTINUE
     ENDIF
C
3230 WRITE(*,'(A,\)')' STARTUP-INV.LEVEL AFTER MAXIMUM? [999999.9]--> '
     READ(*,'(F8.1)')F5ANSWER
     IF(F5ANSWER.LT.0.OR.F5ANSWER.GT.F4ANSWER) THEN
       WRITE(*,3240)
```

```
3240   FORMAT(/,' !!!  CANNOT BE, PLEASE TRY AGAIN  !!!')
        GOTO 3230
      ELSEIF(F5ANSWER.GT.0.AND.F5ANSWER.LT.F2ANSWER) THEN
        WRITE(*,3250)
3250   FORMAT(/,' !!!  CANNOT BE, PLEASE TRY AGAIN  !!!')
        GOTO 3230
      ELSEIF(F5ANSWER.EQ.0)THEN
        F5ANSWER=999999.9
      ELSE
        CONTINUE
      ENDIF
C
3260  WRITE(*,3270)
3270  FORMAT(' WHAT LOADER DO YOU WANT TO USE (32-36) ?      [0]-----> '
     1  ' ',\)
      READ(*,'(I2)')INDEX8
      IF (INDEX8.EQ.0.OR.INDEX8.GE.32.AND.INDEX8.LE.36) THEN
        CONTINUE
      ELSE
        WRITE(*,3280)
3280   FORMAT(/,' !!!  CANNOT BE, PLEASE TRY AGAIN  !!!')
        GOTO 3260
      ENDIF
C
3290  WRITE(*,3300)
3300  FORMAT(' TIME DELAYS HANDELD BY',/
     1       ' BUILD-IN MODEL=0  OR  USERFUNCTION=1 ?        [0]-----> ',
     2       ' ',\)
      READ(*,'(I2)')INDEX7
      IF(INDEX7.GT.1.OR.INDEX7.LT.0) THEN
        WRITE(*,3310)
3310   FORMAT(/,' !!!  CANNOT BE, PLEASE TRY AGAIN  !!!')
        GOTO 3290
      ELSE
        CONTINUE
      ENDIF
C
C
C
C
      WRITE(*,3400)INDEX1,CHRANSWER,INDEX9,F1ANSWER,F2ANSWER,F3ANSWER,
     1             F4ANSWER,F5ANSWER
3400  FORMAT(///////////////////,
     1  '  PROCESS NO.'I2,': 'A,/' -------------'//
     2  '  DISTRIBUTION USED: 'I2,/
     3  '  STARTUP INVENTORY: ',F8.1,/
     4  '  MINIMUM INVENTORY: ',F8.1,/
     5  '  STARTUP MINIMUM  : ',F8.1,/
     6  '  MAXIMUM INVENTORY: ',F8.1,/
     7  '  STARTUP MAXIMUM  : ',F8.1,)
      IF(INDEX8.EQ.0)THEN
        WRITE(*,'(A)')'  LOADER TYPE USED : NONE'
      ELSE
```

```
        WRITE(*,3410)INDEX8
3410    FORMAT('  LOADER TYPE USED : ',I2,)
      ENDIF
      IF(INDEX7.EQ.0)THEN
        WRITE(*,'(A)')'  TIME DELAYS BY    : BUILD-IN FUNCTIONS'
      ELSE
        WRITE(*,'(A)')'  TIME DELAYS BY    : FORTRAN-USERFUNCTION'
      ENDIF
      WRITE(*,'(//,A,\)')'        INPUT DATA OK (Y/N)?    [Y]---->'
      READ(*,'(A1)')CHR1ANSWER
      IF(CHR1ANSWER.EQ.'N')THEN
        GOTO 3105
      ELSE
        PROCNAMES(INDEX1)=CHRANSWER
        USERARR(5,INDEX1)=INDEX9
        INDEX2=XXLEVEL(7)+INDEX1
        XX(INDEX2)=F1ANSWER
        INDEX2=XXLEVEL(3)+INDEX1
        XX(INDEX2)=F2ANSWER
        INDEX2=XXLEVEL(5)+INDEX1
        XX(INDEX2)=F3ANSWER
        INDEX2=XXLEVEL(4)+INDEX1
        XX(INDEX2)=F4ANSWER
        INDEX2=XXLEVEL(6)+INDEX1
        XX(INDEX2)=F5ANSWER
        USERARR(4,INDEX1)=INDEX8
        USERARR(9,INDEX1)=INDEX7
      ENDIF

C
3490  CONTINUE
C
3500  CONTINUE
C
C


C
C
C SPECIFYING THE RESOURCES USED
C -----------------------------
C
5000  WRITE(*,5010)
5010  FORMAT(////////////////////////////////,20X,
     1'FIFTH PHASE'/20X'==========='//,
     2' WE NOW SPECIFY THE RESOURCES E.G. MACHINES WE WANT TO USE IN'/,
     3' EACH PROCESS. FOR EACH ACTIVE PROCESS THE PROGRAM WILL GIVE'/,
     4' A CHOICE OF DIFFERENT MACHINE TYPES. YOU WILL HAVE TO SPECIFY'/,
     5' THE INITIAL NUMBER OF MACHINES FOR EACH TYPE. MULTIPLE TYPES'/,
     6' OF MACHINES WITH DIFFERENT INITIAL NUMBERS OF MACHINES PER '/,
     7' PROCESS ARE POSSIBLE.')
      WRITE(*,5020)
5020  FORMAT(
     1' HOWEVER, IF YOU HAVE SPECIFIED ANY PROCESSES USING LOADERS'/,
```

```
     2' THE PROGRAM WILL PROMPT YOU FIRST TO ENTER HOW MANY'/,
     3' MACHINES FOR EACH LOADER TYPE USED YOU WANT TO EMPLOY.'//,
     4' THE MAXIMUM NUMBER OF MACHINES WHICH THE NETWORK WILL'/,
     5' HANDEL IS APPROXIMATLY 90 MACHINES IN TOTAL.'///)
      WRITE(*,'(A\)') '       (PLEASE HIT RETURN TO CONTINUE)'
      READ (*,'(BZ,I6)') IANSWER
C
C
C
C SPECIFYING THE NUMBER OF LOADER USED:
C ------------------------------------
C
C

4000  INDEX1=0
      INDEX2=0
      INDEX3=0
      INDEX4=0
      INDEX5=0
      DO 4010 INDEX7=1,13,1
        IF (USERARR(4,INDEX7).EQ.32) THEN
          INDEX1=1
        ELSEIF (USERARR(4,INDEX7).EQ.33) THEN
          INDEX2=1
        ELSEIF (USERARR(4,INDEX7).EQ.34) THEN
          INDEX3=1
        ELSEIF (USERARR(4,INDEX7).EQ.35) THEN
          INDEX4=1
        ELSEIF (USERARR(4,INDEX7).EQ.36) THEN
          INDEX5=1
        ELSE
          CONTINUE
        ENDIF
4010  CONTINUE
C
      IF (INDEX1.EQ.0.AND.INDEX2.EQ.0.AND.INDEX3.EQ.0.AND.
     1    INDEX4.EQ.0.AND.INDEX5.EQ.0) GOTO 4900
      WRITE(*,4100)
4100  FORMAT(////////////////////////,
     1'  YOU HAVE TO SPECIFY THE NUMBER OF LOADERS',
     2' YOU WANT TO USE :'/,
     3'------------------------------------------',
     4'-------------------'/)
C
      DO 4150 INDEX7=32,36,1
        IF(INDEX7.EQ.32.AND.INDEX1.EQ.0) GOTO 4140
        IF(INDEX7.EQ.33.AND.INDEX2.EQ.0) GOTO 4140
        IF(INDEX7.EQ.34.AND.INDEX3.EQ.0) GOTO 4140
        IF(INDEX7.EQ.35.AND.INDEX4.EQ.0) GOTO 4140

        IF(INDEX7.EQ.36.AND.INDEX5.EQ.0) GOTO 4140
4110    WRITE(*,4120)INDEX7
```

```
4120    FORMAT(' HOW MANY LOADERS TYPE ',I2,' DO YOU WANT TO USE ?',
     1  '      -----> ',\)
        READ(*,'(I2)')IANSWER
        INDEX8=IANSWER
        IF (IANSWER.LE.0) THEN
          WRITE(*,4130)
4130      FORMAT(/,' !!!  CANNOT BE, PLEASE TRY AGAIN  !!!')
          GOTO 4110
        ELSE
          USERARR(6,INDEX7)=IANSWER
        ENDIF
4140  CONTINUE
4150  CONTINUE
C
C
      WRITE(*,4200)
4200  FORMAT(/////////////////////////////////////,
     1 '  LOADING DEVICES: ',/,
     2 '  ----------------'/)
C
        WRITE(*,4210)USERARR(6,32),USERARR(6,33),USERARR(6,34),
     1  USERARR(6,35),USERARR(6,36)
4210    FORMAT('  NUMBER OF LOADERS TYPE 32 USED          : 'F3.0,/
     1         '  NUMBER OF LOADERS TYPE 33 USED          : 'F3.0,/
     2         '  NUMBER OF LOADERS TYPE 34 USED          : 'F3.0,/
     3         '  NUMBER OF LOADERS TYPE 35 USED          : 'F3.0,/
     4         '  NUMBER OF LOADERS TYPE 36 USED          : 'F3.0)
        WRITE(*,'(//,A,\)')'       INPUT DATA OK (Y/N)?    [Y]---->'
        READ(*,'(A1)')CHRANSWER
        IF(CHRANSWER.EQ.'N')GOTO 4000
C
4900  CONTINUE
C
C
C
CC
C
      INDEX9=0
      DO 5900 INDEX1=1,13,1
C
5050    IF(USERARR(23,INDEX1).EQ.0.AND.USERARR(24,INDEX1).EQ.0)GOTO 5890
C
        IF (INDEX1.EQ.1) THEN
5100      WRITE (*,5110)INDEX1,PROCNAMES(INDEX1)
5110      FORMAT(//////////////////////,3X,'PROCESS NO.'I2,':  ',A,/
     1    3X,'----------------',/
     2    ' THERE ARE FOUR (4) DIFFERENT MACHINE TYPES POSSIBLE:',/)
          DO 5150 INDEX2=1,4,1
5120      WRITE(*,5130)INDEX2
5130      FORMAT(' MACHINE TYPE ',I2,': INITIAL # OF MACHINES ?'
     1    '    [0]-----> ',\)
          READ(*,'(I2)')IANSWER
```

```
          IF(IANSWER.LT.Ø.OR.IANSWER.GT.8Ø) THEN
            WRITE(*,514Ø)
514Ø        FORMAT(/,' !!!  CANNOT BE, PLEASE TRY AGAIN  !!!')
            GOTO 512Ø
          ELSE
            USERARR(6,INDEX2)=IANSWER
            INDEX9=INDEX9+IANSWER
          ENDIF
515Ø      CONTINUE
          IF(USERARR(6,1).EQ.Ø.AND.USERARR(6,2).EQ.Ø.AND.USERARR(6,3).EQ.
     1    Ø.AND.USERARR(6,4).EQ.Ø) THEN
            WRITE (*,516Ø)
516Ø        FORMAT(//,' !!!! YOU HAVE NOT ACTIVATED ANY MACHINE ',
     1      'IN PROCESS 1  !!!!',/
     2      ' !!!!'15X'THIS CANNOT BE, PLEASE TRY AGAIN'15X'!!!!',/)
            GOTO 51ØØ
         ENDIF
C
       ELSEIF (INDEX1.GE.2.AND.INDEX1.LE.1Ø) THEN
52ØØ     WRITE (*,521Ø)INDEX1,PROCNAMES(INDEX1)
521Ø     FORMAT(///////////////////////,3X,'PROCESS NO.'I2,':  ',A,/
     1   3X,'---------------',/
     2   ' THERE ARE THREE (3) DIFFERENT MACHINE TYPES POSSIBLE:',/)
         DO 525Ø INDEX2=1,3,1
           INDEX3=INDEX1*3-2+INDEX2
522Ø       WRITE(*,523Ø)INDEX3
523Ø       FORMAT(' MACHINE TYPE ',I2,': INITIAL # OF MACHINES ?'
     1     '    [Ø]-----> ',\)
           READ(*,'(I2)')IANSWER
           INDEX8=INDEX9+IANSWER
           IF(IANSWER.LT.Ø.OR.INDEX8.GT.8Ø) THEN
             WRITE(*,524Ø)
524Ø         FORMAT(/,' !!!  CANNOT BE, PLEASE TRY AGAIN  !!!')
             GOTO 522Ø
           ELSE
            USERARR(6,INDEX3)=IANSWER
            INDEX9=INDEX9+IANSWER
           ENDIF
525Ø     CONTINUE
         INDEX4=INDEX1*3-1
         INDEX5=INDEX1*3
         INDEX6=INDEX1*3+1
         IF(USERARR(6,INDEX4).EQ.Ø.AND.USERARR(6,INDEX5).EQ.Ø.AND.
     1   USERARR(6,INDEX6).EQ.Ø) THEN
           WRITE (*,526Ø)INDEX1
526Ø       FORMAT(//,' !!!! YOU HAVE NOT ACTIVATED ANY MACHINE ',
     1     'IN PROCESS ',I2,'  !!!!',/
     2     ' !!!!'15X'THIS CANNOT BE, PLEASE TRY AGAIN'15X'!!!!',/)
           GOTO 52ØØ
         ENDIF
C
       ELSEIF (INDEX1.EQ.11) THEN
53ØØ     WRITE (*,531Ø)INDEX1,PROCNAMES(INDEX1)
```

```
5310     FORMAT(///////////////////////,3X,'PROCESS NO.'I2,':  ',A,/
     1   3X,'---------------',/
     2   ' FOR THIS PROCESS YOU HAVE TO SPECIFY HOW MUCH OF THE'/
     3   ' INCOMING INVENTORY WILL BE ROUTED TO THE TWO FOLLOWING'/
     4   ' PROCESSES',/)
5320     WRITE(*,5330)
5330     FORMAT(' HOW MUCH INVENTORY IN % GOES ROUTE 1 ?   %%%.%%----->'
     1   ' ',\)
        READ(*,'(F6.2)')F1ANSWER
        WRITE(*,5340)
5340     FORMAT(' HOW MUCH INVENTORY IN % GOES ROUTE 2 ?   %%%.%%----->'
     1  ' ',\)
        READ(*,'(F6.2)')F2ANSWER
        F3ANSWER=F1ANSWER+F2ANSWER
        IF (F3ANSWER.NE.100) THEN
          WRITE(*,5350)
5350      FORMAT(/,' !!!  CANNOT BE, PLEASE TRY AGAIN  !!!')
          GOTO 5320
        ELSE
          USERARR(7,3)=F1ANSWER
          USERARR(7,4)=F2ANSWER
        ENDIF
C
       ELSEIF (INDEX1.EQ.12) THEN
5400     WRITE (*,5410)INDEX1,PROCNAMES(INDEX1)
5410     FORMAT(////////////////////////,3X,'PROCESS NO.'I2,':  ',A,/
     1   3X,'---------------',/
     2   ' FOR THIS PROCESS YOU CAN SPECIFY A PRIMARY TRANSPORTAION',/
     3   ' DEVICE AND A SCONDARY ONE.',/
     3   ' THE MACHINE WHICH REQUIRES THESE TRANSPORTATION DEVICES',/
     4   ' (EX.: CHIPPER) IS AUTOMATICALLY INVOKED.'/)
5420     WRITE(*,5430)
5430     FORMAT(' HOW MANY PRIMARY TRANSPORTERS DO YOU WANT TO USE   ?'
     1   '        -----> ',\)
         READ(*,'(I2)')IANSWER
         INDEX8=INDEX9+IANSWER
         IF (IANSWER.EQ.0.OR.INDEX8.GT.80) THEN
           WRITE(*,5440)
5440       FORMAT(/,' !!!  CANNOT BE, PLEASE TRY AGAIN  !!!')
           GOTO 5420
        ELSE
           INDEX9=INDEX9+IANSWER
           USERARR(6,39)=IANSWER
           USERARR(6,37)=1
        ENDIF
5450    WRITE(*,5460)
5460    FORMAT(' HOW MANY SECONDARY TRANSPORTERS DO YOU WANT TO USE  ?'
     1  '       [0]-----> ',\)
        READ(*,'(I2)')IANSWER
        INDEX8=INDEX9+IANSWER
        IF (INDEX8.GT.80) THEN
          WRITE(*,5470)
```

```
5470      FORMAT(/,' !!!  CANNOT BE, PLEASE TRY AGAIN  !!!')
          GOTO 5450
        ELSE
          INDEX9=INDEX9+IANSWER
          USERARR(6,40)=IANSWER
        ENDIF
C
       ELSE
5500     WRITE (*,5510)INDEX1,PROCNAMES(INDEX1)
5510     FORMAT(////////////////////////////,3X,'PROCESS NO.'I2,':  ',A,/
     1   3X,'---------------'/,
     2   ' FOR THIS PROCESS YOU CAN SPECIFY A PRIMARY AND A SECONDARY',/
     3   ' TRANSPORTING DEVICE.',/)
5520     WRITE(*,5530)
5530     FORMAT(' HOW MANY PRIMARY TRANSPORTERS DO YOU WANT TO USE',
     1   ' ?        -----> ',\)
         READ(*,'(I2)')IANSWER
         INDEX8=INDEX9+IANSWER
         IF (IANSWER.EQ.0.OR.INDEX8.GT.80) THEN
           WRITE(*,5540)
5540       FORMAT(/,' !!!  CANNOT BE, PLEASE TRY AGAIN  !!!')
           GOTO 5520
         ELSE
           INDEX9=INDEX9+IANSWER
           USERARR(6,41)=IANSWER
         ENDIF
5550     WRITE(*,5560)
5560     FORMAT(' HOW MANY SECONDARY TRANSPORTERS DO YOU WANT TO USE',
     1   ' ?  [0]-----> ',\)
         READ(*,'(I2)')IANSWER
         INDEX8=INDEX9+IANSWER
         IF (INDEX8.GT.80) THEN
            WRITE(*,5570)
5570        FORMAT(/,' !!!  CANNOT BE, PLEASE TRY AGAIN  !!!')
            GOTO 5550
         ELSE
            INDEX9=INDEX9+IANSWER
            USERARR(6,42)=IANSWER
         ENDIF
C
       ENDIF
C
C
C
C
      WRITE(*,5600)INDEX1,PROCNAMES(INDEX1)
5600  FORMAT(//////////////////////////////////////,
     1 '  PROCESS NO. 'I2' : 'A,/
     2 '  ----------------'/)
C
      IF (INDEX1.EQ.1) THEN
        DO 5620 INDEX2=1,4,1
          WRITE(*,5610)INDEX2,USERARR(6,INDEX2)
```

```
5610      FORMAT('  MACHINE TYPE 'I2', # OF INITIAL MACHINES : ',F3.0)
5620     CONTINUE
C
      ELSEIF (INDEX1.GE.2.AND.INDEX1.LE.10) THEN
        DO 5640 INDEX2=1,3,1
          INDEX3=INDEX1*3-2+INDEX2
          WRITE(*,5630)INDEX3,USERARR(6,INDEX3)
5630      FORMAT('  MACHINE TYPE 'I2', # OF INITIAL MACHINES : ',F3.0)
5640     CONTINUE
C
      ELSEIF (INDEX1.EQ.11) THEN
       WRITE(*,5650)USERARR(7,3),USERARR(7,4)
5650   FORMAT('  % OF INVENTORY GOING ROUTE 1            : 'F6.2' %'/
     1        '  % OF INVENTORY GOING ROUTE 2            : 'F6.2' %')
C
      ELSEIF (INDEX1.EQ.12) THEN
       WRITE(*,5660)USERARR(6,39),USERARR(6,40)
5660   FORMAT('  NUMBER OF PRIMARY TRANSP. DEVICES USED   :  'F3.0,/
     1        '  NUMBER OF SECONDARY TRANSP. DEVICES USED :  'F3.0)
C
      ELSEIF (INDEX1.EQ.13) THEN
       WRITE(*,5670)USERARR(6,41),USERARR(6,42)
5670   FORMAT('  NUMBER OF PRIMARY TRANSP.DEVICES USED    : 'F3.0,/
     1        '  NUMBER OF SECONDARY TRANSP.DEVICES USED  : 'F3.0)
C
      ELSE
       CONTINUE
      ENDIF
C
C
       WRITE(*,'(//,A,\)')'       INPUT DATA OK (Y/N)?    [Y]---->'
       READ(*,'(A1)')CHRANSWER
       IF(CHRANSWER.EQ.'N')GOTO 5050
C
5800   CONTINUE
C
C
5890  CONTINUE
5900  CONTINUE
C
C
C
C
C
C  SPECIFYING THE MACHINE PARAMETERS
C  =================================================================
C
6000  WRITE(*,6010)
6010  FORMAT(///////////////////////////////,20X,
     1'SIXTH PHASE'/20X'==========='//,
     2' HERE WE SPECIFY ALL THE PARAMETERS RELATED TO THE MACHINE '/,
     3' TYPES YOU HAVE SET ACTIVE EARLIER:',/,
     4' - NAME OF MACHINE',/,
```

```
     5' - AVERAGE PROCESSING TIME PER TREE',/,
     6' - FIXED CONSTANT TIME PER LOAD',/,
     7' - FIXED CONSTANT TIME PER ONE WAY HAUL')
      WRITE(*,6020)
6020  FORMAT(
     1' - MACHINE CAPACITY IN CU.FT.',/,
     2' - FIXED COST PER SCHEDULED HOUR',/,
     3' - VARIABLE COST PER MACHINE HOUR',//)
      WRITE(*,'(A\)') '         (PLEASE HIT RETURN TO CONTINUE)'
      READ (*,'(BZ,I6)') IANSWER
C
C
C
C
      DO 6500 INDEX4=1,42,1
        IF(USERARR(6,INDEX4).EQ.0) GOTO 6490
C
        IF (INDEX4.GE.1.AND.INDEX4.LE.4) THEN
          INDEX1=1
        ELSEIF (INDEX4.GE.5.AND.INDEX4.LE.7) THEN
          INDEX1=2
        ELSEIF (INDEX4.GE.8.AND.INDEX4.LE.10) THEN
          INDEX1=3
        ELSEIF (INDEX4.GE.11.AND.INDEX4.LE.13) THEN
          INDEX1=4
        ELSEIF (INDEX4.GE.14.AND.INDEX4.LE.16) THEN
          INDEX1=5
        ELSEIF (INDEX4.GE.17.AND.INDEX4.LE.19) THEN
          INDEX1=6
        ELSEIF (INDEX4.GE.20.AND.INDEX4.LE.22) THEN
          INDEX1=7
        ELSEIF (INDEX4.GE.23.AND.INDEX4.LE.25) THEN
          INDEX1=8
        ELSEIF (INDEX4.GE.26.AND.INDEX4.LE.28) THEN
          INDEX1=9
        ELSEIF (INDEX4.GE.29.AND.INDEX4.LE.31) THEN
          INDEX1=10
        ELSEIF (INDEX4.GE.32.AND.INDEX4.LE.36) THEN
          INDEX1=14
        ELSEIF (INDEX4.GE.37.AND.INDEX4.LE.40) THEN
          INDEX1=12
        ELSEIF (INDEX4.GE.41.AND.INDEX4.LE.42) THEN
          INDEX1=13
        ELSE
          CONTINUE
        ENDIF
C
6100   IF (INDEX1.NE.14) THEN
         WRITE (*,6110)INDEX1,PROCNAMES(INDEX1)
6110     FORMAT(///////////////////////////,
     1   ' PROCESS NO.'I2,':  ',A,/
     2   ' ==============',)
```

```
      ELSE
       WRITE (*,612Ø)
612Ø   FORMAT(///////////////////////////,
     1  ' MACHINE PARAMETERS FOR THE LOADING DEVICE :',/
     2  ' ==============================================',)
      ENDIF
C
       WRITE(*,616Ø)INDEX4
616Ø   FORMAT(//,' MACHINE TYPE 'I2' :',/,' ---------------')
       WRITE(*,'(A,\)')' NAME OF MACHINE TYPE ?             ------> '
       READ(*,'(A2Ø)')CHRANSWER
       MCHNAMES(INDEX4)=CHRANSWER
       IF(INDEX4.EQ.4Ø.OR.INDEX4.EQ.42) GOTO 622Ø
       IF(INDEX4.EQ.39) GOTO 618Ø
617Ø   WRITE(*,'(A,\)')' AVERAGE PROCESSING TIME / TREE?   [Ø]------> '
       READ(*,'(F8.4)')F1ANSWER
       IF (F1ANSWER.LT.Ø) THEN
         WRITE(*,'(/,A)')' !!!  CANNOT BE, PLEASE TRY AGAIN  !!!'
         GOTO 617Ø
       ELSE
         USERARR(1,INDEX4)=F1ANSWER
       ENDIF
618Ø   WRITE(*,'(A,\)')' FIXED CONSTANT TIME / LOAD?       [Ø]------> '
       READ(*,'(F8.4)')F1ANSWER
       IF (F1ANSWER.LT.Ø) THEN
         WRITE(*,'(/,A)')' !!!  CANNOT BE, PLEASE TRY AGAIN  !!!'
         GOTO 618Ø
       ELSE
         USERARR(2,INDEX4)=F1ANSWER
       ENDIF
619Ø   WRITE(*,'(A,\)')' FIXED CONST. TIME / ONE WAY HAUL? [Ø]------> '
       READ(*,'(F8.4)')F1ANSWER
       IF (F1ANSWER.LT.Ø) THEN
         WRITE(*,'(/,A)')' !!!  CANNOT BE, PLEASE TRY AGAIN  !!!'
         GOTO 619Ø
       ELSE
         USERARR(3,INDEX4)=F1ANSWER
       ENDIF
62ØØ   WRITE(*,'(A,\)')' MACHINE CAPACITY IN CU.FT?        [1]------> '
       READ(*,'(F8.2)')F1ANSWER
       IF (F1ANSWER.LT.Ø) THEN
         WRITE(*,'(/,A)')' !!!  CANNOT BE, PLEASE TRY AGAIN  !!!'
         GOTO 62ØØ
       ELSEIF (F1ANSWER.GT.99999) THEN
         WRITE(*,'(/,A)')' !!!  CANNOT BE, PLEASE TRY AGAIN  !!!'
         GOTO 62ØØ
       ELSEIF(F1ANSWER.EQ.Ø) THEN
         USERARR(8,INDEX4)=1
       ELSE
         USERARR(8,INDEX4)=F1ANSWER
       ENDIF
```

```
6220   WRITE(*,'(A,\)')' FIXED COST / SCHEDULED HOUR?      [0]------> '
       READ(*,'(F8.2)')F1ANSWER
       IF (F1ANSWER.LT.0) THEN
         WRITE(*,'(/,A)')' !!!  CANNOT BE, PLEASE TRY AGAIN  !!!'
         GOTO 6220
       ELSE
         USERARR(21,INDEX4)=F1ANSWER
       ENDIF
6230   WRITE(*,'(A,\)')' VARIABLE COST/ MACHINE HOUR?      [0]------> '
       READ(*,'(F8.2)')F1ANSWER
       IF (F1ANSWER.LT.0) THEN
         WRITE(*,'(/,A)')' !!!  CANNOT BE, PLEASE TRY AGAIN  !!!'
         GOTO 6230
       ELSE
         USERARR(22,INDEX4)=F1ANSWER
       ENDIF
C
C
       IF (INDEX1.NE.14) THEN
         WRITE (*,6240)INDEX1,PROCNAMES(INDEX1)
6240     FORMAT(////////////////////////,' PROCESS NO. 'I2,' : ',A,/
     1   ' ================',)
       ELSE
         WRITE (*,6250)
6250     FORMAT(////////////////////////,' LOADING DEVICE ',/
     1   ' ==============',)
       ENDIF
       WRITE(*,6260)INDEX4
6260   FORMAT(,' MACHINE TYPE 'I2,/,
     1        ' ---------------',/)
       WRITE(*,6270)MCHNAMES(INDEX4)
6270   FORMAT(' NAME OF MACHINE TYPE                : ',A)
       IF(INDEX4.EQ.40.OR.INDEX4.EQ.42)GOTO 6310
       IF (INDEX4.EQ.39) GOTO 6285
       WRITE(*,6280)USERARR(1,INDEX4)
6280   FORMAT(' AVERAGE PROCESSING TIME / TREE    :    ',F8.4)
6285   WRITE(*,6290)USERARR(2,INDEX4)
6290   FORMAT(' FIXED CONSTANT TIME / LOAD        :    ',F8.4)
       WRITE(*,6300)USERARR(3,INDEX4)
6300   FORMAT(' FIXED CONST. TIME / ONE WAY HAUL  :    ',F8.4)
       WRITE(*,6340)USERARR(8,INDEX4)
6340   FORMAT(' MACHINE CAPACITY IN CU.FT         : ',F8.2)
6310   WRITE(*,6320)USERARR(21,INDEX4)
6320   FORMAT(' FIXED COST / SCHEDULED HOUR       : ',F8.2)
       WRITE(*,6330)USERARR(22,INDEX4)
6330   FORMAT(' VARIABLE COST/ MACHINE HOUR       : ',F8.2)
       WRITE(*,'(//,A,\)')'       INPUT DATA OK (Y/N)?    [Y]---->'
       READ(*,'(A1)')CHRANSWER
       IF(CHRANSWER.EQ.'N')GOTO 6100
C
C
C
C
```

```
6490    CONTINUE
6500   CONTINUE
C
C
C
C
C
C  DEFINING THE MACHINE BREAKDOWN PARAMETERS
C  ========================================
C
C
C
7000  WRITE(*,7020)
7020  FORMAT(////////////////////////////////,20X,
     1'SEVENTH PHASE'/20X'===============''//,
     2' IN THIS LAST PHASE YOU ARE ABLE TO SPECIFY THE MACHINE '/,
     3' BREAKDOWN PARAMETERS FOR EACH ACTIVE MACHINE. IN ORDER',/,
     4' TO DO SO YOU WILL HAVE TO INPUT THE CUMULATIVE FREQUENCY',/,
     5' DISTRIBUTION FOR THE TIME BETWEEN FAILURES AND THE ',/,
     6' ACTUAL REPAIR TIME. EACH OF THESE TWO DISTRIBUTIONS CAN ',/,
     7' HAVE UP TO TEN CLASSES.')
      WRITE(*,7040)
7040  FORMAT(
     1'  !!!   DONT FORGET THE DECIMAL POINT FOR INPUT  !!!',///)
      WRITE(*,'(A\)') '        (PLEASE HIT RETURN TO CONTINUE)'
      READ (*,'(BZ,I6)') IANSWER
C
C
      DO 7500 INDEX1=1,42,1
       IF (USERARR(6,INDEX1).EQ.0) GOTO 7490
7050   F3ANSWER=0
C
       WRITE(*,7060)INDEX1,MCHNAMES(INDEX1)
7060   FORMAT(//////////////////////////////,
     1 '  MACHINE TYPE ',I2,' : ',A,/
     2 '  ================',//,
     3 '  FREQUENCY DISTRIBUTION FOR TIMES BETWEEN FAILURES:'/,
     4 '  ------------------------------------------------',/)
7080    DO 7200INDEX2=1,10,1
7100      WRITE(*,7120)INDEX2
7120      FORMAT(/,' CLASS ',I2,': CUM.REL.FREQENCY?      [0]----->'\)
          READ(*,'(BN,F8.2)')F1ANSWER
          WRITE(*,7140)INDEX2
7140      FORMAT(' CLASS ',I2,': TIME BETWEEN FAILURES?  [0]----->'\)
          READ(*,'(BN,F8.2)')F2ANSWER
            IF(INDEX2.EQ.1.AND.F1ANSWER.EQ.0)THEN
             GOTO 7490
            ELSEIF(F2ANSWER.LE.0)THEN
             WRITE(*,'(/,A)')' !!  CANNOT BE, PLEASE TRY AGAIN  !!'
             GOTO 7100
            ELSEIF(F1ANSWER.GT.100)THEN
             WRITE(*,'(/,A)')' !!  CANNOT BE, PLEASE TRY AGAIN  !!'
             GOTO 7100
```

```
          ELSEIF(INDEX2.GT.1.AND.F1ANSWER.LE.F3ANSWER)THEN
           WRITE(*,'(/,A)')' !!  CANNOT BE, PLEASE TRY AGAIN  !!'
           GOTO 7100
          ELSEIF(INDEX2.EQ.10.AND.F1ANSWER.NE.100)THEN
           WRITE(*,'(/,A)')' !!  CANNOT BE, PLEASE TRY AGAIN  !!'
           GOTO 7100
          ELSE
           F3ANSWER=F1ANSWER
           MCHARR(INDEX1,1,INDEX2)=F1ANSWER
           MCHARR(INDEX1,2,INDEX2)=F2ANSWER
           IF(F1ANSWER.EQ.100)GOTO 7210
         ENDIF
7190    CONTINUE
7200   CONTINUE
C
C
7210   F3ANSWER=0
       WRITE(*,7220)INDEX1,MCHNAMES(INDEX1)
7220   FORMAT(////,
     1 '  MACHINE TYPE 'I2' : 'A,/
     2 '  ==================',//
     3 '  FREQUENCY DISTRIBUTION FOR MACHINE REPAIR TIMES:'/,
     4 '  ---------------------------------------------'/)
7240    DO 7400 INDEX2=1,10,1
7260      WRITE(*,7280)INDEX2
7280       FORMAT(/,' CLASS ',I2,': CUM.REL.FREQENCY?       [0]----->'\)
          READ(*,'(BN,F8.2)')F1ANSWER
          WRITE(*,7300)INDEX2
7300      FORMAT(' CLASS ',I2,': REPAIR TIME?          [0]----->'\)
          READ(*,'(BN,F8.2)')F2ANSWER
            IF(INDEX2.EQ.1.AND.F1ANSWER.EQ.0)THEN
             WRITE(*,'(/,A)')' !!  CANNOT BE, PLEASE TRY AGAIN  !!'
             GOTO 7260
            ELSEIF(F2ANSWER.LE.0)THEN
             WRITE(*,'(/,A)')' !!  CANNOT BE, PLEASE TRY AGAIN  !!'
             GOTO 7260
            ELSEIF(F1ANSWER.GT.100)THEN
             WRITE(*,'(/,A)')' !!  CANNOT BE, PLEASE TRY AGAIN  !!'
             GOTO 7260
            ELSEIF(INDEX2.GT.1.AND.F1ANSWER.LE.F3ANSWER)THEN
             WRITE(*,'(/,A)')' !!  CANNOT BE, PLEASE TRY AGAIN  !!'
             GOTO 7260
            ELSEIF(INDEX2.EQ.10.AND.F1ANSWER.NE.100)THEN
             WRITE(*,'(/,A)')' !!  CANNOT BE, PLEASE TRY AGAIN  !!'
             GOTO 7260
            ELSE
             F3ANSWER=F1ANSWER
             MCHARR(INDEX1,3,INDEX2)=F1ANSWER
             MCHARR(INDEX1,4,INDEX2)=F2ANSWER
             IF(F1ANSWER.EQ.100)GOTO 7490
            ENDIF
7390     CONTINUE
```

```
7400   CONTINUE
C
C
7490   CONTINUE
C
C
      IF (USERARR(6,INDEX1).EQ.0) GOTO 7790
7520  WRITE(*,7540)INDEX1,MCHNAMES(INDEX1)
7540  FORMAT(///////////////////////////////,
     1 '  FREQUENCY DISTRIBUTIONS MACHINE TYPE 'I2' : 'A,/,
     2 '  ========================================',//
     3 '   CLASS     CUM FREQ.%      TIME BETW.FAILURE      CUM.FREQ.%',
     4 '      REPAIR TIME'/,
     5 '  ------------------------------------------------------------',
     6 '------------------')
       IF(MCHARR(INDEX1,1,1).EQ.0.)THEN
         WRITE(*,'(//,20X,A)')'******  DISTRIBUTION NOT USED  ********'
       ELSE
         DO 7600 INDEX2=1,10,1
           IF(MCHARR(INDEX1,1,INDEX2).EQ.0.AND.
     1       MCHARR(INDEX1,3,INDEX2).EQ.0) THEN
             GOTO 7590
           ELSEIF (MCHARR(INDEX1,1,INDEX2).GT.0.AND.
     1       MCHARR(INDEX1,3,INDEX2).EQ.0) THEN
             WRITE(*,7560)INDEX2,MCHARR(INDEX1,1,INDEX2),
     1       MCHARR(INDEX1,2,INDEX2)
7560         FORMAT(4X,I2,10X,F8.2,10X,F8.2)
           ELSEIF (MCHARR(INDEX1,1,INDEX2).EQ.0.AND.
     1       MCHARR(INDEX1,3,INDEX2).GT.0) THEN
             WRITE(*,7570)INDEX2,MCHARR(INDEX1,3,INDEX2),
     1       MCHARR(INDEX1,4,INDEX2)
7570         FORMAT(4X,I2,10X,8X,10X,8X,10X,F8.2,10X,F8.2)
           ELSE
             WRITE(*,7580)INDEX2,MCHARR(INDEX1,1,INDEX2),
     1       MCHARR(INDEX1,2,INDEX2),MCHARR(INDEX1,3,INDEX2),
     2       MCHARR(INDEX1,4,INDEX2)
7580         FORMAT(4X,I2,10X,F8.2,10X,F8.2,10X,F8.2,10X,F8.2)
           ENDIF
7590       CONTINUE
7600     CONTINUE
       ENDIF
       WRITE(*,'(//,A,\)')'  DISTRIBUTION OK (Y/N) ?  [Y]---->'
       READ(*,'(A1)')CHRANSWER
       IF(CHRANSWER.EQ.'N')THEN
         DO 7602 INDEX2=1,4,1
           DO 7601 INDEX3=1,10,1
             MCHARR(INDEX1,INDEX2,INDEX3)=0
7601       CONTINUE
7602     CONTINUE
         GOTO 7050
       ENDIF
C
```

```
7790     CONTINUE
C
C
C
7500  CONTINUE
C
C
C
C
C   SAVING MODEL ON DISK
C   ===================
C
C
8000   WRITE(*,8002)
8002   FORMAT(///////////////////////////////,20X,
     1'END OF SUBROUTINE READIN'/20X
     2'========================'//,
     3'  YOU HAVE NOW DEFINED A MODEL FOR THE MECHANIZED LOG '/,
     4'  HARVESTING SIMULATOR. DO YOU WANT TO SAVE THIS MODEL ON'/,
     5'  DISK? IF YOU DONT DO SO ALL YOUR WORK WILL BE LOST !!'//)
C
8004  WRITE(*,'(/A\)')'  SAVE MODEL ON DISK Y/N ?      [Y]------> '
      READ(*,'(A1)')CHRANSWER
C
      IF (CHRANSWER.EQ.'N') THEN
        WRITE(*,'(/,A,\)')'  ARE YOU REALY SURE Y/N ?      [N]------> '
        READ(*,'(A1)')CHRANSWER
        IF (CHRANSWER.EQ.'Y') THEN
          GOTO 9998
        ELSE
          GOTO 8004
        ENDIF
      ELSE
        CONTINUE
      ENDIF
C
8010  INQUIRE(FILE=FILENAME,EXIST=FILESTATUS)
      IF(.NOT.FILESTATUS) THEN
        OPEN(10,FILE=FILENAME,STATUS='NEW')
      ELSE
        WRITE(*,8022)FILENAME
8022    FORMAT(/,'  !!!!  FILE: 'A' ALREADY EXISTS !!!!'/
     1  '  OVERWRITE OLD FILE?            [N]-----> ',\)
        READ(*,'(A1)')CHRANSWR
        IF(CHRANSWER.EQ.'Y')THEN
          OPEN(10,FILE=FILENAME,STATUS='OLD')
          REWIND 10
        ELSE
          WRITE(*,'(A\)')'  INPUT NEW FILENAME FOR MODEL:       -----> '
          READ(*,'(A20)')FILENAME
          GOTO 8010
        ENDIF
      ENDIF
```

```
C
C
      WRITE(10,'(F8.1)') XX(1)
      WRITE(10,'(F8.1)') XX(2)
      WRITE(10,'(F8.1)') XX(3)
      WRITE(10,'(F8.0)') XX(4)
      WRITE(10,'(F8.1)') XX(5)
      WRITE(10,'(F8.1)') XX(6)
      WRITE(10,'(F8.1)') XX(7)
      WRITE(10,'(F8.1)') XX(8)
      WRITE(10,'(F8.1)') XX(9)
      WRITE(10,'(F8.4)') XX(10)
      DO 8023 INDEX1=11,100,1
        WRITE(10,8020) XX(INDEX1)
8020    FORMAT(F8.1)
8023  CONTINUE
C
C
      DO 8028 INDEX1=1,3,1
      DO 8026 INDEX2=1,42,1
        WRITE(10,8024) USERARR(INDEX1,INDEX2)
8024    FORMAT(F8.4)
8026  CONTINUE
8028  CONTINUE
C
      DO 8035 INDEX1=4,26,1
      DO 8034 INDEX2=1,42,1
        WRITE(10,8033) USERARR(INDEX1,INDEX2)
8033    FORMAT(F8.2)
8034  CONTINUE
8035  CONTINUE
C
C
C
C
      DO 8044 INDEX1=1,8,1
      DO 8042 INDEX2=1,10,1
        WRITE(10,8040) DISARR(INDEX1,INDEX2)
8040    FORMAT(F8.2)
8042   CONTINUE
8044   CONTINUE

C
      DO 8052 INDEX1=1,42,1
      DO 8050 INDEX2=1,4,1
      DO 8048 INDEX3=1,10,1
        WRITE(10,8046) MCHARR(INDEX1,INDEX2,INDEX3)
8046    FORMAT(F8.2)
8048   CONTINUE
8050   CONTINUE
8052   CONTINUE
```

```
C
      DO 8Ø56 INDEX1=1,52,1
        WRITE(1Ø,8Ø54) MCHNAMES(INDEX1)
8Ø54    FORMAT(A)
8Ø56   CONTINUE
C
      DO 8Ø6Ø INDEX1=1,2Ø,1
        WRITE(1Ø,8Ø58) PROCNAMES(INDEX1)
8Ø58    FORMAT(A)
8Ø6Ø  CONTINUE
C
      DO 8Ø64 INDEX1=1,4,1
        WRITE(1Ø,8Ø62) DISTRIBNAMES(INDEX1)
8Ø62    FORMAT(A)
8Ø64  CONTINUE
C
      REWIND 1Ø
      CLOSE(1Ø,STATUS='KEEP')
      WRITE(*,8Ø66)
8Ø66  FORMAT(///,2ØX,'!!!!  MODEL HAS BEEN SAVED  !!!!'/,
     12ØX,'    PRESS RETURN TO CONTINUE'\)
      READ(*,'(I2)')IANSWER
C
C
C
C END OF SUBROUTINE:
C ================
C
9998  RETURN
      END
```

APPENDIX D

4. Listing, PRINTOUT.FOR

```
C*************************************************************
C*                                                           *
C*                  OREGON STATE UNIVERSITY                  *
C*                         JUNE 1986                         *
C*                                                           *
C*                  >>>  L O G S I M <<<                     *
C*                                                           *
C*     SIMULATION OF MECHANIZED LOG HARVESTING SYSTEMS       *
C*                                                           *
C*                                                           *
C*     DESIGNED BY : CHRISTOPH WIESE                         *
C*                   MASTERS CANDIDATE, DEP. OF INDUSTRIAL   *
C*                   ENGINEERING, OREGON STATE UNIVERSITY    *
C*                                                           *
C*     DESIGNED FOR: DEPARTMENT OF FOREST ENGINEERING        *
C*                   OREGON STATE UNIVERSITY                 *
C*                                                           *
C*                                                           *
C*     SUPERVISION : DR. ELDON OLSEN                         *
C*                   ASSOCIATE PROFESSOR, DEP. OF INDUSTRIAL *
C*                   ENGINEERING, OREGON STATE UNIVERSITY    *
C*                                                           *
C*                                                           *
C*************************************************************
C*                                                           *
C*            FORTRAN INPUT USR-INTERFACE: MODELPRINT.FOR    *
C*                                                           *
C*                     31-MAY-87  18:55                      *
C*                                                           *
C*************************************************************
C
C
C
C
C
$INCLUDE:'PRCTL.FOR'
C
C PROGRAM DECLARATION:
C ===================
C
      SUBROUTINE MODELPRINT
C
C COMMON BLOCK :
C =============
C
$INCLUDE:'VARBLOCK.DOC'
C
C DEFINE LOCAL VARIABLES, NAMES & TYPE:
C ===================================
C
```

```
      INTEGER*4 IANSWER,INDEX1,INDEX2,INDEX3,INDEX4,INDEX5
      INTEGER*4 INDEX6,INDEX7,INDEX8,INDEX9
      CHARACTER*20 CHRANSWER
      REAL F1ANSWER,F2ANSWER,F3ANSWER
      LOGICAL*4 FILESTATUS
C
C BEGIN PROCESSING:
C =================
C
C OPENING SCREEN:
C ---------------
C
      WRITE(*,100)
100   FORMAT('1'////////////////////////////////////5X,
     1'                 SUBROUTINE PRINT'/5X,
     2'                 ================'//5X,
     3'WITH THIS SUBROUTINE YOU CAN PRINTOUT THE DATA OF A '/5X,
     4'SIMULATION MODEL PREVIOUSLY DEFINED WITH SUBROUTINE '/5X,
     5'READIN.',///)
      WRITE(*,'(7X,A\)') 'DO YOU WISH TO CONTINUE (Y/N) ?  [Y]-----> '
      READ (*,'(BZ,A1)') CHRANSWER
      IF (CHRANSWER.EQ.'N') THEN
        GOTO 9998
      ELSE
        CONTINUE
      ENDIF
C
C
C
C INITIALIZATION OF ALL VARIABLES AN READIN THE FILE
C ==================================================
C
C
      CALL INITREAD
      WRITE(*,244)
244   FORMAT(///,
     1 7X,'OUTPUT SHOULD BE ROUTED TO:'//
     2 7X,' SCREEN                  = 1',/
     3 7X,' SCREEN & PRINTER        = 2',//
     4 7X,'PLEASE ENTER CHOICE -----> ',\)
245   READ(*,'(I2)')OUTFLAG
      IF (OUTFLAG.LT.1.OR.OUTFLAG.GT.2) THEN
        WRITE(*,247)
247     FORMAT(/7X,'!!!  CANNOT BE, PLEASE ENTER AGAIN  !!!',/
     1            'PLEASE ENTER CHOICE -----> ',\)
        GOTO 245
      ELSEIF(OUTFLAG.EQ.1) THEN
        CONTINUE
      ELSE
        OPEN(11,FILE='LPT1')
        WRITE(*,250)
```

```
250     FORMAT(//7X,
     1  'ALIGN PAPER IN PRINTER, THEN PRESS RETURN TO CONTINUE'\)
        READ(*,'(I2)')IANSWER
      ENDIF
C
C
C
C
C BEGIN OF PRINTING THE MODEL
C =========================
C
C PRINTING GENERAL PARAMETERS AND MACHINE CONFIGURATION
C ----------------------------------------------------      IF (ATRIB(5).EQ.1) THEN
        WRITE(*,290)
        IF (OUTFLAG.GT.1) WRITE(15,290)
290     FORMAT(//,
     1  10X,'*********************************************************'/,
     2  10X,'*                                                       *'/,
     3  10X,'*                  >>> L O G S I M <<<                  *'/,
     4  10X,'*                HARVESTING CONFIGURATION               *'/,
     5  10X,'*                                                       *'/,
     6  10X,'*********************************************************'/,
     7  ///)
CC
300   WRITE(*,302)FILENAME,FILENAME,XX(4),XX(10)
      IF(OUTFLAG.GT.1)WRITE(11,302)FILENAME,FILENAME,XX(4),XX(10)
302   FORMAT(/,37X,A,/,33X,'**********************',///
     1 '  NAME OF SIMULATION MODEL          : 'A,/,
     2 '  AMOUNT TO BE HARVESTED (CU.FT.)  : 'F8.0,/,
     3 '  TIME DELAY PARAMETER             :     'F8.4,////,
     4    12X'MACHINE CONFIGURATION',/
     5    12X'=====================',/)
C
      WRITE(*,321)
      IF(OUTFLAG.GT.1)WRITE(11,321)
321   FORMAT(' PROCESS #     IN ORIGIN    OUT DESTINATION'/,
     1' --------------------------------------------')
C
      DO 326 INDEX1=1,13,1
C
      IF(USERARR(23,INDEX1).EQ.0.AND.USERARR(24,INDEX1).EQ.0)THEN
        WRITE(*,'(4X,I2)')INDEX1
        IF(OUTFLAG.GT.1)WRITE(11,'(4X,I2)')INDEX1
        GOTO 327
      ENDIF
      IF(INDEX1.EQ.1) THEN
        WRITE (*,322)INDEX1,USERARR(23,INDEX1)
        IF(OUTFLAG.GT.1)WRITE (11,322)INDEX1,USERARR(23,INDEX1)
 322    FORMAT (4X,I2,23X,F3.0)
      ELSEIF(INDEX1.GE.2.AND.INDEX1.LE.10) THEN
        WRITE (*,323)INDEX1,USERARR(24,INDEX1),USERARR(23,INDEX1)
        IF(OUTFLAG.GT.1)WRITE (11,323)INDEX1,USERARR(24,INDEX1),
     1     USERARR(23,INDEX1)
```

```
323     FORMAT (4X,I2,10X,F3.0,10X,F3.0)
      ELSEIF(INDEX1.EQ.11) THEN
        WRITE (*,324)INDEX1,USERARR(24,INDEX1),USERARR(7,5),
     1  USERARR(7,6)
        IF(OUTFLAG.GT.1)WRITE (11,324)INDEX1,USERARR(24,INDEX1),
     1     USERARR(7,5),USERARR(7,6)
324     FORMAT (4X,I2,10X,F3.0,10X,F3.0,2X,F3.0)
      ELSE
        WRITE (*,325)INDEX1,USERARR(24,INDEX1)
        IF(OUTFLAG.GT.1)WRITE (11,325)INDEX1,USERARR(24,INDEX1)
325     FORMAT (4X,I2,10X,F3.0)
      ENDIF
327   CONTINUE
326   CONTINUE
C
C
C
C DISTRIBUTION DATA OF THE TREES/LOGS/PULPLOGS/SAWLOGS
C ---------------------------------------------------
C
C
400   WRITE(*,402)
      IF(OUTFLAG.GT.1)WRITE(11,402)
402   FORMAT(/////,12X'MATERIAL FREQUENCY DISTRIBUTIONS'/
     1              12X'=================================='/)
C
      DO 406 INDEX1=1,4,1
      WRITE(*,404)INDEX1,DISTRIBNAMES(INDEX1)
      IF(OUTFLAG.GT.1)WRITE(11,404)INDEX1,DISTRIBNAMES(INDEX1)
404   FORMAT('  CUMULATIVE FREQUENCY DISTRIBUTION NO.'I1' : 'A)
406   CONTINUE
C
      WRITE(*,408)
      IF(OUTFLAG.GT.1)WRITE(11,408)
408   FORMAT(//,10X,
     1'DISTRIBUTION 1'4X'DISTRIBUTION 2'4X'DISTRIBUTION 3'4X
     2'DISTRIBUTION 4',/,
     3' CLASS    '
     4'FREQ.%   CU.FT'4X'FREQ.%   CU.FT'4X'FREQ.%   CU.FT'4X
     5'FREQ.%   CU.FT'/,
     6'-------------------------------------------------------------',
     7'---------------')
C
C
C
      DO 450 INDEX1=1,10,1
       IF(DISARR(1,INDEX1).EQ.0.AND.DISARR(2,INDEX1).EQ.0.AND.
     1   DISARR(3,INDEX1).EQ.0.AND.DISARR(4,INDEX1).EQ.0.AND.
     2   DISARR(5,INDEX1).EQ.0.AND.DISARR(6,INDEX1).EQ.0.AND.
     3   DISARR(7,INDEX1).EQ.0.AND.DISARR(8,INDEX1).EQ.0) THEN
         GOTO 449
```

```
       ELSE
         WRITE(*,412)INDEX1
         IF(OUTFLAG.GT.1)WRITE(11,412)INDEX1
412      FORMAT(/,2X,I2,1X,\)
       ENDIF
C
       DO 440 INDEX2=1,8,1
        IF(DISARR(INDEX2,INDEX1).EQ.0) THEN
          WRITE(*,'(9X,\)')
          IF(OUTFLAG.GT.1)WRITE(11,'(9X,\)')
        ELSEIF(DISARR(INDEX2,INDEX1).GT.0) THEN
          WRITE(*,'(1X,F8.2,\)')DISARR(INDEX2,INDEX1)
          IF(OUTFLAG.GT.1)WRITE(11,'(1X,F8.2,\)')DISARR(INDEX2,INDEX1)
        ELSE
          CONTINUE
        ENDIF
440     CONTINUE
C
449   CONTINUE
450   CONTINUE
C
C
C
C
C DESCRIPTION OF THE PROCESSES USED
C --------------------------------
C
C
C
500   WRITE(*,502)
      IF(OUTFLAG.GT.1)WRITE(11,502)
502   FORMAT(/////,12X'INVENTORY AND BUFFER SIZES '/,
     1 12X'==========================')
C
532   WRITE(*,534)
      IF(OUTFLAG.GT.1)WRITE(11,534)
534   FORMAT(/,
     1' PRO-  NAME                   DISTRI-  STARTUP  MINIMUM  STARTUP',
     2'  MAXIMUM  STARTUP'/,
     2' CESS                         BUTION     INV.     INV.   MINIMUM',
     3'     INV.   MAXIMUM'/,
     4' ---------------------------------------------------------------'
     5'------------------')
C
      DO 570 INDEX1=1,13,1
      IF(USERARR(23,INDEX1).EQ.0.AND.USERARR(24,INDEX1).EQ.0) GOTO 569
      IF(INDEX1.EQ.1)THEN
        INDEX2=XXLEVEL(7)+INDEX1
        WRITE(*,536)INDEX1,PROCNAMES(INDEX1),USERARR(5,INDEX1)
        IF(OUTFLAG.GT.1)WRITE(11,536)INDEX1,PROCNAMES(INDEX1),
     1                   USERARR(5,INDEX1)
```

```
536      FORMAT(2X,I2,3X,A,2X,F2.0)
       ELSE
         INDEX2=XXLEVEL(7)+INDEX1
         INDEX3=XXLEVEL(3)+INDEX1
         INDEX4=XXLEVEL(5)+INDEX1
         INDEX5=XXLEVEL(4)+INDEX1
         INDEX6=XXLEVEL(6)+INDEX1
         WRITE(*,538)INDEX1,PROCNAMES(INDEX1),USERARR(5,INDEX1),
     1   XX(INDEX2),XX(INDEX3),XX(INDEX4),XX(INDEX5),XX(INDEX6)
         IF(OUTFLAG.GT.1)WRITE(11,538)INDEX1,PROCNAMES(INDEX1),
     1                  USERARR(5,INDEX1),XX(INDEX2),XX(INDEX3),
     2                  XX(INDEX4),XX(INDEX5),XX(INDEX6)
538      FORMAT(2X,I2,3X,A,2X,F2.0,5X,F8.1,1X,F8.1,
     1   1X,F8.1,1X,F8.1,1X,F8.1)
       ENDIF
569    CONTINUE
570    CONTINUE
C
C
C
C
C PROCESS DESCRIPTION
C --------------------
C
       DO 699 INDEX1=1,13,1
       IF (USERARR(23,INDEX1).EQ.0.AND.USERARR(24,INDEX1).EQ.0) GOTO 698
C
600    WRITE(*,602)INDEX1,PROCNAMES(INDEX1)
       IF(OUTFLAG.GT.1)WRITE(11,602)INDEX1,PROCNAMES(INDEX1)
602    FORMAT(/////,12X'PROCESS NO.'I2': 'A,/,
     1  12X'==============='/)
C
       IF (INDEX1.EQ.1) THEN
         INDEX2=1
         INDEX3=4
         INDEX4=USERARR(23,INDEX1)
         INDEX5=USERARR(5,INDEX1)
604      WRITE(*,605)USERARR(23,INDEX1),PROCNAMES(INDEX4),
     1   USERARR(5,INDEX1),DISTRIBNAMES(INDEX5)
         IF(OUTFLAG.GT.1)WRITE(11,605)USERARR(23,INDEX1),
     1        PROCNAMES(INDEX4),USERARR(5,INDEX1),DISTRIBNAMES(INDEX5)
605      FORMAT(
     1   '  OUTGOING DESTINATION    :  PROCESS NO.'F3.0'  'A,/
     2   '  DISTRIBUTION USED       :  ',F3.0,10X,A)
C
         IF(USERARR(4,INDEX1).EQ.0) THEN
           WRITE(*,606)
           IF(OUTFLAG.GT.1)WRITE(11,606)
606        FORMAT('  LOADER USED             :  NONE')
         ELSE
           INDEX4=USERARR(4,INDEX1)
           WRITE(*,607)USERARR(4,INDEX1),MCHNAMES(INDEX4)
```

```
        IF(OUTFLAG.GT.Ø)WRITE(11,6Ø7)USERARR(4,INDEX1),
     1                 MCHNAMES(INDEX4)
6Ø7     FORMAT('  LOADER USED              :  ',F3.Ø,'   ',A)
       ENDIF
C
       IF(USERARR(9,INDEX1).EQ.Ø) THEN
        WRITE(*,6Ø8)
        IF(OUTFLAG.GT.1)WRITE(11,6Ø8)
6Ø8     FORMAT(
     1   ' TIME DELAYS HANDELED BY :  STANDARD BUILD-IN FUNCTIONS')
       ELSE
        WRITE(*,6Ø9)
        IF(OUTFLAG.GT.1)WRITE(11,6Ø9)
6Ø9     FORMAT(
     1   ' TIME DELAYS HANDELED BY :  USER-WRITTEN FORTRAN SUBROUTINE')
       ENDIF
C
       WRITE(*,612)
       IF(OUTFLAG.GT.1)WRITE(11,612)
612    FORMAT(/,
     1  '  MACHINES USED          :',/
     2  '  TYPE    NAME                        INITIAL # OF MACHINES'/,
     3  '  --------------------------------------------------------')
       DO 618 INDEX4=INDEX2,INDEX3,1
       IF (USERARR(6,INDEX4).EQ.Ø) THEN
         GOTO 616
       ELSE
         WRITE(*,614)INDEX4,MCHNAMES(INDEX4),USERARR(6,INDEX4)
         IF(OUTFLAG.GT.1)WRITE(11,614)INDEX4,MCHNAMES(INDEX4),
     1                  USERARR(6,INDEX4)
614      FORMAT( 3X,I2,5X,A,12X,F3.Ø)
       ENDIF
C
616    CONTINUE
618    CONTINUE
C
C
      ELSEIF (INDEX1.NE.11) THEN
        INDEX2=XXLEVEL(7)+INDEX1
        INDEX3=XXLEVEL(3)+INDEX1
        INDEX4=XXLEVEL(5)+INDEX1
        INDEX5=XXLEVEL(4)+INDEX1
        INDEX6=XXLEVEL(6)+INDEX1
        INDEX7=USERARR(24,INDEX1)
        INDEX8=USERARR(23,INDEX1)
        INDEX9=USERARR(5,INDEX1)
       IF(USERARR(23,INDEX1).EQ.Ø) THEN
        WRITE(*,62Ø)USERARR(24,INDEX1),PROCNAMES(INDEX7)
        IF(OUTFLAG.GT.1)WRITE(11,62Ø)USERARR(24,INDEX1),
     1                 PROCNAMES(INDEX7)
62Ø     FORMAT(
     1  '  INCOMING ORIGIN         :  PROCESS NO.'F3.Ø'  'A)
       ELSE
```

```
      WRITE(*,621)USERARR(24,INDEX1),PROCNAMES(INDEX7),
     1 USERARR(23,INDEX1),PROCNAMES(INDEX8)
      IF(OUTFLAG.GT.1)WRITE(11,621)USERARR(24,INDEX1),
     1  PROCNAMES(INDEX7),USERARR(23,INDEX1),PROCNAMES(INDEX8)
621   FORMAT(
     1 '  INCOMING ORIGIN          :  PROCESS NO.'F3.0'  'A,/,
     2 '  OUTGOING DESTINATION     :  PROCESS NO.'F3.0'  'A)
     ENDIF
      WRITE(*,622)USERARR(5,INDEX1),DISTRIBNAMES(INDEX9),
     1   XX(INDEX2),XX(INDEX3),XX(INDEX4),XX(INDEX5),XX(INDEX6)
      IF(OUTFLAG.GT.1)WRITE(11,622)USERARR(5,INDEX1),
     1  DISTRIBNAMES(INDEX9),XX(INDEX2),XX(INDEX3),XX(INDEX4),
     2  XX(INDEX5),XX(INDEX6)
622   FORMAT(
     1 '  DISTRIBUTION USED       : ',F8.0,10X,A,/
     2 '  STARTUP-INVENTORY LEVEL :  ',F8.1,/
     3 '  MINIMUM INVENTORY LEVEL :  ',F8.1,/
     4 '  STARTUP LEVEL MINIMUM   :  ',F8.1,/
     5 '  MAXIMUM INVENTORY LEVEL :  ',F8.1,/
     6 '  STARTUP LEVEL MAXIMUM   :  ',F8.1)
C
      IF(USERARR(4,INDEX1).EQ.0) THEN
628     WRITE(*,630)
        IF(OUTFLAG.GT.1)WRITE(11,630)
630     FORMAT('  LOADER USED               :  NONE')
      ELSE
        INDEX4=USERARR(4,INDEX1)
632     WRITE(*,634)USERARR(4,INDEX1),MCHNAMES(INDEX4)
        IF(OUTFLAG.GT.1)WRITE(11,634)USERARR(4,INDEX1),
     1    MCHNAMES(INDEX4)
634     FORMAT('  LOADER USED               : ',F8.0,'   ',A)
      ENDIF
C
      IF(USERARR(9,INDEX1).EQ.0) THEN
       WRITE(*,638)
       IF(OUTFLAG.GT.1)WRITE(11,638)
638    FORMAT(
     1  '  TIME DELAYS HANDELED BY :  STANDARD BUILD-IN FUNCTIONS')
      ELSE
       WRITE(*,640)
       IF(OUTFLAG.GT.1)WRITE(11,640)
640    FORMAT(
     1  '  TIME DELAYS HANDELED BY :  USER-WRITTEN FORTRAN SUBROUTINE')
      ENDIF
C
      IF(INDEX1.GE.2.AND.INDEX1.LE.10) THEN
        INDEX2=INDEX1*3-1
        INDEX3=INDEX1*3+1
      ELSEIF(INDEX1.EQ.12) THEN
        INDEX2=37
        INDEX3=40
```

```
        ELSEIF(INDEX1.EQ.13) THEN
          INDEX2=41
          INDEX3=42
        ELSE
          CONTINUE
        ENDIF
        WRITE(*,642)
        IF(OUTFLAG.GT.1)WRITE(11,642)
642     FORMAT(/,
     1  ' MACHINES USED           :'/,
     2  ' TYPE    NAME                         INITIAL # OF MACHINES'/,
     3  ' -----------------------------------------------------')
        DO 648 INDEX4=INDEX2,INDEX3,1
        IF (USERARR(6,INDEX4).EQ.0) THEN
          GOTO 646
        ELSE
          WRITE(*,644)INDEX4,MCHNAMES(INDEX4),USERARR(6,INDEX4)
          IF(OUTFLAG.GT.1)WRITE(11,644)INDEX4,MCHNAMES(INDEX4),
     1        USERARR(6,INDEX4)
644       FORMAT( 3X,I2,5X,A,12X,F3.0)
        ENDIF
646     CONTINUE
648     CONTINUE
C
       ELSEIF(INDEX1.EQ.11) THEN
        INDEX2=XXLEVEL(7)+INDEX1
        INDEX3=XXLEVEL(3)+INDEX1
        INDEX4=XXLEVEL(5)+INDEX1
        INDEX5=XXLEVEL(4)+INDEX1
        INDEX6=XXLEVEL(6)+INDEX1
        INDEX7=USERARR(24,INDEX1)
        INDEX8=USERARR(7,5)
        INDEX9=USERARR(7,6)
650     WRITE(*,652)USERARR(24,INDEX1),PROCNAMES(INDEX7),
     1  USERARR(7,5),PROCNAMES(INDEX8),
     2  USERARR(7,6),PROCNAMES(INDEX9),
     3  USERARR(7,3),USERARR(7,4)
        IF(OUTFLAG.GT.1)WRITE(11,652)
     1  USERARR(24,INDEX1),PROCNAMES(INDEX7),
     2  USERARR(7,5),PROCNAMES(INDEX8),
     3  USERARR(7,6),PROCNAMES(INDEX9),
     4  USERARR(7,3),USERARR(7,4)
652     FORMAT(
     1  '  INCOMING ORIGIN         :  PROCESS NO.'F3.0'  'A,/,
     2  '  OUTGOING ROUTE 1        :  PROCESS NO.'F3.0'  'A,/,
     3  '  OUTGOING ROUTE 2        :  PROCESS NO.'F3.0'  'A,/,
     4  '  % GOING ROUTE 1         :   ',F8.2,' %',/
     5  '  % GOING ROUTE 2         :   ',F8.2,' %')
        INDEX9=USERARR(5,INDEX1)
654     WRITE(*,656)USERARR(5,INDEX1),DISTRIBNAMES(INDEX9),XX(INDEX2),
     1   XX(INDEX3),XX(INDEX4),XX(INDEX5),XX(INDEX6)
```

```
      IF(OUTFLAG.GT.1)WRITE(11,656)USERARR(5,INDEX1),
     1  DISTRIBNAMES(INDEX9),XX(INDEX2),
     2  XX(INDEX3),XX(INDEX4),XX(INDEX5),XX(INDEX6)
656   FORMAT(
     1 '  DISTRIBUTION USED       : ',F8.0,10X,A,/,
     2 '  STARTUP-INVENTORY LEVEL :  ',F8.1,/,
     3 '  MINIMUM INVENTORY LEVEL :  ',F8.1,/,
     4 '  STARTUP LEVEL MINIMUM   :  ',F8.1,/,
     5 '  MAXIMUM INVENTORY LEVEL :  ',F8.1,/,
     6 '  STARTUP LEVEL MAXIMUM   :  ',F8.1)
C
      IF(USERARR(4,INDEX1).EQ.0) THEN
658     WRITE(*,660)
        IF(OUTFLAG.GT.1)WRITE(11,660)
660     FORMAT('  LOADER USED              :  NONE')
      ELSE
        INDEX4=USERARR(4,INDEX1)
662     WRITE(*,664)USERARR(4,INDEX1),MCHNAMES(INDEX4)
        IF(OUTFLAG.GT.1)WRITE(11,664)USERARR(4,INDEX1),
     1     MCHNAMES(INDEX4)
664     FORMAT('  LOADER USED              : ',F8.0,'    ',A)
      ENDIF
C
      IF(USERARR(9,INDEX1).EQ.0) THEN
       WRITE(*,668)
       IF(OUTFLAG.GT.1)WRITE(11,668)
668    FORMAT(
     1  '  TIME DELAYS HANDELED BY :  STANDARD BUILD-IN FUNCTIONS')
      ELSE
       WRITE(*,670)
       IF(OUTFLAG.GT.1)WRITE(11,670)
670    FORMAT(
     1  '  TIME DELAYS HANDELED BY :  USER-WRITTEN FORTRAN SUBROUTINE')
      ENDIF
C
     ELSE
      CONTINUE
     ENDIF
C
698  CONTINUE
699  CONTINUE
C
C
C
C MACHINE PARAMETERS
C ==================================================================
C
C
     DO 780 INDEX1=1,42,1
      IF (USERARR(6,INDEX1).EQ.0) GOTO 779
C
700   WRITE(*,702)INDEX1,MCHNAMES(INDEX1)
      IF(OUTFLAG.GT.1)WRITE(11,702)INDEX1,MCHNAMES(INDEX1)
```

```
702   FORMAT(/////,12X'MACHINE TYPE 'I2': 'A,/,
     1  12X'================'/)
C
      IF(INDEX1.EQ.40.OR.INDEX1.EQ.42)GOTO 711
      WRITE(*,703)USERARR(6,INDEX1)
      IF(OUTFLAG.GT.1)WRITE(11,703)USERARR(6,INDEX1)
703   FORMAT('  INITIAL NUMBER OF MACHINES        :'F6.0)
      IF (INDEX1.EQ.39) GOTO 705
      WRITE(*,704)USERARR(1,INDEX1)
      IF(OUTFLAG.GT.1)WRITE(11,704)USERARR(1,INDEX1)
704   FORMAT('  AVERAGE PROCESSING TIME / TREE    :   ',F8.4)
705   WRITE(*,706)USERARR(2,INDEX1)
      IF(OUTFLAG.GT.1)WRITE(11,706)USERARR(2,INDEX1)
706   FORMAT('  FIXED CONSTANT TIME / LOAD        :   ',F8.4)
      WRITE(*,708)USERARR(3,INDEX1)
      IF(OUTFLAG.GT.1)WRITE(11,708)USERARR(3,INDEX1)
708   FORMAT('  FIXED CONST. TIME / ONE WAY HAUL  :   ',F8.4)
711   WRITE(*,712)USERARR(21,INDEX1)
      IF(OUTFLA.GT.1)WRITE(11,712)USERARR(21,INDEX1)
712   FORMAT('  FIXED COST / SCHEDULED HOUR       : ',F8.2)
      WRITE(*,714)USERARR(22,INDEX1)
      IF(OUTFLAG.GT.1)WRITE(11,714)USERARR(22,INDEX1)
714   FORMAT('  VARIABLE COST/ MACHINE HOUR       : ',F8.2)
      WRITE(*,710)USERARR(8,INDEX1)
      IF(OUTFLAG.GT.1)WRITE(11,710)USERARR(8,INDEX1)
710   FORMAT('  MACHINE CAPACITY IN CU.FT         : ',F8.2)
C
C
720   WRITE(*,722)INDEX1,MCHNAMES(INDEX1)
      IF(OUTFLAG.GT.1)WRITE(11,722)INDEX1,MCHNAMES(INDEX1)
722   FORMAT(/,
     1 '  FREQUENCY DISTRIBUTIONS MACHINE TYPE 'I2' : 'A,/
     3 '   CLASS       CUM FREQ.%    TIME BETW.FAILURE    CUM.FREQ.%',
     4 '     REPAIR TIME'/,
     5 ' ------------------------------------------------------------',
     6 '-----------------')
       IF(MCHARR(INDEX1,1,1).EQ.0.)THEN
         WRITE(*,723)
         IF(OUTFLAG.GT.1)WRITE(11,723)
723      FORMAT(/,20X,'******  DISTRIBUTION NOT USED  ******')
       ELSE
         DO 760 INDEX2=1,10,1
           IF(MCHARR(INDEX1,1,INDEX2).EQ.0.AND.
     1     MCHARR(INDEX1,3,INDEX2).EQ.0) THEN
            GOTO 759
           ELSEIF (MCHARR(INDEX1,1,INDEX2).GT.0.AND.
     1     MCHARR(INDEX1,3,INDEX2).EQ.0) THEN
            WRITE(*,724)INDEX2,MCHARR(INDEX1,1,INDEX2),
     1      MCHARR(INDEX1,2,INDEX2)
            IF(OUTFLAG.GT.1)WRITE(11,724)INDEX2,
     1         MCHARR(INDEX1,1,INDEX2),MCHARR(INDEX1,2,INDEX2)
```

```
724          FORMAT(4X,I2,10X,F8.2,10X,F8.2)
            ELSEIF (MCHARR(INDEX1,1,INDEX2).EQ.0.AND.
     1      MCHARR(INDEX1,3,INDEX2).GT.0) THEN
             WRITE(*,726)INDEX2,MCHARR(INDEX1,3,INDEX2),
     1       MCHARR(INDEX1,4,INDEX2)
             IF(OUTFLAG.GT.1)WRITE(11,726)INDEX2,MCHARR(INDEX1,3,INDEX2)
     1         ,MCHARR(INDEX1,4,INDEX2)
726          FORMAT(4X,I2,10X,8X,10X,8X,10X,F8.2,10X,F8.2)
            ELSE
             WRITE(*,728)INDEX2,MCHARR(INDEX1,1,INDEX2),
     1       MCHARR(INDEX1,2,INDEX2),MCHARR(INDEX1,3,INDEX2),
     2       MCHARR(INDEX1,4,INDEX2)
             IF(OUTFLAG.GT.1)WRITE(11,728)INDEX2,MCHARR(INDEX1,1,INDEX2)
     1       ,MCHARR(INDEX1,2,INDEX2),MCHARR(INDEX1,3,INDEX2),
     2       MCHARR(INDEX1,4,INDEX2)
728          FORMAT(4X,I2,10X,F8.2,10X,F8.2,10X,F8.2,10X,F8.2)
            ENDIF
759         CONTINUE
760        CONTINUE
        ENDIF
C
779     CONTINUE
780   CONTINUE
C
C
C END OF SUBROUTINE
C ================
C
      WRITE(*,'(////)')
      IF(OUTFLAG.GT.1)WRITE(11,'(////)')
      IF(OUTFLAG.GT.1)CLOSE(11)
9998  RETURN
C
      END
```

APPENDIX D

5. Listing, MODIFY.FOR

```
C*************************************************************
C*                                                           *
C*                 OREGON STATE UNIVERSITY                   *
C*                        JUNE 1986                          *
C*                                                           *
C*                   >>> L O G S I M <<<                     *
C*                                                           *
C*    SIMULATION OF MECHANIZED LOG HARVESTING SYSTEMS        *
C*                                                           *
C*                                                           *
C*    DESIGNED BY : CHRISTOPH WIESE                          *
C*                  MASTERS CANDIDATE, DEP. OF INDUSTRIAL    *
C*                  ENGINEERING, OREGON STATE UNIVERSITY     *
C*                                                           *
C*    DESIGNED FOR: DEPARTMENT OF FOREST ENGINEERING         *
C*                  OREGON STATE UNIVERSITY                  *
C*                                                           *
C*                                                           *
C*    SUPERVISION : DR. ELDON OLSEN                          *
C*                  ASSOCIATE PROFESSOR, DEP. OF INDUSTRIAL  *
C*                  ENGINEERING, OREGON STATE UNIVERSITY     *
C*                                                           *
C*                                                           *
C*************************************************************
C*                                                           *
C*       FORTRAN INPUT USER-INTERFACE: MODIFY.FOR            *
C*                                                           *
C*                   31-MAY-87  18:55                        *
C*                                                           *
C*************************************************************
C
C
C
C
C
$INCLUDE:'PRCTL.FOR'
C
C PROGRAM DECLARATION:
C ===================
C
      SUBROUTINE MODIFY
C
C COMMON BLOCK :
C =============
C
$INCLUDE:'VARBLOCK.DOC'
C
C DEFINE LOCAL VARIABLES, NAMES & TYPE:
C ====================================
C
```

```
      INTEGER*4 IANSWER,INDEX1,INDEX2,INDEX3,INDEX4,INDEX5
      INTEGER*4 INDEX6,INDEX7,INDEX8,INDEX9,IND10,IND11
      CHARACTER*20 CHRANSWER
      REAL F1ANSWER,F2ANSWER,F3ANSWER,F4ANSWER,F5ANSWER,F6ANSWER
      REAL F7ANSWR,F8ANSWR,F9ANSWER,F10ANSWR,F11ANSWER,F12ANSWER
      LOGICAL*4 FILESTATUS
C
C BEGIN PROCESSING:
C =================
C
C OPENING SCREEN:
C ---------------
C
      WRITE(*,100)
100   FORMAT('1'/////////////////////////////////////5X,
     1'                   SUBROUTINE MODIFY'/5X,
     2'                   ================='//5X,
     3'THIS SUBROUTINE ALLOWS YOU TO MODIFY THE DATA OF A '/5X,
     4'SIMULATION MODEL PREVIOUSLY DEFINED WITH SUBROUTINE '/5X,
     5'READIN.',///)
      WRITE(*,'(7X,A\)') 'DO YOU WISH TO CONTINUE (Y/N) ?  [Y]-----> '
      READ (*,'(BZ,A1)') CHRANSWER
      IF (CHRANSWER.EQ.'N') THEN
        GOTO 9998
      ELSE
        CONTINUE
      ENDIF
C
C
C
C INITIALIZATION OF ALL VARIABLES AN READIN THE FILE
C ==================================================
C
C
      CALL INITREAD
C
C
200   WRITE(*,205)
205   FORMAT(/////////////////////,
     1 5X'SUBRROUTINE MODIFY CHOICES:',/,
     2 5X'==========================',//,
     3 2X'EDIT SYSTEM PARAMETERS                   = 1',/,
     4 2X'EDIT MATERIAL FREQUENCY DISTRIBUTIONS    = 2',/,
     5 2X'EDIT PROCESS PARAMETERS                  = 3',/,
     6 2X'EDIT MACHINE PARAMETERS                  = 4',/,
     7 2X'EDIT MACHINE DISTRIBUTIONS               = 5')
      WRITE(*,210)
210   FORMAT(
     1 2X'SAVE MODIFYIED MODEL                     = 6',/,
     2 2X'RETURN TO MAIN MENU                      = 0',//,
     3 2X'        PLEASE ENTER CHOICE         -----> '\)
220   READ(*,'(I2)')IANSWER
C
```

```
      IF(IANSWER.LT.Ø.OR.IANSWER.GT.6)THEN
        WRITE(*,23Ø)
23Ø     FORMAT(/,2X,'  !!!  CANNOT BE, PLEASE TRY AGAIN  !!!',/,
     1  2X'       PLEASE ENTER CHOICE        -----> '\)
        GOTO 22Ø
      ELSEIF(IANSWER.EQ.Ø)THEN
        RETURN
      ELSEIF(IANSWER.EQ.1)THEN
        GOTO 1ØØØ
      ELSEIF(IANSWER.EQ.2)THEN
        GOTO 2ØØØ
      ELSEIF(IANSWER.EQ.3)THEN
        GOTO 3ØØØ
      ELSEIF(IANSWER.EQ.4)THEN
        GOTO 4ØØØ
      ELSEIF(IANSWER.EQ.5)THEN
        GOTO 5ØØØ
      ELSEIF(IANSWER.EQ.6)THEN
        GOTO 6ØØØ
      ELSE
        CONTINUE
      ENDIF
C
C
C
C
C
C MODIFYING SYSTEM PARAMETERS
C ==========================
C
C
1ØØØ  WRITE(*,1Ø1Ø)FILENAME,XX(4),XX(1Ø)
1Ø1Ø  FORMAT(////////////////////////,
     1 '  EDITING SYSTEM PARAMETERS:',/,
     2 '  --------------------------',//
     1 2X'      NAME OF SIMULATION MODEL         : 'A,/,
     2 2X'1 =   AMOUNT TO BE HARVESTED (CU.FT.)  : 'F8.Ø,/,
     3 2X'2 =   TIME DELAY PARAMETER             :    'F8.4,/,
     4 2X'Ø =   RETURN TO MODIFY MENU',//,
     4 2X'PLEASE ENTER CHOICE    ----> ',\)
1Ø2Ø  READ(*,'(I2)')IANSWER
C
      IF(IANSWER.LT.Ø.OR.IANSWER.GT.2)THEN
        WRITE(*,1Ø3Ø)
1Ø3Ø     FORMAT(/,2X,'!!!  CANNOT BE, PLEASE TRY AGAIN  !!!',/,
     1  2X'PLEASE ENTER CHOICE   ----> '\)
        GOTO 1Ø2Ø
      ELSEIF(IANSWER.EQ.Ø)THEN
        GOTO 2ØØ
      ELSEIF(IANSWER.EQ.1) THEN
1Ø4Ø    WRITE(*,'(//,A,\)')' HOW MANY CU.FT SHOULD BE HARVESTED? --->'
        READ(*,'(F8.Ø)')F1ANSWER
```

```
        IF(F1ANSWER.EQ.0) THEN
          WRITE (*,'(/,A)')' !!! CANNOT BE, PLEASE TRY AGAIN !!!'
          GOTO 1040
        ELSEIF(F1ANSWER.GE.9999998) THEN
          WRITE (*,'(/,A)')' !!! CANNOT BE, PLEASE TRY AGAIN !!!'
          GOTO 1040
        ELSE
          XX(4)=F1ANSWER
        ENDIF
      ELSEIF(IANSWER.EQ.2) THEN
1050    WRITE(*,'(//,A,\)')' VALUE OF THE TIME DELAY PARAMETER?  --->'
        READ(*,'(F8.4)')F1ANSWER
        IF(F1ANSWER.EQ.0) THEN
          WRITE (*,'(/,A)')' !!! CANNOT BE, PLEASE TRY AGAIN !!!'
          GOTO 1050
        ELSEIF(F1ANSWER.GE.999.OR.F1ANSWER.LT.0.0001) THEN
          WRITE (*,'(/,A)')' !!! CANNOT BE, PLEASE TRY AGAIN !!!'
          GOTO 1050
        ELSE
          XX(10)=F1ANSWER
        ENDIF
      ELSE
        CONTINUE
      ENDIF
C
      GOTO 1000
C
C
C MODIFYING DISTRIBUTION DATA OF THE TREES/LOGS/PULPLOGS/SAWLOGS
C ==============================================================
C
C
2000  WRITE(*,2010)DISTRIBNAMES(1),DISTRIBNAMES(2),DISTRIBNAMES(3),
     1  DISTRIBNAMES(4)
2010  FORMAT(////////////////,2X'EDITING MATERIAL DISTRIBUTIONS:',/,
     1 '  ------------------------------',//
     2 2X'1 =   EDIT DISTRIBUTION NO.1 ',A,/,
     3 2X'2 =   EDIT DISTRIBUTION NO.2 ',A,/,
     4 2X'3 =   EDIT DISTRIBUTION NO.3 ',A,/,
     5 2X'4 =   EDIT DISTRIBUTION NO.4 ',A,/,
     6 2X'0 =   RETURN TO MODIFY MENU',//,
     7 2X'PLEASE ENTER CHOICE   ----> ',\)
2020   READ(*,'(I2)')IANSWER
C
      IF(IANSWER.LT.0.OR.IANSWER.GT.4)THEN
        WRITE(*,2030)
2030      FORMAT(/,2X,'!!!  CANNOT BE, PLEASE TRY AGAIN  !!!',/,
     1  2X'PLEASE ENTER CHOICE  ----> '\)
        GOTO 2020
      ELSEIF(IANSWER.EQ.0)THEN
        GOTO 200
```

```
      ELSE
        INDEX1=IANSWER
      ENDIF
C
C
C
2035  WRITE(*,2040)INDEX1,DISTRIBNAMES(INDEX1)
2040  FORMAT(////////////////////////////////,
     1 10X,'DISTRIBUTION NO.',
     2 I2' :  ',A,/,
     3 10X,'====================='//,
     4'   CLASS      CUM.REL.FREQ.%        CU.FT'/,
     5'  -----------------------------------------',)
        IF(INDEX1.EQ.1)THEN
          IF(DISARR(1,1).EQ.0)THEN
            WRITE(*,'(/,3X,A)')'*******  DISTRIBUTION NOT USED  *******'
          ELSE
            DO 2060 INDEX2=1,10,1
              IF(DISARR(1,INDEX2).EQ.0)GOTO 2060
              WRITE(*,2052)INDEX2,DISARR(1,INDEX2),DISARRR(2,INDEX2)
2052          FORMAT(4X,I2,10X,F8.2,10X,F8.2)
2060        CONTINUE
          ENDIF
          WRITE(*,'(//,A,\)')'  EDIT DISTRIBUTION  (Y/N) ?  [N]---->'
          READ(*,'(A1)')CHRANSWER
          IF(CHRANSWER.EQ.'Y')GOTO 2100
        ELSEIF(INDEX1.EQ.2)THEN
          IF(DISARR(3,1).EQ.0)THEN
            WRITE(*,'(/,3X,A)')'*******  DISTRIBUTION NOT USED  *******'
          ELSE
            DO 2062 INDEX2=1,10,1
              IF(DISARR(3,INDEX2).EQ.0)GOTO 2062
              WRITE(*,2061)INDEX2,DISARR(3,INDEX2),DISARRR(4,INDEX2)
2061          FORMAT(4X,I2,10X,F8.2,10X,F8.2)
2062        CONTINUE
          ENDIF
          WRITE(*,'(//,A,\)')'  EDIT DISTRIBUTION  (Y/N) ?  [N]---->'
          READ(*,'(A1)')CHRANSWER
          IF(CHRANSWER.EQ.'Y')GOTO 2100
        ELSEIF(INDEX1.EQ.3)THEN
          IF(DISARR(5,1).EQ.0)THEN
            WRITE(*,'(/,3X,A)')'*******  DISTRIBUTION NOT USED  *******'
          ELSE
            DO 2064 INDEX2=1,10,1
              IF(DISARR(5,INDEX2).EQ.0)GOTO 2064
              WRITE(*,2063)INDEX2,DISARR(5,INDEX2),DISARRR(6,INDEX2)
2063          FORMAT(4X,I2,10X,F8.2,10X,F8.2)
2064        CONTINUE
          ENDIF
          WRITE(*,'(//,A,\)')'  EDIT DISTRIBUTION  (Y/N) ?  [N]---->'
          READ(*,'(A1)')CHRANSWER
          IF(CHRANSWER.EQ.'Y')GOTO 2100
```

```
      ELSE
        IF(DISARR(7,1).EQ.Ø)THEN
          WRITE(*,'(/,3X,A)')'*******  DISTRIBUTION NOT USED  *******'
        ELSE
          DO 2Ø66 INDEX2=1,1Ø,1
            IF(DISARR(7,INDEX2).EQ.Ø)GOTO 2Ø66
            WRITE(*,2Ø65)INDEX2,DISARR(7,INDEX2),DISARRR(8,INDEX2)
2Ø65         FORMAT(4X,I2,1ØX,F8.2,1ØX,F8.2)
2Ø66       CONTINUE
        ENDIF
        WRITE(*,'(//,A,\)')'  EDIT DISTRIBUTION  (Y/N) ?  [N]---->'
        READ(*,'(A1)')CHRANSWER
        IF(CHRANSWER.EQ.'Y')GOTO 21ØØ
      ENDIF
      GOTO 2ØØØ
C
C  INITIALIZE THE DISTRIBUTION
C
21ØØ  DO 21Ø5 INDEX2=1,1Ø,1
      IF(INDEX1.EQ.1)THEN
        DISARR(1,INDEX2)=Ø
        DISARR(2,INDEX2)=Ø
      ELSEIF(INDEX1.EQ.2)THEN
        DISARR(3,INDEX2)=Ø
        DISARR(4,INDEX2)=Ø
      ELSEIF(INDEX1.EQ.3)THEN
        DISARR(5,INDEX2)=Ø
        DISARR(6,INDEX2)=Ø
      ELSE
        DISARR(7,INDEX2)=Ø
        DISARR(8,INDEX2)=Ø
      ENDIF
21Ø5  CONTINUE
C
C
      WRITE(*,211Ø)INDEX1
211Ø   FORMAT(/,
     1  ' FREQUENCY DISTRIBUTION NO.'I2,':'/,
     2  ' ----------------------------')
C
      WRITE(*,212Ø)
212Ø  FORMAT(/,'  NAME OF THIS DISTRIBUTION ?   -----> ',\)
      READ(*,'(A)')CHRANSWER
      DISTRIBNAMES(INDEX1)=CHRANSWER
C
213Ø  DO 218Ø INDEX2=1,1Ø,1
214Ø    WRITE(*,215Ø)INDEX2
215Ø    FORMAT(/,' CLASS ',I2,': CUM.REL.FREQENCY?  [Ø]--->'\)
        READ(*,'(BN,F8.2)')F1ANSWER
        WRITE(*,216Ø)INDEX2
```

```
2160      FORMAT(' CLASS ',I2,': VOLUME CU.FT?      [0]--->'\)
          READ(*,'(BN,F8.2)')F2ANSWER
            IF(INDEX2.EQ.1.AND.F1ANSWER.EQ.0)THEN
             GOTO 2190
            ELSEIF(F2ANSWER.LE.0)THEN
             WRITE(*,'(A,/)')' !!  CANNOT BE, PLEASE TRY AGAIN  !!'
             GOTO 2140
            ELSEIF(F1ANSWER.GT.100)THEN
             WRITE(*,'(A,/)')' !!  CANNOT BE, PLEASE TRY AGAIN  !!'
             GOTO 2140
            ELSEIF(INDEX2.GT.1.AND.F1ANSWER.LE.F3ANSWER)THEN
             WRITE(*,'(A,/)')' !!  CANNOT BE, PLEASE TRY AGAIN  !!'
             GOTO 2140
            ELSEIF(INDEX2.EQ.10.AND.F1ANSWER.NE.100)THEN
             WRITE(*,'(A,/)')' !!  CANNOT BE, PLEASE TRY AGAIN  !!'
             GOTO 2140
            ELSE
             F3ANSWER=F1ANSWER
             IF(INDEX1.EQ.1)THEN
               DISARR(1,INDEX2)=F1ANSWER
               DISARR(2,INDEX2)=F2ANSWER
               IF(F1ANSWER.EQ.100)GOTO 2190
             ELSEIF(INDEX1.EQ.2)THEN
               DISARR(3,INDEX2)=F1ANSWER
               DISARR(4,INDEX2)=F2ANSWER
               IF(F1ANSWER.EQ.100)GOTO 2190
             ELSEIF(INDEX1.EQ.3)THEN
               DISARR(5,INDEX2)=F1ANSWER
               DISARR(6,INDEX2)=F2ANSWER
               IF(F1ANSWER.EQ.100)GOTO 2190
             ELSE
               DISARR(7,INDEX2)=F1ANSWER
               DISARR(8,INDEX2)=F2ANSWER
               IF(F1ANSWER.EQ.100)GOTO 2190
             ENDIF
           ENDIF
2180  CONTINUE
2190  CONTINUE
C
      GOTO 2035
C
C
C
C  EDIT PROCESS PARAMETERS
C  =======================
C
C
3000  WRITE(*,3010)
3010  FORMAT(///////////////////////,2X'EDITING PROCESS PARAMETERS:',/,
     1                          '  ---------------------------',/)
      DO 3050,INDEX1=1,13,1
      IF(USERARR(23,INDEX1).EQ.0.AND.USERARR(24,INDEX1).EQ.0)GOTO 3040
        WRITE(*,3020)INDEX1,INDEX1,PROCNAMES(INDEX1)
```

```
3020   FORMAT(2X,I2'  =   PROCESS NO.'I2' :',A)
3040   CONTINUE
3050  CONTINUE
      WRITE(*,3060)
3060  FORMAT(2X' 0  = RETURN TO MODIFY MENU',//
     1       2X'PLEASE ENTER CHOICE   ----> ',\)
3070   READ(*,'(I2)')IANSWER
C
      IF(IANSWER.LT.0.OR.IANSWER.GT.13)THEN
        WRITE(*,3080)
3080    FORMAT(/,2X,'!!!  CANNOT BE, PLEASE TRY AGAIN  !!!',/,
     1  2X'PLEASE ENTER CHOICE  ----> '\)
        GOTO 3070
      ELSEIF(IANSWER.EQ.0)THEN
        GOTO 200
      ELSEIF(USERARR(23,IANSWER).EQ.0.AND.USERARR(24,IANSWER).EQ.0)THEN
        WRITE(*,3090)
3090    FORMAT(/,2X,'!!!  PROCESS NOT ACTIVE, PLEASE TRY AGAIN  !!!',/,
     1  2X'PLEASE ENTER CHOICE  ----> '\)
        GOTO 3070
      ELSE
        INDEX1=IANSWER
      ENDIF
C
C
C
3100  WRITE(*,3110)INDEX1,PROCNAMES(INDEX1)
3110  FORMAT(////////////////////////////////,
     1       12X'PROCESS NO.'I2': 'A,/,
     2       12X'==============='/)
C
      IF (INDEX1.EQ.1) THEN
        INDEX2=1
        INDEX3=4
        INDEX4=USERARR(23,INDEX1)
        INDEX5=USERARR(5,INDEX1)
3120    WRITE(*,3130)USERARR(23,INDEX1),PROCNAMES(INDEX4),
     1  USERARR(5,INDEX1),DISTRIBNAMES(INDEX5)
3130    FORMAT(
     1  '  OUTGOING DESTINATION    :  PROCESS NO.'F3.0'  'A,/
     2  '  DISTRIBUTION USED       :  ',F3.0,10X,A)
C
        IF(USERARR(4,INDEX1).EQ.0) THEN
          WRITE(*,3140)
3140      FORMAT('  LOADER USED              :  NONE')
        ELSE
          INDEX4=USERARR(4,INDEX1)
          WRITE(*,3150)USERARR(4,INDEX1),MCHNAMES(INDEX4)
3150      FORMAT('  LOADER USED              :  ',F3.0,'   ',A)
        ENDIF
C
        IF(USERARR(9,INDEX1).EQ.0) THEN
         WRITE(*,3160)
```

```
3160    FORMAT(
     1   ' TIME DELAYS HANDELED BY :  STANDARD BUILD-IN FUNCTIONS')
        ELSE
         WRITE(*,3170)
3170    FORMAT(
     1   ' TIME DELAYS HANDELED BY :  USER-WRITTEN FORTRAN SUBROUTINE')
        ENDIF
C
C
      ELSEIF (INDEX1.NE.11) THEN
        INDEX2=XXLEVEL(7)+INDEX1
        INDEX3=XXLEVEL(3)+INDEX1
        INDEX4=XXLEVEL(5)+INDEX1
        INDEX5=XXLEVEL(4)+INDEX1
        INDEX6=XXLEVEL(6)+INDEX1
        INDEX7=USERARR(24,INDEX1)
        INDEX8=USERARR(23,INDEX1)
        INDEX9=USERARR(5,INDEX1)
       IF(USERARR(23,INDEX1).EQ.0) THEN
        WRITE(*,3180)USERARR(24,INDEX1),PROCNAMES(INDEX7)
3180    FORMAT(
     1  '  INCOMING ORIGIN        :  PROCESS NO.'F3.0'  'A)
       ELSE
        WRITE(*,3190)USERARR(24,INDEX1),PROCNAMES(INDEX7),
     1  USERARR(23,INDEX1),PROCNAMES(INDEX8)
3190    FORMAT(
     1  '  INCOMING ORIGIN        :  PROCESS NO.'F3.0'  'A,/,
     2  '  OUTGOING DESTINATION   :  PROCESS NO.'F3.0'  'A)
       ENDIF
        WRITE(*,3200)USERARR(5,INDEX1),DISTRIBNAMES(INDEX9),
     1    XX(INDEX2),XX(INDEX3),XX(INDEX4),XX(INDEX5),XX(INDEX6)
3200    FORMAT(
     1  '  DISTRIBUTION USED      : ',F8.0,10X,A,/
     2  '  STARTUP-INVENTORY LEVEL :  ',F8.1,/
     3  '  MINIMUM INVENTORY LEVEL :  ',F8.1,/
     4  '  STARTUP LEVEL MINIMUM   :  ',F8.1,/
     5  '  MAXIMUM INVENTORY LEVEL :  ',F8.1,/
     6  '  STARTUP LEVEL MAXIMUM   :  ',F8.1)
C
        IF(USERARR(4,INDEX1).EQ.0) THEN
3210      WRITE(*,3220)
3220      FORMAT('  LOADER USED              :  NONE')
        ELSE
          INDEX4=USERARR(4,INDEX1)
3230      WRITE(*,3240)USERARR(4,INDEX1),MCHNAMES(INDEX4)
3240      FORMAT('  LOADER USED              : ',F8.0,'   ',A)
        ENDIF
C
        IF(USERARR(9,INDEX1).EQ.0) THEN
         WRITE(*,3250)
3250     FORMAT(
     1   ' TIME DELAYS HANDELED BY :  STANDARD BUILD-IN FUNCTIONS')
```

```
       ELSE
        WRITE(*,3260)
3260    FORMAT(
     1   ' TIME DELAYS HANDELED BY :  USER-WRITTEN FORTRAN SUBROUTINE')
       ENDIF
C
C
      ELSEIF(INDEX1.EQ.11) THEN
       INDEX2=XXLEVEL(7)+INDEX1
       INDEX3=XXLEVEL(3)+INDEX1
       INDEX4=XXLEVEL(5)+INDEX1
       INDEX5=XXLEVEL(4)+INDEX1
       INDEX6=XXLEVEL(6)+INDEX1
       INDEX7=USERARR(24,INDEX1)
       INDEX8=USERARR(7,5)
       INDEX9=USERARR(7,6)
3280   WRITE(*,3290)USERARR(24,INDEX1),PROCNAMES(INDEX7),
     1 USERARR(7,5),PROCNAMES(INDEX8),
     2 USERARR(7,6),PROCNAMES(INDEX9),
     3 USERARR(7,3),USERARR(7,4)
3290   FORMAT(
     1 '  INCOMING ORIGIN          :  PROCESS NO.'F3.0'  'A,/,
     2 '  OUTGOING ROUTE 1         :  PROCESS NO.'F3.0'  'A,/,
     3 '  OUTGOING ROUTE 2         :  PROCESS NO.'F3.0'  'A,/,
     4 '  % GOING ROUTE 1          :   ',F8.2,' %',/
     5 '  % GOING ROUTE 2          :   ',F8.2,' %')
       INDEX9=USERARR(5,INDEX1)
3300   WRITE(*,3310)USERARR(5,INDEX1),DISTRIBNAMES(INDEX9),XX(INDEX2),
     1  XX(INDEX3),XX(INDEX4),XX(INDEX5),XX(INDEX6)
3310   FORMAT(
     1 '  DISTRIBUTION USED        : ',F8.0,10X,A,/,
     2 '  STARTUP-INVENTORY LEVEL :  ',F8.1,/,
     3 '  MINIMUM INVENTORY LEVEL :  ',F8.1,/,
     4 '  STARTUP LEVEL MINIMUM   :  ',F8.1,/,
     5 '  MAXIMUM INVENTORY LEVEL :  ',F8.1,/,
     6 '  STARTUP LEVEL MAXIMUM   :  ',F8.1)
C
       IF(USERARR(4,INDEX1).EQ.0) THEN
3320     WRITE(*,3340)
3340     FORMAT('  LOADER USED                 :  NONE')
       ELSE
         INDEX4=USERARR(4,INDEX1)
3350     WRITE(*,3360)USERARR(4,INDEX1),MCHNAMES(INDEX4)
3360     FORMAT('  LOADER USED                 : ',F8.0,'   ',A)
       ENDIF
C
       IF(USERARR(9,INDEX1).EQ.0) THEN
        WRITE(*,3370)
3370    FORMAT(
     1   ' TIME DELAYS HANDELED BY :  STANDARD BUILD-IN FUNCTIONS')
       ELSE
        WRITE(*,3380)
```

```
3380     FORMAT(
     1    ' TIME DELAYS HANDELED BY :  USER-WRITTEN FORTRAN SUBROUTINE')
         ENDIF
C
       ELSE
        CONTINUE
       ENDIF
C
       WRITE(*,3390)
3390   FORMAT(/,2X,'CONTINUE EDITING? Y/N   [N]----> '\)
       READ(*,'(A)')CHRANSWER
       IF (CHRANSWER.EQ.'Y') THEN
         GOTO 3490
       ELSE
         GOTO 3000
       ENDIF
C
C
C
C
3490   WRITE(*,3500)INDEX1
3500   FORMAT(//' EDITING PROCESS NO.'I2,/
     1          ' ---------------------')
       WRITE(*,'(A,\)')' NAME OF PROCESS?    -----> '
       READ(*,'(A20)')CHRANSWER
       PROCNAMES(INDEX1)=CHRANSWER
C
3510   WRITE(*,'(A,\)')' NO. OF DISTRIBUTION TO USE?              -----> '
       READ(*,'(I2)')IANSWER
       INDEX2=IANSWER*2
       INDEX2=INDEX2-1
       IF(IANSWER.LT.1.OR.IANSWER.GT.4)THEN
         WRITE(*,3520)
3520     FORMAT(' !!!  CANNOT BE, PLEASE TRY AGAIN  !!!'/)
         GOTO 3510
       ELSEIF(DISARR(INDEX2,1).EQ.0.) THEN
         WRITE(*,3530)
3530     FORMAT(' !!!  DISTRIBUTION NOT ACTIVE, PLEASE TRY AGAIN  !!!'/)
         GOTO 3510
       ELSE
         USERARR(5,INDEX1)=IANSWER
       ENDIF
       IF(INDEX1.EQ.1) GOTO 3660
C
C
       IF (INDEX1.EQ.11) THEN
3531     WRITE(*,3532)
3532     FORMAT(' HOW MUCH INVENTORY IN % GOES ROUTE 1 ?   %%%.%%----->'
     1    ' ',\)
         READ(*,'(F6.2)')F1ANSWER
         WRITE(*,3533)
```

```
3533    FORMAT(' HOW MUCH INVENTORY IN % GOES ROUTE 2 ?   %%%.%%----->'
     1  ' ',\)
        READ(*,'(F6.2)')F2ANSWER
        F3ANSWER=F1ANSWER+F2ANSWER
        IF (F3ANSWER.NE.100) THEN
          WRITE(*,3534)
3534      FORMAT(/,' !!!  CANNOT BE, PLEASE TRY AGAIN  !!!',/)
          GOTO 3531
        ELSE
          USERARR(7,3)=F1ANSWER
          USERARR(7,4)=F2ANSWER
        ENDIF
      ENDIF
C
C
3540  WRITE(*,'(A,\)')' STARTUP-INVENTORY LEVEL?              [1]-----> '
      READ(*,'(F8.1)')F1ANSWER
      IF(F1ANSWER.LT.0) THEN
        WRITE(*,3550)
3550    FORMAT(' !!!  CANNOT BE, PLEASE TRY AGAIN  !!!'/)
        GOTO 3540
      ELSEIF(F1ANSWER.EQ.0)THEN
        INDEX2=XXLEVEL(7)+INDEX1
        XX(INDEX2)=1
      ELSE
        INDEX2=XXLEVEL(7)+INDEX1
        XX(INDEX2)=F1ANSWER
      ENDIF
C
3560  WRITE(*,'(A,\)')' MINNIMUM INFEED INVENTORY LEVEL?     [0]-----> '
      READ(*,'(F8.1)')F1ANSWER
      IF(F1ANSWER.LT.0.OR.F1ANSWER.GT.XX(INDEX2)) THEN
        WRITE(*,3570)
3570    FORMAT(' !!!  CANNOT BE, PLEASE TRY AGAIN  !!!'/)
        GOTO 3560
      ELSE
        INDEX2=XXLEVEL(3)+INDEX1
        XX(INDEX2)=F1ANSWER
      ENDIF
3580  WRITE(*,'(A,\)')' STARTUP-INV.LEVEL AFTER MINIMUM?     [0]-----> '
      READ(*,'(F8.1)')F1ANSWER
      IF(F1ANSWER.LT.0.OR.XX(INDEX2).GT.F1ANSWER) THEN
        WRITE(*,3590)
3590    FORMAT(' !!!  CANNOT BE, PLEASE TRY AGAIN  !!!'/)
        GOTO 3580
      ELSE
        INDEX2=XXLEVEL(5)+INDEX1
        XX(INDEX2)=F1ANSWER
      ENDIF
C
3600  WRITE(*,'(A,\)')' MAXIMUM INFEED INV. LEVEL?    [999999.9]-----> '
      READ(*,'(F8.1)')F1ANSWER
      INDEX3=XXLEVEL(5)+INDEX1
```

```
      IF(F1ANSWR.LT.0) THEN
        WRITE(*,3610)
3610    FORMAT(' !!!  CANNOT BE, PLEASE TRY AGAIN  !!!'/)
        GOTO 3600
      ELSEIF(F1ANSWER.GT.0.AND.XX(INDEX3).GE.F1ANSWER) THEN
        WRITE(*,3620)
3620    FORMAT(' !!!  CANNOT BE, PLEASE TRY AGAIN  !!!'/)
        GOTO 3600
      ELSEIF(F1ANSWER.EQ.0)THEN
        F3ANSWER=999999.9
        INDEX2=XXLEVEL(4)+INDEX1
        XX(INDEX2)=999999.9
      ELSE
        F3ANSWER=F1ANSWER
        INDEX2=XXLEVEL(4)+INDEX1
        XX(INDEX2)=F1ANSWER
      ENDIF
C
3630  WRITE(*,'(A,\)')' STARTUP-INV.LEVEL AFTER MAXIMUM? [999999.9]--> '
      READ(*,'(F8.1)')F1ANSWER
      INDEX3=XXLEVEL(3)+INDEX1
      IF(F1ANSWER.LT.0.OR.F1ANSWER.GT.F3ANSWER) THEN
        WRITE(*,3640)
3640    FORMAT(' !!!  CANNOT BE, PLEASE TRY AGAIN  !!!'/)
        GOTO 3630
      ELSEIF(F1ANSWER.GT.0.AND.F1ANSWER.LT.XX(INDEX3)) THEN
        WRITE(*,3650)
3650     FORMAT(' !!!  CANNOT BE, PLEASE TRY AGAIN  !!!'/)
        GOTO 3630
      ELSEIF(F1ANSWER.EQ.0)THEN
          INDEX2=XXLEVEL(6)+INDEX1
          XX(INDEX2)=999999.9
      ELSE
          INDEX2=XXLEVEL(6)+INDEX1
          XX(INDEX2)=F1ANSWER
      ENDIF
3660  CONTINUE
C
3670  WRITE(*,3680)
3680  FORMAT(/' WHAT LOADER DO YOU WANT TO USE (32-36) ?    [0]----->'
     1  ' ',\)
      READ(*,'(I2)')IANSWER
      IF (IANSWER.EQ.0) THEN
        USERARR(4,INDEX1)=IANSWER
      ELSEIF (IANSWER.GE.32.AND.IANSWER.LE.36) THEN
        IF(USERARR(6,IANSWER).LE.0)THEN
          WRITE(*,3685)
3685      FORMAT(' !!!  CANNOT BE, PLEASE TRY AGAIN  !!!'/)
          GOTO 3670
        ELSE
          USERARR(4,INDEX1)=IANSWER
        ENDIF
```

```
      ELSE
        WRITE(*,3690)
3690    FORMAT(' !!!  CANNOT BE, PLEASE TRY AGAIN  !!!'/)
        GOTO 3670
      ENDIF
C
3700  WRITE(*,3710)
3710  FORMAT(' BUILD-IN MODEL=0  OR  USERFUNCTION=1 ?      [0]----->',
     1  ' ',\)
      READ(*,'(I2)')IANSWER
      IF(IANSWER.GT.1.OR.IANSWER.LT.0) THEN
        WRITE(*,3720)
3720    FORMAT(' !!!  CANNOT BE, PLEASE TRY AGAIN  !!!'/)
        GOTO 3700
      ELSE
        USERARR(9,INDEX1)=IANSWER
      ENDIF
C
      GOTO 3100
C
C
C
C
C
C
C
C  EDITING MACHINE PARAMETERS
C  ========================================================================
C
C
C
4000  WRITE(*,4010)
4010  FORMAT(/////////////////////////////,
     1    2X,'EDITING MACHINE PARAMETERS:',/,
     2    2X,'---------------------------',/)
      WRITE(*,4020)
4020  FORMAT(2X,'PLEASE ENTER THE NUMBER OF THE MACHINE YOU',/
     1       2X,'WANT TO EDIT. IF THE MACHINE HAS NOT BEEN',/
     2       2X,'SET ACTIVE PREVIOUSLY  YOU CAN ACTIVATE'/,
     3       2X,'IT NOW BY SPECIFYING THE INITIAL NUMBER OF',/
     4       2X,'MACHINES GREATER THAN 0.'/,
     5       2X,'1-42 = MACHINE NUMBER',/
     6       2X,' 0   = RETURN TO MODIFY MENU'//,
     7       2X,'PLEASE ENTER CHOICE    ----> ',\)
4030   READ(*,'(I2)')IANSWER
C
       IF(IANSWER.LT.0.OR.IANSWER.GT.42)THEN
         WRITE(*,4040)
4040     FORMAT(/,2X,'!!!  CANNOT BE, PLEASE TRY AGAIN  !!!',/,
     1  2X'PLEASE ENTER CHOICE  ----> '\)
         GOTO 4030
       ELSEIF(IANSWER.EQ.0)THEN
         GOTO 200
```

```
      ELSE
        INDEX1=IANSWER
      ENDIF
C
C
C
4042   IF(INDEX1.GE.1.AND.INDEX1.LE.4)THEN
         INDEX9=1
       ELSEIF(INDEX1.GE.5.AND.INDEX1.LE.7)THEN
         INDEX9=2
       ELSEIF(INDEX1.GE.7.AND.INDEX1.LE.10)THEN
         INDEX9=3
       ELSEIF(INDEX1.GE.11.AND.INDEX1.LE.13)THEN
         INDEX9=4
       ELSEIF(INDEX1.GE.14.AND.INDEX1.LE.16)THEN
         INDEX9=5
       ELSEIF(INDEX1.GE.17.AND.INDEX1.LE.19)THEN
         INDEX9=6
       ELSEIF(INDEX1.GE.20.AND.INDEX1.LE.22)THEN
         INDEX9=7
       ELSEIF(INDEX1.GE.23.AND.INDEX1.LE.25)THEN
         INDEX9=8
       ELSEIF(INDEX1.GE.26.AND.INDEX1.LE.28)THEN
         INDEX9=9
       ELSEIF(INDEX1.GE.29.AND.INDEX1.LE.31)THEN
         INDEX9=10
       ELSEIF(INDEX1.GE.32.AND.INDEX1.LE.36)THEN
         INDEX9=14
       ELSEIF(INDEX1.GE.37.AND.INDEX1.LE.40)THEN
         INDEX9=12
       ELSEIF(INDEX1.GE.41.AND.INDEX1.LE.42)THEN
         INDEX9=13
       ELSE
         CONTINUE
       ENDIF
C
C
C
4050   IF(INDEX9.NE.14)THEN
         WRITE(*,4060)INDEX9,PROCNAMES(INDEX9)
4060     FORMAT(////////////////////////,
     1     2X'PROCESS NO.'I2': ',A,/
     2     2X'==============')
       ELSEIF(INDEX9.EQ.14) THEN
         WRITE(*,4062)
4062     FORMAT(////////////////////////,
     1     2X'LOADING DEVICES',/
     2     2X'===============')
       ELSE
         CONTINUE
       ENDIF
C
```

```
      WRITE(*,4065)INDEX1,MCHNAMES(INDEX1)
4065  FORMAT(2X'MACHINE TYPE 'I2': 'A,/,
     1       2X'----------------'/)
C
      WRITE(*,4078)USERARR(6,INDEX1)
4078  FORMAT('  INITIAL NUMBER OF MACHINES        :'F6.0)
      IF(INDEX1.EQ.40.OR.INDEX1.EQ.42)GOTO 4090
      IF(INDEX1.EQ.39) GOTO 4081
      WRITE(*,4080)USERARR(1,INDEX1)
4080  FORMAT('  AVERAGE PROCESSING TIME / TREE    :   ',F8.4)
4081  WRITE(*,4082)USERARR(2,INDEX1),
     1      USERARR(3,INDEX1),USERARR(8,INDEX1)
4082  FORMAT('  FIXED CONSTANT TIME / LOAD        :   ',F8.4,/
     1       '  FIXED CONST. TIME / ONE WAY HAUL  :   ',F8.4,/
     2       '  MACHINE CAPACITY IN CU.FT         : ',F8.2)
4090  WRITE(*,4100)USERARR(21,INDEX1),USERARR(22,INDEX1)
4100  FORMAT('  FIXED COST / SCHEDULED HOUR       : ',F8.2,/
     1       '  VARIABLE COST/ MACHINE HOUR       : ',F8.2)
      WRITE(*,4110)
4110  FORMAT(/,2X,'CONTINUE EDITING? Y/N   [N]----> '\)
      READ(*,'(A)')CHRANSWER
      IF (CHRANSWER.EQ.'Y') THEN
        GOTO 4200
      ELSE
        GOTO 4000
      ENDIF
C
C
C
C
4200    WRITE(*,4220)INDEX1
4220    FORMAT(//,' MACHINE TYPE 'I2' :',/,' ---------------')
        WRITE(*,'(A,\)')' NAME OF MACHINE TYPE ?                ------> '
        READ(*,'(A20)')CHRANSWER
        MCHNAMES(INDEX1)=CHRANSWER
        WRITE(*,'(A,\)')' INITIAL NUMBER OF MACHINES ?          ------> '
        READ(*,'(I4)')IANSWER
        USERARR(6,INDEX1)=IANSWER
        IF(INDEX1.EQ.40.OR.INDEX1.EQ.42) GOTO 4270
        IF(INDEX1.EQ.39) GOTO 4240
4230    WRITE(*,'(A,\)')' AVERAGE PROCESSING TIME / TREE?    [0]------> '
        READ(*,'(F8.4)')F1ANSWER
        IF (F1ANSWER.LT.0) THEN
         WRITE(*,'(A,/)')' !!!  CANNOT BE, PLEASE TRY AGAIN  !!!'
         GOTO 4230
        ELSE
         USERARR(1,INDEX1)=F1ANSWER
        ENDIF
4240    WRITE(*,'(A,\)')' FIXED CONSTANT TIME / LOAD?        [0]------> '
        READ(*,'(F8.4)')F1ANSWER
        IF (F1ANSWER.LT.0) THEN
         WRITE(*,'(A,/)')' !!!  CANNOT BE, PLEASE TRY AGAIN  !!!'
         GOTO 4240
```

```
      ELSE
       USERARR(2,INDEX1)=F1ANSWER
      ENDIF
4250  WRITE(*,'(A,\)')' FIXED CONST. TIME / ONE WAY HAUL? [0]------> '
      READ(*,'(F8.4)')F1ANSWER
      IF (F1ANSWER.LT.0) THEN
       WRITE(*,'(A,/)')' !!!  CANNOT BE, PLEASE TRY AGAIN  !!!'
       GOTO 4250
      ELSE
       USERARR(3,INDEX1)=F1ANSWER
      ENDIF
4260  WRITE(*,'(A,\)')' MACHINE CAPACITY IN CU.FT?        [1]------> '
      READ(*,'(F8.2)')F1ANSWER
      IF (F1ANSWER.LT.0) THEN
       WRITE(*,'(A,/)')' !!!  CANNOT BE, PLEASE TRY AGAIN  !!!'
       GOTO 4260
      ELSEIF(F1ANSWER.EQ.0) THEN
        USERARR(8,INDEX1)=1
      ELSE
        USERARR(8,INDEX1)=F1ANSWER
      ENDIF
4270  WRITE(*,'(A,\)')' FIXED COST / SCHEDULED HOUR?      [0]------> '
      READ(*,'(F8.2)')F1ANSWER
      IF (F1ANSWER.LT.0) THEN
       WRITE(*,'(A,/)')' !!!  CANNOT BE, PLEASE TRY AGAIN  !!!'
       GOTO 4270
      ELSE
       USERARR(21,INDEX1)=F1ANSWER
      ENDIF
4280  WRITE(*,'(A,\)')' VARIABLE COST/ MACHINE HOUR?      [0]------> '
      READ(*,'(F8.2)')F1ANSWER
      IF (F1ANSWER.LT.0) THEN
       WRITE(*,'(A,/)')' !!!  CANNOT BE, PLEASE TRY AGAIN  !!!'
       GOTO 4280
      ELSE
       USERARR(22,INDEX1)=F1ANSWER
      ENDIF
C
C
      GOTO 4050
C
C
C
C
C
C  EDITING THE MACHINE BREAKDOWN PARAMETERS
C  ========================================
C
5000 WRITE(*,5010)
5010 FORMAT(/////////////////////////////////////,2X,
    1     'EDITING MACHINE BREAKDOWN PARAMETERS:',/,
    2  2X,'-------------------------------------',/)
     WRITE(*,5020)
```

```
5020 FORMAT(2X,'PLEASE ENTER THE NUMBER OF THE MACHINE FOR',/
    1       2X,'WHICH YOU WANT TO EDIT THE MACHINE BREAK-',/
    2       2X,'DOWN PARAMETERS.'/,
    3       2X,'1-42 = MACHINE NUMBER',//
    4       2X,' 0   = RETURN TO MODIFY MENU'/,
    5       2X'PLEASE ENTER CHOICE   ----> ',\)
5030  READ(*,'(I2)')IANSWER
C
      IF(IANSWER.LT.0.OR.IANSWER.GT.42)THEN
        WRITE(*,5040)
5040    FORMAT(/,2X,'!!! CANNOT BE, PLEASE TRY AGAIN !!!',/,
    1   2X'PLEASE ENTER CHOICE  ----> '\)
        GOTO 5030
      ELSEIF(IANSWER.EQ.0)THEN
        GOTO 200
      ELSE
        INDEX1=IANSWER
      ENDIF
C
C
C
5100  WRITE(*,5110)INDEX1,MCHNAMES(INDEX1)
5110  FORMAT(/////////////////////////////////////////,
    1 '  FREQUENCY DISTRIBUTIONS MACHINE TYPE 'I2' : 'A,/
    3 '   CLASS       CUM FREQ.%    TIME BETW.FAILURE    CUM.FREQ.%',
    4 '     REPAIR TIME'/,
    5 ' -------------------------------------------------------------',
    6 '-----------------')
      IF(MCHARR(INDEX1,1,1).EQ.0.)THEN
        WRITE(*,5120)
5120    FORMAT(/,20X,'****** DISTRIBUTION NOT USED  ******')
      ELSE
        DO 5170 INDEX2=1,10,1
          IF(MCHARR(INDEX1,1,INDEX2).EQ.0.AND.
    1       MCHARR(INDEX1,3,INDEX2).EQ.0) THEN
            GOTO 5160
          ELSEIF (MCHARR(INDEX1,1,INDEX2).GT.0.AND.
    1      MCHARR(INDEX1,3,INDEX2).EQ.0) THEN
           WRITE(*,5130)INDEX2,MCHARR(INDEX1,1,INDEX2),
    1      MCHARR(INDEX1,2,INDEX2)
5130       FORMAT(4X,I2,10X,F8.2,10X,F8.2)
          ELSEIF (MCHARR(INDEX1,1,INDEX2).EQ.0.AND.
    1      MCHARR(INDEX1,3,INDEX2).GT.0) THEN
           WRITE(*,5140)INDEX2,MCHARR(INDEX1,3,INDEX2),
    1      MCHARR(INDEX1,4,INDEX2)
5140       FORMAT(4X,I2,10X,8X,10X,8X,10X,F8.2,10X,F8.2)
          ELSE
           WRITE(*,5150)INDEX2,MCHARR(INDEX1,1,INDEX2),
    1      MCHARR(INDEX1,2,INDEX2),MCHARR(INDEX1,3,INDEX2),
    2      MCHARR(INDEX1,4,INDEX2)
5150       FORMAT(4X,I2,10X,F8.2,10X,F8.2,10X,F8.2,10X,F8.2)
          ENDIF
5160       CONTINUE
```

```
5170       CONTINUE
      ENDIF
C
      WRITE(*,5180)
5180  FORMAT(/,2X,'CONTINUE EDITING? Y/N   [N]----> '\)
      READ(*,'(A)')CHRANSWER
      IF (CHRANSWER.EQ.'Y') THEN
        GOTO 5182
      ELSE
        GOTO 5000
      ENDIF
C
C
C
5182   DO 5185 IND10=1,4,1
         DO 5183 IND11=1,10,1
           MCHARR(INDEX1,IND10,IND11)=0
5183     CONTINUE
5185   CONTINUE
C
       F3ANSWER=0
5190   WRITE(*,5200)INDEX1,MCHNAMES(INDEX1)
5200   FORMAT(//////////////////////////////,
     1 '  MACHINE TYPE ',I2,' : ',A,/
     2 '  =================',//,
     3 '  FREQUENCY DISTRIBUTION FOR TIMES BETWEEN FAILURES:'/,
     4 '  ---------------------------------------------------',/)
5210   DO 5280 INDEX2=1,10,1
5220     WRITE(*,5230)INDEX2
5230     FORMAT(/,' CLASS ',I2,': CUM.REL.FREQENCY?       [0]----->'\)
         READ(*,'(BN,F8.2)')F1ANSWER
         WRITE(*,5240)INDEX2
5240     FORMAT(' CLASS ',I2,': TIME BETWEEN FAILURES?  [0]----->'\)
         READ(*,'(BN,F8.2)')F2ANSWER
           IF(INDEX2.EQ.1.AND.F1ANSWER.EQ.0)THEN
            GOTO 5400
           ELSEIF(F2ANSWER.LE.0)THEN
            WRITE(*,'(A,/)')' !!  CANNOT BE, PLEASE TRY AGAIN  !!'
            GOTO 5220
           ELSEIF(F1ANSWER.GT.100)THEN
            WRITE(*,'(A,/)')' !!  CANNOT BE, PLEASE TRY AGAIN  !!'
            GOTO 5220
           ELSEIF(INDEX2.GT.1.AND.F1ANSWER.LE.F3ANSWER)THEN
            WRITE(*,'(A,/)')' !!  CANNOT BE, PLEASE TRY AGAIN  !!'
            GOTO 5220
           ELSEIF(INDEX2.EQ.10.AND.F1ANSWER.NE.100)THEN
            WRITE(*,'(A,/)')' !!  CANNOT BE, PLEASE TRY AGAIN  !!'
            GOTO 5220
           ELSE
            F3ANSWER=F1ANSWER
            MCHARR(INDEX1,1,INDEX2)=F1ANSWER
            MCHARR(INDEX1,2,INDEX2)=F2ANSWER
            IF(F1ANSWER.EQ.100)GOTO 5290
```

```
          ENDIF
5280  CONTINUE
C
C
5290  F3ANSWER=0
      WRITE(*,5300)INDEX1,MCHNAMES(INDEX1)
5300  FORMAT(//////////////////////////////,
     1 '  MACHINE TYPE 'I2' : 'A,/
     2 '  ==================',//
     3 '  FREQUENCY DISTRIBUTION FOR MACHINE REPAIR TIMES:'/,
     4 '  -----------------------------------------------'/)
5310  DO 5380 INDEX2=1,10,1
5320    WRITE(*,5330)INDEX2
5330    FORMAT(/,' CLASS ',I2,': CUM.REL.FREQENCY?       [0]----->'\)
        READ(*,'(BN,F8.2)')F1ANSWER
        WRITE(*,5340)INDEX2
5340    FORMAT(' CLASS ',I2,': MACHINE REPAIR TIME?    [0]----->'\)
        READ(*,'(BN,F8.2)')F2ANSWER
        IF(INDEX2.EQ.1.AND.F1ANSWER.EQ.0)THEN
           WRITE(*,'(A,/)')' !!  CANNOT BE, PLEASE TRY AGAIN  !!'
           GOTO 5320
        ELSEIF(F2ANSWER.LE.0)THEN
           WRITE(*,'(A,/)')' !!  CANNOT BE, PLEASE TRY AGAIN  !!'
           GOTO 5320
        ELSEIF(F1ANSWER.GT.100)THEN
           WRITE(*,'(A,/)')' !!  CANNOT BE, PLEASE TRY AGAIN  !!'
           GOTO 5320
        ELSEIF(INDEX2.GT.1.AND.F1ANSWER.LE.F3ANSWER)THEN
           WRITE(*,'(A,/)')' !!  CANNOT BE, PLEASE TRY AGAIN  !!'
           GOTO 5320
        ELSEIF(INDEX2.EQ.10.AND.F1ANSWER.NE.100)THEN
           WRITE(*,'(A,/)')' !!  CANNOT BE, PLEASE TRY AGAIN  !!'
           GOTO 5320
        ELSE
           F3ANSWER=F1ANSWER
           MCHARR(INDEX1,3,INDEX2)=F1ANSWER
           MCHARR(INDEX1,4,INDEX2)=F2ANSWER
           IF(F1ANSWER.EQ.100)GOTO 5400
        ENDIF
5380  CONTINUE
C
C
5400  CONTINUE
C
      GOTO 5100
C
C
C
C
C
C
```

```
C   SAVING MODEL ON DISK
C   ===================
C
C
6000   WRITE(*,6002)
6002   FORMAT(////////////////////////////////,20X,
     1'END OF SUBROUTINE MODIFY'/20X
     2'========================'//,
     3'  YOU HAVE TO SAVE THE EDITED MODEL ON DISK,'/,
     4'  OTHERWISE ALL YOUR WORK WILL BE LOST !!'//)
C
6004  WRITE(*,'(/A\)')'  SAVE MODEL ON DISK Y/N ?       [Y]------> '
      READ(*,'(A1)')CHRANSWER
C
      IF (CHRANSWER.EQ.'N') THEN
        WRITE(*,'(/,A,\)')'  ARE YOU REALY SURE Y/N ?       [N]------> '
        READ(*,'(A1)')CHRANSWER
        IF (CHRANSWER.EQ.'Y') THEN
          GOTO 9998
        ELSE
          GOTO 6004
        ENDIF
      ELSE
        CONTINUE
      ENDIF
C
6010  INQUIRE(FILE=FILENAME,EXIST=FILESTATUS)
      IF(.NOT.FILESTATUS) THEN
        OPEN(10,FILE=FILENAME,STATUS='NEW')
      ELSE
        WRITE(*,6012)FILENAME
6012    FORMAT(/,'  !!!!  FILE: 'A' ALREADY EXISTS !!!!'//,
     1  '  OVERWRITE OLD FILE?            [N]------> ',\)
        READ(*,'(A1)')CHRANSWR
        IF(CHRANSWER.EQ.'Y')THEN
          OPEN(10,FILE=FILENAME,STATUS='OLD')
          REWIND 10
        ELSE
          WRITE(*,6013)
6013      FORMAT(/,'  INPUT NEW FILENAME FOR MODEL:       -----> ',\)
          READ(*,'(A20)')FILENAME
          GOTO 6010
        ENDIF
      ENDIF
C
C
      WRITE(10,'(F8.1)') XX(1)
      WRITE(10,'(F8.1)') XX(2)
      WRITE(10,'(F8.1)') XX(3)
      WRITE(10,'(F8.0)') XX(4)
      WRITE(10,'(F8.1)') XX(5)
      WRITE(10,'(F8.1)') XX(6)
      WRITE(10,'(F8.1)') XX(7)
```

```
      WRITE(10,'(F8.1)') XX(8)
      WRITE(10,'(F8.1)') XX(9)
      WRITE(10,'(F8.4)') XX(10)
      DO 6022 INDEX1=11,100,1
        WRITE(10,6020) XX(INDEX1)
6020    FORMAT(F8.1)
6022  CONTINUE
C
C
      DO 6028 INDEX1=1,3,1
      DO 6026 INDEX2=1,42,1
        WRITE(10,6024) USERARR(INDEX1,INDEX2)
6024    FORMAT(F8.4)
6026   CONTINUE
6028  CONTINUE
C
      DO 6038 INDEX1=4,26,1
      DO 6036 INDEX2=1,42,1
        WRITE(10,6034) USERARR(INDEX1,INDEX2)
6034    FORMAT(F8.2)
6036   CONTINUE
6038  CONTINUE
C
C
C
      DO 6044 INDEX1=1,8,1
      DO 6042 INDEX2=1,10,1
        WRITE(10,6040) DISARR(INDEX1,INDEX2)
6040    FORMAT(F8.2)
6042   CONTINUE
6044   CONTINUE

C
      DO 6052 INDEX1=1,42,1
      DO 6050 INDEX2=1,4,1
      DO 6048 INDEX3=1,10,1
        WRITE(10,6046) MCHARR(INDEX1,INDEX2,INDEX3)
6046    FORMAT(F8.2)
6048   CONTINUE
6050   CONTINUE
6052   CONTINUE
C
      DO 6056 INDEX1=1,52,1
        WRITE(10,6054) MCHNAMES(INDEX1)
6054    FORMAT(A)
6056   CONTINUE
C
      DO 6060 INDEX1=1,20,1
        WRITE(10,6058) PROCNAMES(INDEX1)
6058    FORMAT(A)
6060  CONTINUE
C
```

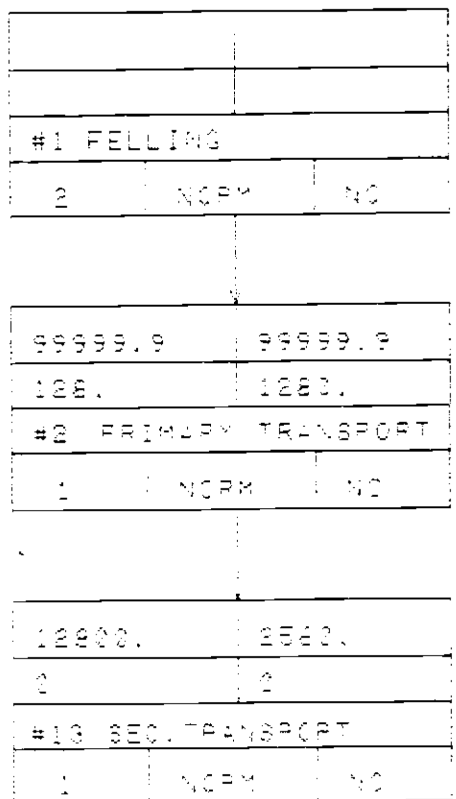
```
      DO 6064 INDEX1=1,4,1
        WRITE(10,6062) DISTRIBNAMES(INDEX1)
6062    FORMAT(A)
6064  CONTINUE
C
      REWIND 10
      CLOSE(10,STATUS='KEEP')
      WRITE(*,6066)
6066  FORMAT(///,20X,'!!!!  MODEL HAS BEEN SAVED  !!!!'/,
     120X,'    PRESS RETURN TO CONTINUE')
      READ(*,'(I2)')IANSWER
      GOTO 200
C
C
C
C END OF SUBROUTINE:
C =================
C
9998  RETURN
      END
```

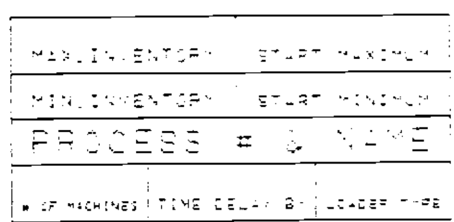
APPENDIX E

TABLE OF CONTENTS:

APPENDIX E

1. Figure, Harvesting system 1





APPENDIX E

2. LOGSIM, Harvesting system 1 printout

```
*******************************************************
*                                                     *
*                 >>> L O G S I M <<<                 *
*              HARVESTING CONFIGURATION               *
*                                                     *
*******************************************************
```

SYSTEM1.MOD

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

NAME OF SIMULATION MODEL : SYSTEM1.MOD
AMOUNT TO BE HARVESTED (CU.FT.) : 255664.
TIME DELAY PARAMETER : .0100

MACHINE CONFIGURATION

PROCESS #	IN ORIGIN	OUT DESTINATION
1		2.
2	1.	13.
3		
4		
5		
6		
7		
8		
9		
10		
11		
12		
13	2.	

MATERIAL FREQUENCY DISTRIBUTIONS

CUMULATIVE FREQUENCY DISTRIBUTION NO.1 : WHOLE TREES
CUMULATIVE FREQUENCY DISTRIBUTION NO.2 : SAWLOGS
CUMULATIVE FREQUENCY DISTRIBUTION NO.3 :
CUMULATIVE FREQUENCY DISTRIBUTION NO.4 :

	DISTRIBUTION 1		DISTRIBUTION 2		DISTRIBUTION 3		DISTRIBUTION 4	
CLASS	FREQ.%	CU.FT	FREQ.%	CU.FT	FREQ.%	CU.FT	FREQ.%	CU.FT
1	11.60	4.40	20.00	3.70				
2	29.70	9.40	50.30	9.30				
3	50.40	18.00	67.30	15.30				
4	69.90	28.30	80.80	21.50				
5	84.40	40.90	91.10	29.20				
6	93.00	54.60	96.10	38.00				
7	97.70	70.20	98.90	48.00				
8	100.00	92.10	100.00	59.40				

INVENTORY AND BUFFER SIZES

PRO-CESS	NAME	DISTRI-BUTION	STARTUP INV.	MINIMUM INV.	STARTUP MINIMUM	MAXIMUM INV.	STARTUP MAXIMUM
1	MANUAL FELLING	1.					
2	CABLE SKIDDING	2.	1280.0	128.0	1280.0	999999.9	999999.9
13	FINAL TRANSPORT	2.	1184.0	.0	.0	12800.0	2560.0

PROCESS NO. 1: MANUAL FELLING

OUTGOING DESTINATION : PROCESS NO. 2. CABLE SKIDDING
DISTRIBUTION USED : 1. WHOLE TREES
LOADER USED : NONE
TIME DELAYS HANDELED BY : STANDARD BUILD-IN FUNCTIONS

MACHINES USED :

TYPE	NAME	INITIAL # OF MACHINES
1	HAND FELLERS	2.

PROCESS NO. 2: CABLE SKIDDING
=============

```
INCOMING ORIGIN          : PROCESS NO. 1.  MANUAL FELLING
OUTGOING DESTINATION     : PROCESS NO.13.  FINAL TRANSPORT
DISTRIBUTION USED        :      2.          SAWLOGS
STARTUP-INVENTORY LEVEL  :   1280.0
MINIMUM INVENTORY LEVEL  :    128.0
STARTUP LEVEL MINIMUM    :   1280.0
MAXIMUM INVENTORY LEVEL  : 999999.9
STARTUP LEVEL MAXIMUM    : 999999.9
LOADER USED              : NONE
TIME DELAYS HANDELED BY  : STANDARD BUILD-IN FUNCTIONS

MACHINES USED            :
TYPE   NAME                     INITIAL # OF MACHINES
-----------------------------------------------------
  5    CAT 528                          1.
```

PROCESS NO.13: FINAL TRANSPORT
=============

```
INCOMING ORIGIN          : PROCESS NO. 2.  CABLE SKIDDING
DISTRIBUTION USED        :      2.          SAWLOGS
STARTUP-INVENTORY LEVEL  :   1184.0
MINIMUM INVENTORY LEVEL  :       .0
STARTUP LEVEL MINIMUM    :       .0
MAXIMUM INVENTORY LEVEL  :  12800.0
STARTUP LEVEL MAXIMUM    :   2560.0
LOADER USED              : NONE
TIME DELAYS HANDELED BY  : STANDARD BUILD-IN FUNCTIONS

MACHINES USED            :
TYPE   NAME                     INITIAL # OF MACHINES
-----------------------------------------------------
 41    LOG TRUCK                        1.
```

MACHINE TYPE 1: HAND FELLERS
===============

```
INITIAL NUMBER OF MACHINES       :   2.
AVERAGE PROCESSING TIME / TREE   :      .1500
```

FIXED CONSTANT TIME / LOAD : .0000
FIXED CONST. TIME / ONE WAY HAUL : .0000
FIXED COST / SCHEDULED HOUR : .53
VARIABLE COST/ MACHINE HOUR : 21.50
MACHINE CAPACITY IN CU.FT : 99999.00

FREQUENCY DISTRIBUTIONS MACHINE TYPE 1 : HAND FELLERS

CLASS	CUM FREQ.%	TIME BETW.FAILURE	CUM.FREQ.%	REPAIR TIME
1	20.00	3.00	30.00	1.00
2	40.00	7.00	55.00	3.00
3	60.00	12.00	75.00	7.00
4	80.00	22.00	90.00	11.00
5	100.00	44.00	100.00	16.00

MACHINE TYPE 5: CAT 528

INITIAL NUMBER OF MACHINES : 1.
AVERAGE PROCESSING TIME / TREE : .1500
FIXED CONSTANT TIME / LOAD : .1500
FIXED CONST. TIME / ONE WAY HAUL : .0000
FIXED COST / SCHEDULED HOUR : 29.44
VARIABLE COST/ MACHINE HOUR : 35.72
MACHINE CAPACITY IN CU.FT : 128.00

FREQUENCY DISTRIBUTIONS MACHINE TYPE 5 : CAT 528

CLASS	CUM FREQ.%	TIME BETW.FAILURE	CUM.FREQ.%	REPAIR TIME
1	20.00	7.00	35.00	.50
2	40.00	14.00	65.00	1.00
3	60.00	28.00	85.00	3.00
4	80.00	56.00	95.00	8.00
5	100.00	100.00	100.00	16.00

MACHINE TYPE 41: LOG TRUCK

INITIAL NUMBER OF MACHINES : 1.
AVERAGE PROCESSING TIME / TREE : .0000
FIXED CONSTANT TIME / LOAD : .7500
FIXED CONST. TIME / ONE WAY HAUL : 1.0000
FIXED COST / SCHEDULED HOUR : 15.04
VARIABLE COST/ MACHINE HOUR : 37.56
MACHINE CAPACITY IN CU.FT : 431.00

FREQUENCY DISTRIBUTIONS MACHINE TYPE 41 : LOG TRUCK

CLASS	CUM FREQ.%	TIME BETW.FAILURE	CUM.FREQ.%	REPAIR TIME
1	20.00	7.00	35.00	.50
2	40.00	14.00	65.00	1.00
3	60.00	28.00	85.00	3.00
4	80.00	56.00	95.00	8.00
5	100.00	100.00	100.00	16.00

APPENDIX E

3. LOGSIM, simulation results system 1

```
*****************************************************
*                                                   *
*                 >>> L O G S I M <<<               *
*                 SIMULATION  RESULTS               *
*                                                   *
*****************************************************
```

SIMULATION MODEL USED: SYSTEM1.MOD
****

DATE= Ø6-Ø1-87
TIME= Ø1:ØØ:29
SIMULATION RUN 1 OF 1.

PROCESS NO. 1 : MANUAL FELLING
==============

```
TIME BEGIN OF PROCESS   : .ØØØØØØØE+ØØ  AVERAGE INVENTORY       : .ØØØØØØØE+ØØ
TIME END OF PROCESS     : .898993ØE+Ø3  MAXIMUM INVENTORY       : .ØØØØØØØE+ØØ
DURATION OF PROCESS     : .898993ØE+Ø3  MINIMUM INVENTORY       : .ØØØØØØØE+ØØ
TIME INVENTORY TOO LOW  : .ØØØØØØØE+ØØ  STD.DEV.INVENTORY       : .ØØØØØØØE+ØØ
TIME INVENTORY TOO HIGH : .ØØØØØØØE+ØØ  # OF OBSERVATIONS INV.  : .ØØØØØØØE+ØØ
% INVENTORY DOWNTIME    : .ØØØØØØØE+ØØ  SUM UNITS PROCESSED     : .255664ØE+Ø6
TOTAL # OF MACHINES     :            2  SUM COST OF PROCESS     : .3Ø68694E+Ø5
SUM SCHEDULED HOURS     : .1797986E+Ø4  COST PER UNIT           : .12ØØ284E+ØØ
SUM MACH.BREAKDOWN HOURS: .415ØØØØE+Ø3  COST PER SCHEDULED HOUR : .17Ø674ØE+Ø2
SUM PRODUCTIVE HOURS    : .1382977E+Ø4
% NET UTILIZATION MACH. : .7691812E+Ø2
% GROSS UTILIZATION MACH: .999995ØE+Ø2
```

MACHINE TYPE 1 : HAND FELLERS

```
TOTAL # OF MACHINES     :            2  COST PER MACHINE        : .1534347E+Ø5
SUM SCHEDULED HOURS     : .1797986E+Ø4  COST PER SCHEDULED HOUR : .17Ø674ØE+Ø2
SUM MACH.BREAKDOWN HOURS: .415ØØØØE+Ø3  % NET UTILIZATION MACH. : .7691812E+Ø2
SUM PRODUCTIVE HOURS    : .1382977E+Ø4  % GROSS UTILIZATION MACH: .999995ØE+Ø2
```

PROCESS NO. 2 : CABLE SKIDDING

TIME BEGIN OF PROCESS	: .3000000E+01	AVERAGE INVENTORY	: .8920633E+05
TIME END OF PROCESS	: .3112599E+04	MAXIMUM INVENTORY	: .1829480E+06
DURATION OF PROCESS	: .3109599E+04	MINIMUM INVENTORY	: .0000000E+00
TIME INVENTORY TOO LOW	: .0000000E+00	STD.DEV.INVENTORY	: .5284388E+05
TIME INVENTORY TOO HIGH	: .0000000E+00	/ OF OBSERVATIONS INV.	: .1143000E+05
% INVENTORY DOWNTIME	: .0000000E+00	SUM UNITS PROCESSED	: .2556599E+06
TOTAL / OF MACHINES	: 1	SUM COST OF PROCESS	: .1954953E+06
SUM SCHEDULED HOURS	: .3109599E+04	COST PER UNIT	: .7646695E+00
SUM MACH.BREAKDOWN HOURS	: .1995000E+03	COST PER SCHEDULED HOUR	: .6286834E+02
SUM PRODUCTIVE HOURS	: .2910099E+04		
% NET UTILIZATION MACH.	: .9358438E+02		
% GROSS UTILIZATION MACH	: .1000000E+03		

MACHINE TYPE 5 : CAT 528

TOTAL / OF MACHINES	: 1	COST PER MACHINE	: .1954953E+06
SUM SCHEDULED HOURS	: .3109599E+04	COST PER SCHEDULED HOUR	: .6286834E+02
SUM MACH.BREAKDOWN HOURS	: .1995000E+03	% NET UTILIZATION MACH.	: .9358438E+02
SUM PRODUCTIVE HOURS	: .2910099E+04	% GROSS UTILIZATION MACH	: .1000000E+03

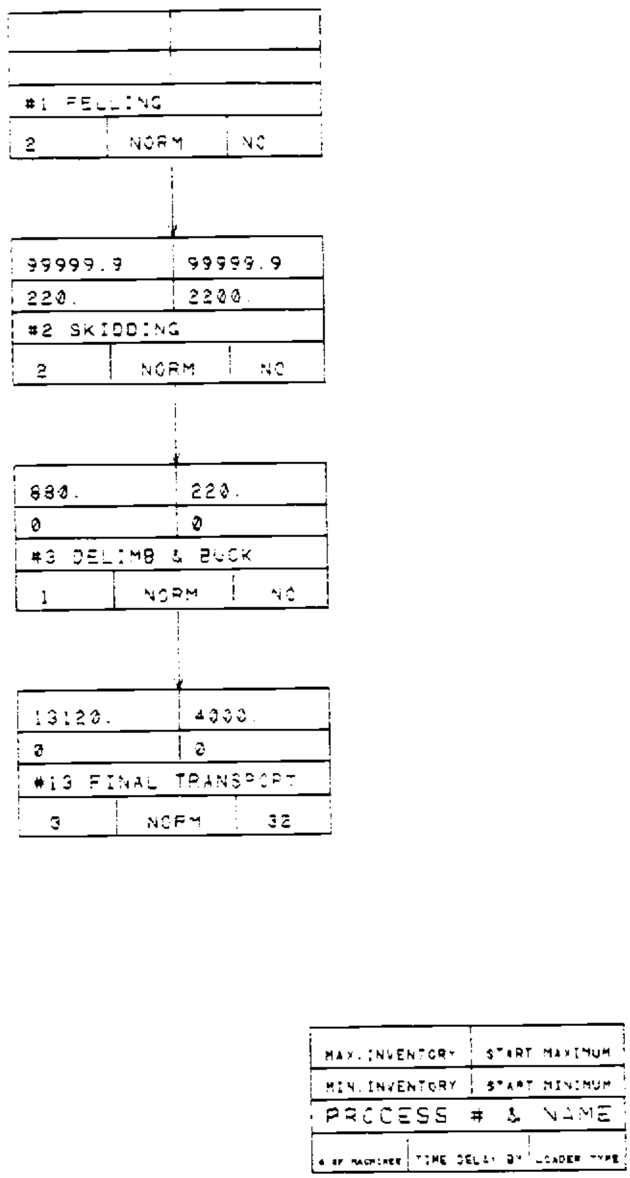
PROCESS NO.13 : FINAL TRANSPORT

TIME BEGIN OF PROCESS	: .1835000E+02	AVERAGE INVENTORY	: .2966347E+03
TIME END OF PROCESS	: .3116751E+04	MAXIMUM INVENTORY	: .1284900E+04
DURATION OF PROCESS	: .3098401E+04	MINIMUM INVENTORY	: .0000000E+00
TIME INVENTORY TOO LOW	: .0000000E+00	STD.DEV.INVENTORY	: .1736150E+03
TIME INVENTORY TOO HIGH	: .0000000E+00	/ OF OBSERVATIONS INV.	: .2822000E+04
% INVENTORY DOWNTIME	: .0000000E+00	SUM UNITS PROCESSED	: .2556600E+06
TOTAL / OF MACHINES	: 1	SUM COST OF PROCESS	: .1097101E+06
SUM SCHEDULED HOURS	: .3098401E+04	COST PER UNIT	: .4291252E+00
SUM MACH.BREAKDOWN HOURS	: .1450000E+03	COST PER SCHEDULED HOUR	: .3540863E+02
SUM PRODUCTIVE HOURS	: .1680250E+04		
% NET UTILIZATION MACH.	: .5422958E+02		
% GROSS UTILIZATION MACH	: .5890942E+02		

```
MACHINE TYPE 41 : LOG TRUCK
-----------------

TOTAL # OF MACHINES      :             1  COST PER MACHINE         : .1097101E+06
SUM SCHEDULED HOURS      : .3098401E+04  COST PER SCHEDULED HOUR : .3540863E+02
SUM MACH.BREAKDOWN HOURS: .1450000E+03  % NET UTILIZATION MACH. : .5422958E+02
SUM PRODUCTIVE HOURS     : .1680250E+04  % GROSS UTILIZATION MACH: .5890942E+02



        COMPLETE HARVESTING SYSTEM STATISTICS
        ====================================


COMPUTER TIME START SIMULATION   DATE: 06-01-87     TIME: 00:31:38
COMPUTER TIME END SIMULATION     DATE: 06-01-87     TIME: 01:44:57

SIMULATION RUN  1 OF  1.
BEGIN OF HARVESTING      : .0000000E+00  END OF HARVESTING        : .3116751E+04
TOTAL # OF MACHINES      :             4  SUM OF UNITS HARVESTED   : .2556640E+06
SUM SCHEDULED HOURS      : .8005986E+04  SUM COST OF SYSTEM       : .3358924E+06
SUM MACH.BREAKDOWN HOURS: .7595000E+03  COST PER UNIT            : .1313804E+01
SUM PRODUCTIVE HOURS     : .5973326E+04  COST PER SYSTEM HOUR     : .1077700E+03
% NET UTILIZATION MACH.  : .7461074E+02
% GROSS UTILIZATION MACH: .8409739E+02


------------------------- END OF RUN # 1 OF  1.-------------------------
```

APPENDIX E

4. Figure, Harvesting system 2



APPENDIX E

5. LOGSIM, Harvesting system 2 printout

```
*****************************************************
*                                                   *
*                 >>> L O G S I M <<<               *
*              HARVESTING CONFIGURATION             *
*                                                   *
*****************************************************
```

```
                        SYSTEM2.MOD
                    **********************

NAME OF SIMULATION MODEL        : SYSTEM2.MOD
AMOUNT TO BE HARVESTED (CU.FT.) :  255664.
TIME DELAY PARAMETER            :        .0100
```

```
          MACHINE CONFIGURATION
          =====================
```

PROCESS #	IN ORIGIN	OUT DESTINATION
1		2.
2	1.	3.
3	2.	13.
4		
5		
6		
7		
8		
9		
10		
11		
12		
13	3.	

MATERIAL FREQUENCY DISTRIBUTIONS

CUMULATIVE FREQUENCY DISTRIBUTION NO.1 : WHOLE TREES
CUMULATIVE FREQUENCY DISTRIBUTION NO.2 : SAWLOGS
CUMULATIVE FREQUENCY DISTRIBUTION NO.3 :
CUMULATIVE FREQUENCY DISTRIBUTION NO.4 :

	DISTRIBUTION 1		DISTRIBUTION 2		DISTRIBUTION 3		DISTRIBUTION 4	
CLASS	FREQ.%	CU.FT	FREQ.%	CU.FT	FREQ.%	CU.FT	FREQ.%	CU.FT
1	11.60	4.40	20.00	3.70				
2	29.70	9.40	50.30	9.30				
3	50.40	18.00	67.30	15.30				
4	69.90	28.30	80.80	21.50				
5	84.40	40.90	91.10	29.20				
6	93.00	54.60	96.10	38.00				
7	97.70	70.20	98.90	48.00				
8	100.00	92.10	100.00	59.40				

INVENTORY AND BUFFER SIZES

PRO-CESS	NAME	DISTRI-BUTION	STARTUP INV.	MINIMUM INV.	STARTUP MINIMUM	MAXIMUM INV.	STARTUP MAXIMUM
1	FELLING	1.					
2	SKIDDING	1.	2200.0	220.0	2200.0	999999.9	999999.9
3	DELIMBING & BUCKING	1.	1.0	.0	.0	880.0	220.0
13	FINAL TRANSPORT	2.	1.0	.0	.0	13120.0	4000.0

PROCESS NO. 1: FELLING

OUTGOING DESTINATION : PROCESS NO. 2. SKIDDING
DISTRIBUTION USED : 1. WHOLE TREES
LOADER USED : NONE
TIME DELAYS HANDELED BY : STANDARD BUILD-IN FUNCTIONS

MACHINES USED :

TYPE	NAME	INITIAL # OF MACHINES
1	CAT 227 FELLER-BNCH	2.

```
          PROCESS NO. 2: SKIDDING
          =============

INCOMING ORIGIN          :  PROCESS NO. 1.  FELLING
OUTGOING DESTINATION     :  PROCESS NO. 3.  DELIMBING & BUCKING
DISTRIBUTION USED        :       1.          WHOLE TREES
STARTUP-INVENTORY LEVEL  :    2200.0
MINIMUM INVENTORY LEVEL  :     220.0
STARTUP LEVEL MINIMUM    :    2200.0
MAXIMUM INVENTORY LEVEL  :  999999.9
STARTUP LEVEL MAXIMUM    :  999999.9
LOADER USED              :  NONE
TIME DELAYS HANDELED BY  :  STANDARD BUILD-IN FUNCTIONS

MACHINES USED            :
TYPE   NAME                          INITIAL # OF MACHINES
-------------------------------------------------------
  5    CAT 528 GRAB-SKIDDER                  2.



          PROCESS NO. 3: DELIMBING & BUCKING
          =============

INCOMING ORIGIN          :  PROCESS NO. 2.  SKIDDING
OUTGOING DESTINATION     :  PROCESS NO.13.  FINAL TRANSPORT
DISTRIBUTION USED        :       1.          WHOLE TREES
STARTUP-INVENTORY LEVEL  :       1.0
MINIMUM INVENTORY LEVEL  :        .0
STARTUP LEVEL MINIMUM    :        .0
MAXIMUM INVENTORY LEVEL  :     880.0
STARTUP LEVEL MAXIMUM    :     220.0
LOADER USED              :  NONE
TIME DELAYS HANDELED BY  :  STANDARD BUILD-IN FUNCTIONS

MACHINES USED            :
TYPE   NAME                          INITIAL # OF MACHINES
-------------------------------------------------------
  8    HAHN HARVESTER                        1.
```

PROCESS NO.13: FINAL TRANSPORT

INCOMING ORIGIN	:	PROCESS NO. 3. DELIMBING & BUCKING
DISTRIBUTION USED	:	2. SAWLOGS
STARTUP-INVENTORY LEVEL	:	1.0
MINIMUM INVENTORY LEVEL	:	.0
STARTUP LEVEL MINIMUM	:	.0
MAXIMUM INVENTORY LEVEL	:	13120.0
STARTUP LEVEL MAXIMUM	:	4000.0
LOADER USED	:	32. CAT 225 LOG LOADER
TIME DELAYS HANDELED BY	:	STANDARD BUILD-IN FUNCTIONS

MACHINES USED :

TYPE	NAME	INITIAL # OF MACHINES
41	LOG TRUCK	3.

MACHINE TYPE 1: CAT 227 FELLER-BNCH

INITIAL NUMBER OF MACHINES	:	2.
AVERAGE PROCESSING TIME / TREE	:	.0000
FIXED CONSTANT TIME / LOAD	:	.0400
FIXED CONST. TIME / ONE WAY HAUL	:	.0000
FIXED COST / SCHEDULED HOUR	:	41.99
VARIABLE COST/ MACHINE HOUR	:	41.35
MACHINE CAPACITY IN CU.FT	:	82.56

FREQUENCY DISTRIBUTIONS MACHINE TYPE 1 : CAT 227 FELLER-BNCH

CLASS	CUM FREQ.%	TIME BETW.FAILURE	CUM.FREQ.%	REPAIR TIME
1	20.00	6.00	50.00	.50
2	40.00	12.00	70.00	1.00
3	60.00	20.00	80.00	2.00
4	80.00	36.00	90.00	5.00
5	100.00	64.00	100.00	10.00

MACHINE TYPE 5: CAT 528 GRAB-SKIDDER

INITIAL NUMBER OF MACHINES	:	2.
AVERAGE PROCESSING TIME / TREE	:	.0000
FIXED CONSTANT TIME / LOAD	:	.1000
FIXED CONST. TIME / ONE WAY HAUL	:	.0000

FIXED COST / SCHEDULED HOUR : 29.44
VARIABLE COST/ MACHINE HOUR : 36.72
MACHINE CAPACITY IN CU.FT : 220.00

FREQUENCY DISTRIBUTIONS MACHINE TYPE 5 : CAT 528 GRAB-SKIDDER

CLASS	CUM FREQ.%	TIME BETW.FAILURE	CUM.FREQ.%	REPAIR TIME
1	20.00	6.00	50.00	.50
2	40.00	12.00	70.00	1.00
3	60.00	20.00	80.00	2.00
4	80.00	36.00	90.00	5.00
5	100.00	64.00	100.00	10.00

MACHINE TYPE 8: HAHN HARVESTER

INITIAL NUMBER OF MACHINES : 1.
AVERAGE PROCESSING TIME / TREE : .0100
FIXED CONSTANT TIME / LOAD : .0000
FIXED CONST. TIME / ONE WAY HAUL : .0000
FIXED COST / SCHEDULED HOUR : 55.44
VARIABLE COST/ MACHINE HOUR : 39.72
MACHINE CAPACITY IN CU.FT : 99999.00

FREQUENCY DISTRIBUTIONS MACHINE TYPE 8 : HAHN HARVESTER

CLASS	CUM FREQ.%	TIME BETW.FAILURE	CUM.FREQ.%	REPAIR TIME
1	20.00	5.00	30.00	1.00
2	40.00	7.00	55.00	3.00
3	60.00	12.00	75.00	7.00
4	80.00	22.00	90.00	11.00
5	100.00	44.00	100.00	16.00

MACHINE TYPE 32: CAT 225 LOG LOADER

INITIAL NUMBER OF MACHINES : 1.
AVERAGE PROCESSING TIME / TREE : .0000
FIXED CONSTANT TIME / LOAD : .2500
FIXED CONST. TIME / ONE WAY HAUL : .0000
FIXED COST / SCHEDULED HOUR : 41.99
VARIABLE COST/ MACHINE HOUR : 41.35
MACHINE CAPACITY IN CU.FT : 1312.00

FREQUENCY DISTRIBUTIONS MACHINE TYPE 32 : CAT 225 LOG LOADER

CLASS	CUM FREQ.%	TIME BETW.FAILURE	CUM.FREQ.%	REPAIR TIME
1	20.00	7.00	35.00	.50
2	40.00	14.00	65.00	1.00
3	60.00	28.00	85.00	3.00
4	80.00	56.00	95.00	8.00
5	100.00	100.00	100.00	16.00

MACHINE TYPE 41: LOG TRUCK

INITIAL NUMBER OF MACHINES	:	3.
AVERAGE PROCESSING TIME / TREE	:	.0000
FIXED CONSTANT TIME / LOAD	:	.5000
FIXED CONST. TIME / ONE WAY HAUL	:	1.0000
FIXED COST / SCHEDULED HOUR	:	15.04
VARIABLE COST/ MACHINE HOUR	:	36.48
MACHINE CAPACITY IN CU.FT	:	1312.00

FREQUENCY DISTRIBUTIONS MACHINE TYPE 41 : LOG TRUCK

CLASS	CUM FREQ.%	TIME BETW.FAILURE	CUM.FREQ.%	REPAIR TIME
1	20.00	7.00	35.00	.50
2	40.00	14.00	65.00	1.00
3	60.00	28.00	85.00	3.00
4	80.00	56.00	95.00	8.00
5	100.00	100.00	100.00	16.00

APPENDIX E

6. LOGSIM, simulation results system 2

```
*******************************************************
*                                                     *
*                  >>> L O G S I M <<<                *
*                  SIMULATION  RESULTS                *
*                                                     *
*******************************************************
```

SIMULATION MODEL USED: SYSTEM2.MOD
****

DATE= 06-01-87
TIME= 00:20:52
SIMULATION RUN 1 OF 1.

PROCESS NO. 1 : FELLING
****

TIME BEGIN OF PROCESS	: .0000000E+00	AVERAGE INVENTORY	: .0000000E+00
TIME END OF PROCESS	: .9504154E+02	MAXIMUM INVENTORY	: .0000000E+00
DURATION OF PROCESS	: .9504154E+02	MINIMUM INVENTORY	: .0000000E+00
TIME INVENTORY TOO LOW	: .0000000E+00	STD.DEV.INVENTORY	: .0000000E+00
TIME INVENTORY TOO HIGH	: .0000000E+00	# OF OBSERVATIONS INV.	: .0000000E+00
% INVENTORY DOWNTIME	: .0000000E+00	SUM UNITS PROCESSED	: .2556640E+06
TOTAL # OF MACHINES	: 2	SUM COST OF PROCESS	: .1491065E+05
SUM SCHEDULED HOURS	: .1900831E+03	COST PER UNIT	: .5832127E-01
SUM MACH.BREAKDOWN HOURS	: .2250000E+02	COST PER SCHEDULED HOUR	: .7844280E+02
SUM PRODUCTIVE HOURS	: .1675710E+03		
% NET UTILIZATION MACH.	: .8815671E+02		
% GROSS UTILIZATION MACH	: .9999364E+02		

MACHINE TYPE 1 : CAT 227 FELLER-BNCH

TOTAL # OF MACHINES	: 2	COST PER MACHINE	: .7455325E+04
SUM SCHEDULED HOURS	: .1900831E+03	COST PER SCHEDULED HOUR	: .7844280E+02
SUM MACH.BREAKDOWN HOURS	: .2250000E+02	% NET UTILIZATION MACH.	: .8815671E+02
SUM PRODUCTIVE HOURS	: .1675710E+03	% GROSS UTILIZATION MACH	: .9999364E+02

PROCESS NO. 2 : SKIDDING

TIME BEGIN OF PROCESS	.7200000E+00	AVERAGE INVENTORY	.5914057E+05
TIME END OF PROCESS	.1894959E+03	MAXIMUM INVENTORY	.1184574E+06
DURATION OF PROCESS	.1887759E+03	MINIMUM INVENTORY	.0000000E+00
TIME INVENTORY TOO LOW	.0000000E+00	STD.DEV.INVENTORY	.3548504E+05
TIME INVENTORY TOO HIGH	.0000000E+00	# OF OBSERVATIONS INV.	.5473000E+04
% INVENTORY DOWNTIME	.0000000E+00	SUM UNITS PROCESSED	.2556656E+06
TOTAL # OF MACHINES	2	SUM COST OF PROCESS	.1583003E+05
SUM SCHEDULED HOURS	.3775518E+03	COST PER UNIT	.6191691E-01
SUM MACH.BREAKDOWN HOURS	.2050000E+02	COST PER SCHEDULED HOUR	.4192809E+02
SUM PRODUCTIVE HOURS	.1284014E+03		
% NET UTILIZATION MACH.	.3400896E+02		
% GROSS UTILIZATION MACH	.3943867E+02		

MACHINE TYPE 5 : CAT 528 GRAB-SKIDDER

TOTAL # OF MACHINES	2	COST PER MACHINE	.7915013E+04
SUM SCHEDULED HOURS	.3775518E+03	COST PER SCHEDULED HOUR	.4192809E+02
SUM MACH.BREAKDOWN HOURS	.2050000E+02	% NET UTILIZATION MACH.	.3400896E+02
SUM PRODUCTIVE HOURS	.1284014E+03	% GROSS UTILIZATION MACH	.3943867E+02

PROCESS NO. 3 : DELIMBING & BUCKING

TIME BEGIN OF PROCESS	.8200001E+00	AVERAGE INVENTORY	.4350824E+03
TIME END OF PROCESS	.1898650E+03	MAXIMUM INVENTORY	.1273986E+04
DURATION OF PROCESS	.1890450E+03	MINIMUM INVENTORY	.0000000E+00
TIME INVENTORY TOO LOW	.0000000E+00	STD.DEV.INVENTORY	.2677341E+03
TIME INVENTORY TOO HIGH	.1184749E+03	# OF OBSERVATIONS INV.	.1064700E+05
% INVENTORY DOWNTIME	.6267021E+02	SUM UNITS PROCESSED	.2556705E+06
TOTAL # OF MACHINES	1	SUM COST OF PROCESS	.1419920E+05
SUM SCHEDULED HOURS	.1890450E+03	COST PER UNIT	.5553708E-01
SUM MACH.BREAKDOWN HOURS	.1600000E+02	COST PER SCHEDULED HOUR	.7511015E+02
SUM PRODUCTIVE HOURS	.9361892E+02		
% NET UTILIZATION MACH.	.4952204E+02		
% GROSS UTILIZATION MACH	.5798563E+02		

MACHINE TYPE 8 : HAHN HARVESTER

TOTAL # OF MACHINES	: 1	COST PER MACHINE	: .1419920E+05
SUM SCHEDULED HOURS	: .1890450E+03	COST PER SCHEDULED HOUR	: .7511015E+02
SUM MACH.BREAKDOWN HOURS	: .1600000E+02	% NET UTILIZATION MACH.	: .4952204E+02
SUM PRODUCTIVE HOURS	: .9361892E+02	% GROSS UTILIZATION MACH	: .5798563E+02

PROCESS NO.13 : FINAL TRANSPORT
===============

TIME BEGIN OF PROCESS	: .8300000E+00	AVERAGE INVENTORY	: .6858059E+04
TIME END OF PROCESS	: .1961243E+03	MAXIMUM INVENTORY	: .1316800E+05
DURATION OF PROCESS	: .1952943E+03	MINIMUM INVENTORY	: .0000000E+00
TIME INVENTORY TOO LOW	: .0000000E+00	STD.DEV.INVENTORY	: .3418662E+04
TIME INVENTORY TOO HIGH	: .7304746E+02	# OF OBSERVATIONS INV.	: .9560000E+04
% INVENTORY DOWNTIME	: .3740379E+02	SUM UNITS PROCESSED	: .2556655E+06
TOTAL # OF MACHINES	: 3	SUM COST OF PROCESS	: .2855527E+05
SUM SCHEDULED HOURS	: .5858829E+03	COST PER UNIT	: .1116900E+00
SUM MACH.BREAKDOWN HOURS	: .2850000E+02	COST PER SCHEDULED HOUR	: .4873886E+02
SUM PRODUCTIVE HOURS	: .5412167E+03		
% NET UTILIZATION MACH.	: .9237627E+02		
% GROSS UTILIZATION MACH	: .9724072E+02		

MACHINE TYPE 41 : LOG TRUCK

TOTAL # OF MACHINES	: 3	COST PER MACHINE	: .9518422E+04
SUM SCHEDULED HOURS	: .5858829E+03	COST PER SCHEDULED HOUR	: .4873886E+02
SUM MACH.BREAKDOWN HOURS	: .2850000E+02	% NET UTILIZATION MACH.	: .9237627E+02
SUM PRODUCTIVE HOURS	: .5412167E+03	% GROSS UTILIZATION MACH	: .9724072E+02

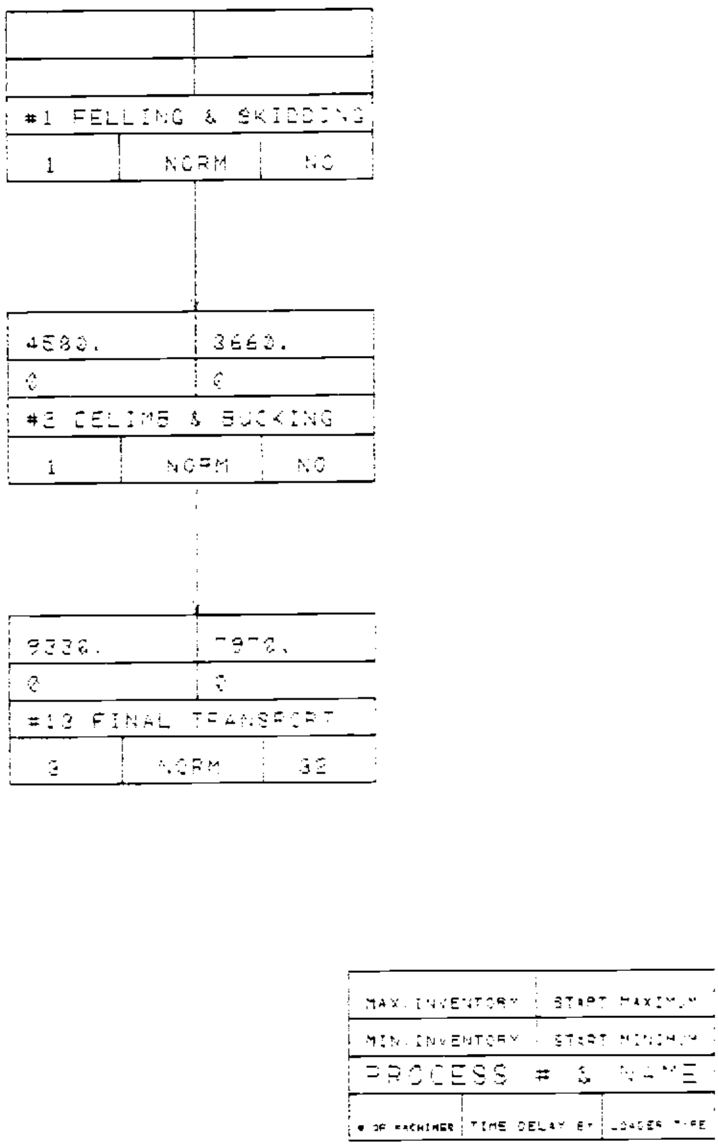
LOADING DEVICES
===============

TOTAL # OF MACHINES	: 1	SUM OF UNITS HARVESTED	: .2556640E+06
SUM SCHEDULED HOURS	: .1961243E+03	SUM COST LOADER DEVICES	: .1021485E+05
SUM MACH.BREAKDOWN HOURS	: .1500000E+01	COST PER UNIT	: .3995419E-01

```
SUM PRODUCTIVE HOURS     : .4871679E+02  COST PER SCHEDULED HOUR : .5208353E+02
% NET UTILIZATION MACH. : .2483975E+02
% GROSS UTILIZATION MACH: .2560457E+02


MACHINE TYPE 32 : CAT 225 LOG LOADER
-----------------

TOTAL # OF MACHINES      :             1  COST PER MACHINE        : .1024970E+05
SUM SCHEDULED HOURS      : .1961243E+03  COST PER SCHEDULED HOUR : .5226124E+02
SUM MACH.BREAKDOWN HOURS: .1500000E+01  % NET UTILIZATION MACH. : .2483975E+02
SUM PRODUCTIVE HOURS     : .4871679E+02  % GROSS UTILIZATION MACH: .2560457E+02



     COMPLETE HARVESTING SYSTEM STATISTICS
     =====================================


COMPUTER TIME START SIMULATION   DATE: 06-01-87     TIME: 00:05:32
COMPUTER TIME END SIMULATION     DATE: 06-01-87     TIME: 00:30:06

SIMULATION RUN  1 OF  1.
BEGIN OF HARVESTING      : .0000000E+00  END OF HARVESTING       : .1961243E+03
TOTAL # OF MACHINES      :             9  SUM OF UNITS HARVESTED  : .2556640E+06
SUM SCHEDULED HOURS      : .1538687E+04  SUM COST OF SYSTEM      : .8370998E+05
SUM MACH.BREAKDOWN HOURS: .8900000E+02  COST PER UNIT           : .3274219E+00
SUM PRODUCTIVE HOURS     : .9795249E+03  COST PER SYSTEM HOUR    : .4268211E+03
% NET UTILIZATION MACH. : .6365979E+02
% GROSS UTILIZATION MACH: .6944394E+02


------------------------- END OF RUN # 1 OF  1.-------------------------
```

APPENDIX E

7. Figure, Harvesting system 3



APPENDIX E

8. LOGSIM, Harvesting system 3 printout

```
*******************************************************
*                                                     *
*                 >>> L O G S I M <<<                 *
*              HARVESTING CONFIGURATION               *
*                                                     *
*******************************************************
```

SYSTEM3.MOD
****

NAME OF SIMULATION MODEL : SYSTEM3.MOD
AMOUNT TO BE HARVESTED (CU.FT.) : 255664.
TIME DELAY PARAMETER : .0100

MACHINE CONFIGURATION
===================

PROCESS #	IN ORIGIN	OUT DESTINATION
1		2.
2	1.	13.
3		
4		
5		
6		
7		
8		
9		
10		
11		
12		
13	2.	

MATERIAL FREQUENCY DISTRIBUTIONS
==============================

CUMULATIVE FREQUENCY DISTRIBUTION NO.1 : WHOLE TREES
CUMULATIVE FREQUENCY DISTRIBUTION NO.2 : SAWLOGS
CUMULATIVE FREQUENCY DISTRIBUTION NO.3 :
CUMULATIVE FREQUENCY DISTRIBUTION NO.4 :

CLASS	DISTRIBUTION 1 FREQ.%	CU.FT	DISTRIBUTION 2 FREQ.%	CU.FT	DISTRIBUTION 3 FREQ.%	CU.FT	DISTRIBUTION 4 FREQ.%	CU.FT
1	11.60	4.40	20.00	3.70				
2	29.70	9.40	50.30	9.30				
3	50.40	18.00	67.30	15.30				
4	69.90	28.30	80.80	21.50				
5	84.40	40.90	91.10	29.20				
6	93.00	54.60	96.10	38.00				
7	97.70	70.20	98.90	48.00				
8	100.00	92.10	100.00	59.40				

INVENTORY AND BUFFER SIZES

PRO-CESS	NAME	DISTRI-BUTION	STARTUP INV.	MINIMUM INV.	STARTUP MINIMUM	MAXIMUM INV.	STARTUP MAXIMUM
1	FELLING & SKIDDING	1.					
2	DELIMB & BUCKING	1.	2740.0	.0	.0	4580.0	3660.0
13	FINAL TRANSPORT	2.	3320.0	1311.0	1311.0	9330.0	7970.0

PROCESS NO. 1: FELLING & SKIDDING

OUTGOING DESTINATION : PROCESS NO. 2. DELIMB & BUCKING
DISTRIBUTION USED : 1. WHOLE TREES
LOADER USED : NONE
TIME DELAYS HANDELED BY : STANDARD BUILD-IN FUNCTIONS

MACHINES USED :

TYPE	NAME	INITIAL # OF MACHINES
1	CLAMBUNK FELL-SKID	1.

PROCESS NO. 2: DELIMB & BUCKING

```
INCOMING ORIGIN         : PROCESS NO. 1.  FELLING & SKIDDING
OUTGOING DESTINATION    : PROCESS NO.13.  FINAL TRANSPORT
DISTRIBUTION USED       :      1.         WHOLE TREES
STARTUP-INVENTORY LEVEL :   2740.0
MINIMUM INVENTORY LEVEL :      .0
STARTUP LEVEL MINIMUM   :      .0
MAXIMUM INVENTORY LEVEL :   4580.0
STARTUP LEVEL MAXIMUM   :   3660.0
LOADER USED             : NONE
TIME DELAYS HANDELED BY : STANDARD BUILD-IN FUNCTIONS

MACHINES USED           :
TYPE   NAME                      INITIAL # OF MACHINES
------------------------------------------------------
  5    GRAPPLE PROCESSOR                 1.
```

PROCESS NO.13: FINAL TRANSPORT

```
INCOMING ORIGIN         : PROCESS NO. 2.  DELIMB & BUCKING
DISTRIBUTION USED       :      2.         SAWLOGS
STARTUP-INVENTORY LEVEL :   3320.0
MINIMUM INVENTORY LEVEL :   1311.0
STARTUP LEVEL MINIMUM   :   1311.0
MAXIMUM INVENTORY LEVEL :   9330.0
STARTUP LEVEL MAXIMUM   :   7970.0
LOADER USED             :     32.   LOG LOADER
TIME DELAYS HANDELED BY : STANDARD BUILD-IN FUNCTIONS

MACHINES USED           :
TYPE   NAME                      INITIAL # OF MACHINES
------------------------------------------------------
 41    LOG TRUCK                         3.
```

MACHINE TYPE 1: CLAMBUNK FELL-SKID

```
INITIAL NUMBER OF MACHINES        :   1.
AVERAGE PROCESSING TIME / TREE    :     .0000
FIXED CONSTANT TIME / LOAD        :     .6180
FIXED CONST. TIME / ONE WAY HAUL  :     .0000
FIXED COST / SCHEDULED HOUR       :   57.20
```

VARIABLE COST/ MACHINE HOUR : 46.61
MACHINE CAPACITY IN CU.FT : 915.22

FREQUENCY DISTRIBUTIONS MACHINE TYPE 1 : CLAMBUNK FELL-SKID

CLASS	CUM FREQ.%	TIME BETW.FAILURE	CUM.FREQ.%	REPAIR TIME
1	20.00	3.00	20.00	.40
2	40.00	9.00	40.00	.90
3	60.00	16.00	60.00	2.00
4	80.00	28.00	80.00	5.00
5	100.00	53.00	100.00	10.00

MACHINE TYPE 5: GRAPPLE PROCESSOR

INITIAL NUMBER OF MACHINES : 1.
AVERAGE PROCESSING TIME / TREE : .0067
FIXED CONSTANT TIME / LOAD : .0000
FIXED CONST. TIME / ONE WAY HAUL : .0000
FIXED COST / SCHEDULED HOUR : 48.00
VARIABLE COST/ MACHINE HOUR : 46.00
MACHINE CAPACITY IN CU.FT : 99999.00

FREQUENCY DISTRIBUTIONS MACHINE TYPE 5 : GRAPPLE PROCESSOR

CLASS	CUM FREQ.%	TIME BETW.FAILURE	CUM.FREQ.%	REPAIR TIME
1	20.00	3.00	30.00	1.00
2	40.00	7.00	55.00	3.00
3	60.00	12.00	75.00	7.00
4	80.00	22.00	90.00	11.00
5	100.00	44.00	100.00	16.00

MACHINE TYPE 32: LOG LOADER

INITIAL NUMBER OF MACHINES : 1.
AVERAGE PROCESSING TIME / TREE : .0000
FIXED CONSTANT TIME / LOAD : .2500
FIXED CONST. TIME / ONE WAY HAUL : .0000
FIXED COST / SCHEDULED HOUR : 41.99
VARIABLE COST/ MACHINE HOUR : 41.35
MACHINE CAPACITY IN CU.FT : 1316.00

FREQUENCY DISTRIBUTIONS MACHINE TYPE 32 : LOG LOADER

CLASS	CUM FREQ.%	TIME BETW.FAILURE	CUM.FREQ.%	REPAIR TIME
1	20.00	7.00	35.00	.50
2	40.00	14.00	65.00	1.00
3	60.00	28.00	85.00	3.00
4	80.00	56.00	95.00	8.00
5	100.00	100.00	100.00	16.00

MACHINE TYPE 41: LOG TRUCK
****

INITIAL NUMBER OF MACHINES : 3.
AVERAGE PROCESSING TIME / TREE : .0000
FIXED CONSTANT TIME / LOAD : .0000
FIXED CONST. TIME / ONE WAY HAUL : 1.0000
FIXED COST / SCHEDULED HOUR : 15.04
VARIABLE COST/ MACHINE HOUR : 36.48
MACHINE CAPACITY IN CU.FT : 1316.00

FREQUENCY DISTRIBUTIONS MACHINE TYPE 41 : LOG TRUCK

CLASS	CUM FREQ.%	TIME BETW.FAILURE	CUM.FREQ.%	REPAIR TIME
1	20.00	7.00	35.00	.50
2	40.00	14.00	65.00	1.00
3	60.00	28.00	85.00	3.00
4	80.00	56.00	95.00	8.00
5	100.00	100.00	100.00	16.00

APPENDIX E

9. LOGSIM, simulation results system 3

```
          ***********************************************************
          *                                                         *
          *                  >>> L O G S I M <<<                    *
          *                  SIMULATION  RESULTS                    *
          *                                                         *
          ***********************************************************


SIMULATION MODEL USED: SYSTEM3.MOD
**********************

DATE= 06-01-87
TIME= 00:03:20
SIMULATION RUN  1 OF  1.



         PROCESS NO. 1 : FELLING & SKIDDING
         ===============


TIME BEGIN OF PROCESS    : .0000000E+00  AVERAGE INVENTORY        : .0000000E+00
TIME END OF PROCESS      : .2542438E+03  MAXIMUM INVENTORY        : .0000000E+00
DURATION OF PROCESS      : .2542438E+03  MINIMUM INVENTORY        : .0000000E+00
TIME INVENTORY TOO LOW   : .0000000E+00  STD.DEV.INVENTORY        : .0000000E+00
TIME INVENTORY TOO HIGH  : .0000000E+00  # OF OBSERVATIONS INV.   : .0000000E+00
% INVENTORY DOWNTIME     : .0000000E+00  SUM UNITS PROCESSED      : .2556640E+06
TOTAL # OF MACHINES      :            1  SUM COST OF PROCESS      : .2278094E+05
SUM SCHEDULED HOURS      : .2542438E+03  COST PER UNIT            : .8910500E-01
SUM MACH.BREAKDOWN HOURS: .4060000E+02  COST PER SCHEDULED HOUR  : .8960272E+02
SUM PRODUCTIVE HOURS     : .1767473E+03
% NET UTILIZATION MACH.  : .6951881E+02
% GROSS UTILIZATION MACH: .8548773E+02



MACHINE TYPE  1 : CLAMBUNK FELL-SKID
-----------------

TOTAL # OF MACHINES      :            1  COST PER MACHINE         : .2278094E+05
SUM SCHEDULED HOURS      : .2542438E+03  COST PER SCHEDULED HOUR  : .8960272E+02
SUM MACH.BREAKDOWN HOURS: .4060000E+02  % NET UTILIZATION MACH.  : .6951881E+02
SUM PRODUCTIVE HOURS     : .1767473E+03  % GROSS UTILIZATION MACH: .8548773E+02
```

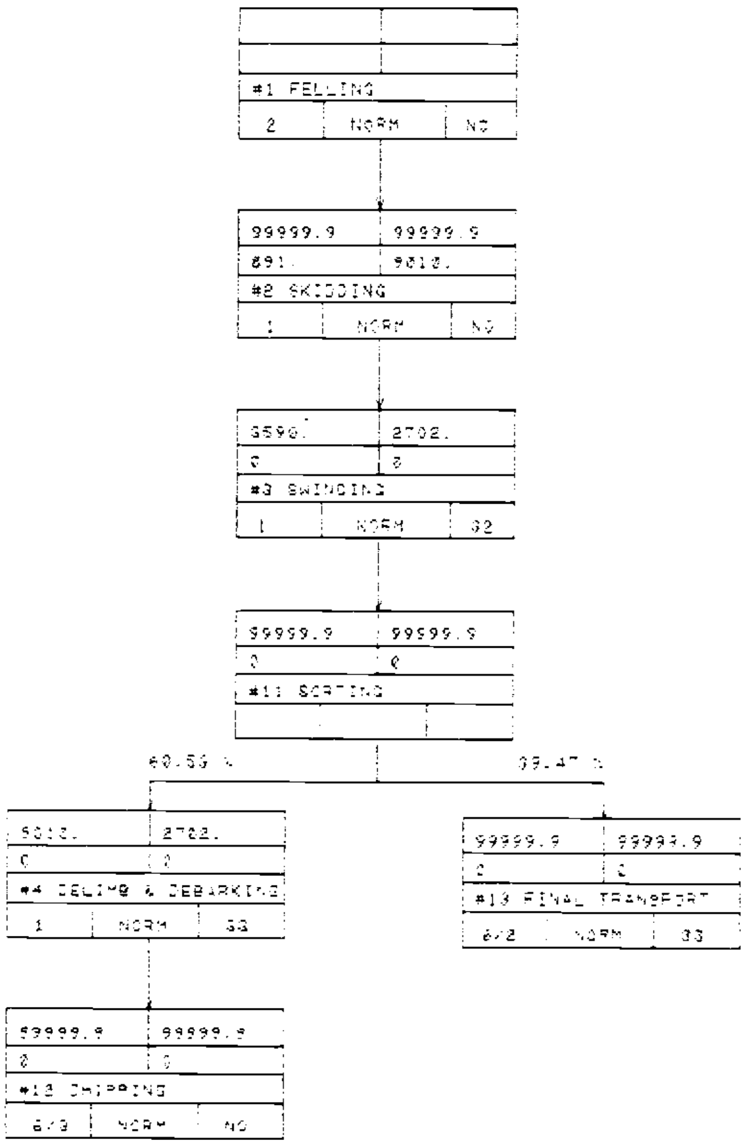
```
        PROCESS NO. 2 : DELIMB & BUCKING
        ===============

TIME BEGIN OF PROCESS     : .2472000E+01  AVERAGE INVENTORY         : .7481161E+03
TIME END OF PROCESS       : .2544584E+03  MAXIMUM INVENTORY         : .5383209E+04
DURATION OF PROCESS       : .2519864E+03  MINIMUM INVENTORY         : .0000000E+00
TIME INVENTORY TOO LOW    : .0000000E+00  STD.DEV.INVENTORY         : .9263040E+03
TIME INVENTORY TOO HIGH   : .3689656E+02  # OF OBSERVATIONS INV.    : .9891000E+04
% INVENTORY DOWNTIME      : .1464228E+02  SUM UNITS PROCESSED       : .2556695E+06
TOTAL # OF MACHINES       :            1  SUM COST OF PROCESS       : .1505511E+05
SUM SCHEDULED HOURS       : .2519864E+03  COST PER UNIT             : .5888504E-01
SUM MACH.BREAKDOWN HOURS: .7200000E+02  COST PER SCHEDULED HOUR : .5974572E+02
SUM PRODUCTIVE HOURS      : .6434268E+02
% NET UTILIZATION MACH.  : .2553419E+02
% GROSS UTILIZATION MACH: .5410716E+02


MACHINE TYPE  5 : GRAPPLE PROCESSOR
-----------------

TOTAL # OF MACHINES       :            1  COST PER MACHINE          : .1505511E+05
SUM SCHEDULED HOURS       : .2519864E+03  COST PER SCHEDULED HOUR : .5974572E+02
SUM MACH.BREAKDOWN HOURS: .7200000E+02  % NET UTILIZATION MACH.  : .2553419E+02
SUM PRODUCTIVE HOURS      : .6434268E+02  % GROSS UTILIZATION MACH: .5410716E+02


        PROCESS NO.13 : FINAL TRANSPORT
        ===============

TIME BEGIN OF PROCESS     : .3255904E+01  AVERAGE INVENTORY         : .1317993E+04
TIME END OF PROCESS       : .2591564E+03  MAXIMUM INVENTORY         : .7058067E+04
DURATION OF PROCESS       : .2559005E+03  MINIMUM INVENTORY         : .0000000E+00
TIME INVENTORY TOO LOW    : .1987365E+03  STD.DEV.INVENTORY         : .1207547E+04
TIME INVENTORY TOO HIGH   : .0000000E+00  # OF OBSERVATIONS INV.    : .9802000E+04
% INVENTORY DOWNTIME      : .7766161E+02  SUM UNITS PROCESSED       : .2556640E+06
TOTAL # OF MACHINES       :            3  SUM COST OF PROCESS       : .2769113E+05
SUM SCHEDULED HOURS       : .7677016E+03  COST PER UNIT             : .1083106E+00
SUM MACH.BREAKDOWN HOURS: .2650000E+02  COST PER SCHEDULED HOUR : .3607017E+02
SUM PRODUCTIVE HOURS      : .4425683E+03
% NET UTILIZATION MACH.  : .5764848E+02
% GROSS UTILIZATION MACH: .6110035E+02
```

```
MACHINE TYPE 41 : LOG TRUCK
-----------------

TOTAL # OF MACHINES      :             3  COST PER MACHINE         : .9230375E+04
SUM SCHEDULED HOURS      : .7677016E+03  COST PER SCHEDULED HOUR : .3607017E+02
SUM MACH.BREAKDOWN HOURS: .2650000E+02  % NET UTILIZATION MACH.  : .5764848E+02
SUM PRODUCTIVE HOURS     : .4425683E+03  % GROSS UTILIZATION MACH: .6110035E+02



        LOADING DEVICES
        ===============

TOTAL # OF MACHINES      :             1  SUM OF UNITS HARVESTED  : .2556640E+06
SUM SCHEDULED HOURS      : .2591564E+03  SUM COST LOADER DEVICES : .1275357E+05
SUM MACH.BREAKDOWN HOURS: .5000000E+01  COST PER UNIT            : .4988410E-01
SUM PRODUCTIVE HOURS     : .4856844E+02  COST PER SCHEDULED HOUR : .4921185E+02
% NET UTILIZATION MACH.  : .1874097E+02
% GROSS UTILIZATION MACH: .2067031E+02


MACHINE TYPE 32 : LOG LOADER
-----------------

TOTAL # OF MACHINES      :             1  COST PER MACHINE         : .1289028E+05
SUM SCHEDULED HOURS      : .2591564E+03  COST PER SCHEDULED HOUR : .4973940E+02
SUM MACH.BREAKDOWN HOURS: .5000000E+01  % NET UTILIZATION MACH.  : .1874097E+02
SUM PRODUCTIVE HOURS     : .4856844E+02  % GROSS UTILIZATION MACH: .2067031E+02



        COMPLETE HARVESTING SYSTEM STATISTICS
        =====================================

COMPUTER TIME START SIMULATION   DATE: 05-31-87     TIME: 23:41:40
COMPUTER TIME END SIMULATION     DATE: 06-01-87     TIME: 00:03:25

SIMULATION RUN  1 OF  1.
BEGIN OF HARVESTING      : .0000000E+00  END OF HARVESTING        : .2591564E+03
TOTAL # OF MACHINES      :             6  SUM OF UNITS HARVESTED  : .2556640E+06
SUM SCHEDULED HOURS      : .1533088E+04  SUM COST OF SYSTEM       : .7828074E+05
SUM MACH.BREAKDOWN HOURS: .1441000E+03  COST PER UNIT            : .3061860E+00
SUM PRODUCTIVE HOURS     : .7322267E+03  COST PER SYSTEM HOUR    : .3020598E+03
% NET UTILIZATION MACH.  : .4776155E+02
% GROSS UTILIZATION MACH: .5716088E+02
```

-------------------------- END OF RUN # 1 OF 1.--------------------------

APPENDIX E

10. Figure, Harvesting system 4



APPENDIX E

11. LOGSIM, Harvesting system 4 printout

```
*******************************************************
*                                                     *
*                 >>> L O G S I M <<<                 *
*               HARVESTING CONFIGURATION              *
*                                                     *
*******************************************************
```

SYSTEM4.MOD
****

```
NAME OF SIMULATION MODEL        : SYSTEM4.MOD
AMOUNT TO BE HARVESTED (CU.FT.) :  255664.
TIME DELAY PARAMETER            :        .1000
```

MACHINE CONFIGURATION
=====================

PROCESS #	IN ORIGIN	OUT DESTINATION
1		2.
2	1.	3.
3	2.	11.
4	11.	12.
5		
6		
7		
8		
9		
10		
11	3.	4. 13.
12	4.	
13	11.	

MATERIAL FREQUENCY DISTRIBUTIONS

CUMULATIVE FREQUENCY DISTRIBUTION NO.1 : WHOLE TREES
CUMULATIVE FREQUENCY DISTRIBUTION NO.2 : SAWLOGS
CUMULATIVE FREQUENCY DISTRIBUTION NO.3 : PULPWOOD
CUMULATIVE FREQUENCY DISTRIBUTION NO.4 :

	DISTRIBUTION 1		DISTRIBUTION 2		DISTRIBUTION 3		DISTRIBUTION 4	
CLASS	FREQ.%	CU.FT	FREQ.%	CU.FT	FREQ.%	CU.FT	FREQ.%	CU.FT
1	3.30	2.00	53.30	25.00	3.30	2.00		
2	14.60	6.00	79.30	38.00	14.60	6.00		
3	32.10	11.00	93.60	48.00	34.20	11.00		
4	52.10	19.00	100.00	59.00	77.50	18.00		
5	70.00	29.00			82.10	25.00		
6	85.00	42.00			100.00	29.00		
7	93.30	56.00						
8	97.90	73.00						
9	100.00	93.00						

INVENTORY AND BUFFER SIZES

PRO-CESS	NAME	DISTRI-BUTION	STARTUP INV.	MINIMUM INV.	STARTUP MINIMUM	MAXIMUM INV.	STARTUP MAXIMUM
1	FELLING	1.					
2	SKIDDING	1.	9010.0	891.0	9010.0	999999.9	999999.9
3	SWINGING	1.	901.0	.0	.0	3590.0	2702.0
4	DELIMB & DEBARK	3.	1.0	.0	.0	9010.0	2702.0
11	DISTRIBUTION	1.	1.0	.0	.0	999999.9	999999.9
12	CHIPPING	3.	1.0	.0	.0	999999.9	999999.9
13	LOG FTRAPO	2.	1.0	.0	.0	999999.9	999999.9

PROCESS NO. 1: FELLING

OUTGOING DESTINATION : PROCESS NO. 2. SKIDDING
DISTRIBUTION USED : 1. WHOLE TREES
LOADER USED : NONE
TIME DELAYS HANDELED BY : STANDARD BUILD-IN FUNCTIONS

```
MACHINES USED            :
TYPE   NAME                        INITIAL # OF MACHINES
--------------------------------------------------------
  1    CAT 227 FELL-BUNCH             2.



          PROCESS NO. 2: SKIDDING
          ==============

INCOMING ORIGIN          : PROCESS NO. 1.  FELLING
OUTGOING DESTINATION     : PROCESS NO. 3.  SWINGING
DISTRIBUTION USED        :       1.        WHOLE TREES
STARTUP-INVENTORY LEVEL  :   9010.0
MINIMUM INVENTORY LEVEL  :    891.0
STARTUP LEVEL MINIMUM    :   9010.0
MAXIMUM INVENTORY LEVEL  : 999999.9
STARTUP LEVEL MAXIMUM    : 999999.9
LOADER USED              : NONE
TIME DELAYS HANDELED BY  : STANDARD BUILD-IN FUNCTIONS

MACHINES USED            :
TYPE   NAME                        INITIAL # OF MACHINES
--------------------------------------------------------
  5    TJ CLAMBUNK SKID               1.



          PROCESS NO. 3: SWINGING
          ==============

INCOMING ORIGIN          : PROCESS NO. 2.  SKIDDING
OUTGOING DESTINATION     : PROCESS NO.11.  DISTRIBUTION
DISTRIBUTION USED        :       1.        WHOLE TREES
STARTUP-INVENTORY LEVEL  :    901.0
MINIMUM INVENTORY LEVEL  :       .0
STARTUP LEVEL MINIMUM    :       .0
MAXIMUM INVENTORY LEVEL  :   3590.0
STARTUP LEVEL MAXIMUM    :   2702.0
LOADER USED              :      32.  CAT 225 LOG LOADER
TIME DELAYS HANDELED BY  : STANDARD BUILD-IN FUNCTIONS

MACHINES USED            :
TYPE   NAME                        INITIAL # OF MACHINES
--------------------------------------------------------
  8    SET-OUT TRUCK                  1.
```

```
        PROCESS NO. 4: DELIMB & DEBARK
        ==============

INCOMING ORIGIN         :  PROCESS NO.11.  DISTRIBUTION
OUTGOING DESTINATION    :  PROCESS NO.12.  CHIPPING
DISTRIBUTION USED       :      3.          PULPWOOD
STARTUP-INVENTORY LEVEL :      1.0
MINIMUM INVENTORY LEVEL :       .0
STARTUP LEVEL MINIMUM   :       .0
MAXIMUM INVENTORY LEVEL :   9010.0
STARTUP LEVEL MAXIMUM   :   2702.0
LOADER USED             :     33.   CAT 225 W/SLASHER
TIME DELAYS HANDELED BY :  STANDARD BUILD-IN FUNCTIONS

MACHINES USED           :
TYPE   NAME                       INITIAL # OF MACHINES
-------------------------------------------------------
 11    CHAIN-FLAIL DEB.                   1.


        PROCESS NO.11: DISTRIBUTION
        ==============

INCOMING ORIGIN         :  PROCESS NO. 3.  SWINGING
OUTGOING ROUTE 1        :  PROCESS NO. 4.  DELIMB & DEBARK
OUTGOING ROUTE 2        :  PROCESS NO.13.  LOG FTRAPO
% GOING ROUTE 1         :     60.53 %
% GOING ROUTE 2         :     39.47 %
DISTRIBUTION USED       :      1.          WHOLE TREES
STARTUP-INVENTORY LEVEL :      1.0
MINIMUM INVENTORY LEVEL :       .0
STARTUP LEVEL MINIMUM   :       .0
MAXIMUM INVENTORY LEVEL :  999999.9
STARTUP LEVEL MAXIMUM   :  999999.9
LOADER USED             :  NONE
TIME DELAYS HANDELED BY :  STANDARD BUILD-IN FUNCTIONS


        PROCESS NO.12: CHIPPING
        ==============

INCOMING ORIGIN         :  PROCESS NO. 4.  DELIMB & DEBARK
DISTRIBUTION USED       :      3.          PULPWOOD
STARTUP-INVENTORY LEVEL :      1.0
MINIMUM INVENTORY LEVEL :       .0
```

STARTUP LEVEL MINIMUM : .0
MAXIMUM INVENTORY LEVEL : 999999.9
STARTUP LEVEL MAXIMUM : 999999.9
LOADER USED : NONE
TIME DELAYS HANDELED BY : STANDARD BUILD-IN FUNCTIONS

MACHINES USED :

TYPE	NAME	INITIAL # OF MACHINES
37	CHIPPER	1.
39	CHIP TRAILER	6.
40	CHIP TRAKTOR	3.

PROCESS NO. 13: LOG FTRAPO

INCOMING ORIGIN : PROCESS NO.11. DISTRIBUTION
DISTRIBUTION USED : 2. SAWLOGS
STARTUP-INVENTORY LEVEL : 1.0
MINIMUM INVENTORY LEVEL : .0
STARTUP LEVEL MINIMUM : .0
MAXIMUM INVENTORY LEVEL : 999999.9
STARTUP LEVEL MAXIMUM : 999999.9
LOADER USED : 33. CAT 225 W/SLASHER
TIME DELAYS HANDELED BY : STANDARD BUILD-IN FUNCTIONS

MACHINES USED :

TYPE	NAME	INITIAL # OF MACHINES
41	LOG TRAILER	6.
42	LOG TRAKTOR	2.

MACHINE TYPE 1: CAT 227 FELL-BUNCH

INITIAL NUMBER OF MACHINES : 2.
AVERAGE PROCESSING TIME / TREE : .0000
FIXED CONSTANT TIME / LOAD : .0384
FIXED CONST. TIME / ONE WAY HAUL : .0000
FIXED COST / SCHEDULED HOUR : 41.99
VARIABLE COST/ MACHINE HOUR : 41.35
MACHINE CAPACITY IN CU.FT : 82.65

FREQUENCY DISTRIBUTIONS MACHINE TYPE 1 : CAT 227 FELL-BUNCH

CLASS	CUM FREQ.%	TIME BETW.FAILURE	CUM.FREQ.%	REPAIR TIME
1	20.00	6.00	50.00	.50
2	40.00	12.00	70.00	1.00
3	60.00	20.00	80.00	2.00
4	80.00	36.00	90.00	5.00
5	100.00	64.00	100.00	10.00

MACHINE TYPE 5: TJ CLAMBUNK SKID

INITIAL NUMBER OF MACHINES : 1.
AVERAGE PROCESSING TIME / TREE : .0072
FIXED CONSTANT TIME / LOAD : .0170
FIXED CONST. TIME / ONE WAY HAUL : .1207
FIXED COST / SCHEDULED HOUR : 57.20
VARIABLE COST/ MACHINE HOUR : 46.61
MACHINE CAPACITY IN CU.FT : 901.00

FREQUENCY DISTRIBUTIONS MACHINE TYPE 5 : TJ CLAMBUNK SKID

CLASS	CUM FREQ.%	TIME BETW.FAILURE	CUM.FREQ.%	REPAIR TIME
1	20.00	6.00	50.00	.50
2	40.00	12.00	70.00	1.00
3	60.00	20.00	80.00	2.00
4	80.00	36.00	90.00	5.00
5	100.00	64.00	100.00	10.00

MACHINE TYPE 8: SET-OUT TRUCK

INITIAL NUMBER OF MACHINES : 1.
AVERAGE PROCESSING TIME / TREE : .0000
FIXED CONSTANT TIME / LOAD : .1500
FIXED CONST. TIME / ONE WAY HAUL : .3000
FIXED COST / SCHEDULED HOUR : 15.04
VARIABLE COST/ MACHINE HOUR : 36.48
MACHINE CAPACITY IN CU.FT : 1842.00

FREQUENCY DISTRIBUTIONS MACHINE TYPE 8 : SET-OUT TRUCK

CLASS	CUM FREQ.%	TIME BETW.FAILURE	CUM.FREQ.%	REPAIR TIME
1	20.00	7.00	35.00	.50
2	40.00	14.00	65.00	1.00
3	60.00	28.00	85.00	3.00
4	80.00	56.00	95.00	8.00
5	100.00	100.00	100.00	16.00

MACHINE TYPE 11: CHAIN-FLAIL DEB.

===============

INITIAL NUMBER OF MACHINES : 1.
AVERAGE PROCESSING TIME / TREE : .0031
FIXED CONSTANT TIME / LOAD : .0000
FIXED CONST. TIME / ONE WAY HAUL : .0000
FIXED COST / SCHEDULED HOUR : 57.20
VARIABLE COST/ MACHINE HOUR : 30.00
MACHINE CAPACITY IN CU.FT : 99999.00

FREQUENCY DISTRIBUTIONS MACHINE TYPE 11 : CHAIN-FLAIL DEB.

CLASS	CUM FREQ.%	TIME BETW.FAILURE	CUM.FREQ.%	REPAIR TIME
1	50.00	4.00	50.00	.50
2	75.00	8.00	70.00	1.00
3	85.00	20.00	80.00	4.00
4	95.00	36.00	90.00	8.00
5	100.00	64.00	100.00	16.00

MACHINE TYPE 32: CAT 225 LOG LOADER

===============

INITIAL NUMBER OF MACHINES : 1.
AVERAGE PROCESSING TIME / TREE : .0000
FIXED CONSTANT TIME / LOAD : .2500
FIXED CONST. TIME / ONE WAY HAUL : .0000
FIXED COST / SCHEDULED HOUR : 41.99
VARIABLE COST/ MACHINE HOUR : 41.35
MACHINE CAPACITY IN CU.FT : 1842.00

FREQUENCY DISTRIBUTIONS MACHINE TYPE 32 : CAT 225 LOG LOADER

CLASS	CUM FREQ.%	TIME BETW.FAILURE	CUM.FREQ.%	REPAIR TIME
1	20.00	7.00	35.00	.50
2	40.00	14.00	65.00	1.00
3	60.00	28.00	85.00	3.00
4	80.00	56.00	95.00	8.00
5	100.00	100.00	100.00	16.00

MACHINE TYPE 33: CAT 225 W/SLASHER

===============

INITIAL NUMBER OF MACHINES : 1.
AVERAGE PROCESSING TIME / TREE : .0033
FIXED CONSTANT TIME / LOAD : .0000
FIXED CONST. TIME / ONE WAY HAUL : .0000
FIXED COST / SCHEDULED HOUR : 41.99
VARIABLE COST/ MACHINE HOUR : 41.35
MACHINE CAPACITY IN CU.FT : 82.56

FREQUENCY DISTRIBUTIONS MACHINE TYPE 33 : CAT 225 W/SLASHER

CLASS	CUM FREQ.%	TIME BETW.FAILURE	CUM.FREQ.%	REPAIR TIME
1	20.00	6.00	50.00	.50
2	40.00	12.00	70.00	1.00
3	60.00	20.00	80.00	2.00
4	80.00	36.00	90.00	5.00
5	100.00	64.00	100.00	10.00

MACHINE TYPE 37: CHIPPER

===============

INITIAL NUMBER OF MACHINES : 1.
AVERAGE PROCESSING TIME / TREE : .0031
FIXED CONSTANT TIME / LOAD : .0000
FIXED CONST. TIME / ONE WAY HAUL : .0000
FIXED COST / SCHEDULED HOUR : 41.98
VARIABLE COST/ MACHINE HOUR : 58.58
MACHINE CAPACITY IN CU.FT : 99999.00

FREQUENCY DISTRIBUTIONS MACHINE TYPE 37 : CHIPPER

CLASS	CUM FREQ.%	TIME BETW.FAILURE	CUM.FREQ.%	REPAIR TIME
1	50.00	4.00	50.00	.50
2	75.00	8.00	70.00	1.00
3	85.00	20.00	80.00	4.00
4	95.00	36.00	90.00	8.00
5	100.00	64.00	100.00	16.00

MACHINE TYPE 39: CHIP TRAILER

INITIAL NUMBER OF MACHINES : 6.
FIXED CONSTANT TIME / LOAD : .1500
FIXED CONST. TIME / ONE WAY HAUL : 1.9000
FIXED COST / SCHEDULED HOUR : 3.02
VARIABLE COST/ MACHINE HOUR : 2.88
MACHINE CAPACITY IN CU.FT : 1316.00

FREQUENCY DISTRIBUTIONS MACHINE TYPE 39 : CHIP TRAILER

CLASS	CUM FREQ.%	TIME BETW.FAILURE	CUM.FREQ.%	REPAIR TIME

****** DISTRIBUTION NOT USED ******

MACHINE TYPE 40: CHIP TRAKTOR

FIXED COST / SCHEDULED HOUR : 13.12
VARIABLE COST/ MACHINE HOUR : 34.24
MACHINE CAPACITY IN CU.FT : .00

FREQUENCY DISTRIBUTIONS MACHINE TYPE 40 : CHIP TRAKTOR

CLASS	CUM FREQ.%	TIME BETW.FAILURE	CUM.FREQ.%	REPAIR TIME
1	20.00	7.00	35.00	.50
2	40.00	14.00	65.00	1.00
3	60.00	28.00	85.00	3.00
4	80.00	56.00	95.00	8.00
5	100.00	100.00	100.00	16.00

MACHINE TYPE 41: LOG TRAILER
================

INITIAL NUMBER OF MACHINES	:	6.
AVERAGE PROCESSING TIME / TREE	:	.0000
FIXED CONSTANT TIME / LOAD	:	.1500
FIXED CONST. TIME / ONE WAY HAUL	:	1.9000
FIXED COST / SCHEDULED HOUR	:	1.28
VARIABLE COST/ MACHINE HOUR	:	2.24
MACHINE CAPACITY IN CU.FT	:	1316.00

FREQUENCY DISTRIBUTIONS MACHINE TYPE 41 : LOG TRAILER

CLASS	CUM FREQ.%	TIME BETW.FAILURE	CUM.FREQ.%	REPAIR TIME

****** DISTRIBUTION NOT USED ******

MACHINE TYPE 42: LOG TRAKTOR
================

FIXED COST / SCHEDULED HOUR	:	13.12
VARIABLE COST/ MACHINE HOUR	:	34.24
MACHINE CAPACITY IN CU.FT	:	.00

FREQUENCY DISTRIBUTIONS MACHINE TYPE 42 : LOG TRAKTOR

CLASS	CUM FREQ.%	TIME BETW.FAILURE	CUM.FREQ.%	REPAIR TIME

****** DISTRIBUTION NOT USED ******

APPENDIX E

12. LOGSIM, simulation results system 4

```
*****************************************************
*                                                   *
*                 >>> L O G S I M <<<               *
*                 SIMULATION  RESULTS               *
*                                                   *
*****************************************************
```

SIMULATION MODEL USED: SYSTEM4.MOD
****

DATE= 05-31-87
TIME= 22:16:34
SIMULATION RUN 1 OF 1.

PROCESS NO. 1 : FELLING
===============

TIME BEGIN OF PROCESS	: .0000000E+00	AVERAGE INVENTORY	: .0000000E+00
TIME END OF PROCESS	: .8798502E+02	MAXIMUM INVENTORY	: .0000000E+00
DURATION OF PROCESS	: .8798502E+02	MINIMUM INVENTORY	: .0000000E+00
TIME INVENTORY TOO LOW	: .0000000E+00	STD.DEV.INVENTORY	: .0000000E+00
TIME INVENTORY TOO HIGH	: .0000000E+00	# OF OBSERVATIONS INV.	: .0000000E+00
% INVENTORY DOWNTIME	: .0000000E+00	SUM UNITS PROCESSED	: .2556640E+06
TOTAL # OF MACHINES	: 2	SUM COST OF PROCESS	: .1408602E+05
SUM SCHEDULED HOURS	: .1759700E+03	COST PER UNIT	: .5509583E-01
SUM MACH.BREAKDOWN HOURS	: .1400000E+02	COST PER SCHEDULED HOUR	: .8004783E+02
SUM PRODUCTIVE HOURS	: .1619598E+03		
% NET UTILIZATION MACH.	: .9203829E+02		
% GROSS UTILIZATION MACH	: .9999419E+02		

MACHINE TYPE 1 : CAT 227 FELL-BUNCH

TOTAL # OF MACHINES	: 2	COST PER MACHINE	: .7043010E+04
SUM SCHEDULED HOURS	: .1759700E+03	COST PER SCHEDULED HOUR	: .8004783E+02
SUM MACH.BREAKDOWN HOURS	: .1400000E+02	% NET UTILIZATION MACH.	: .9203829E+02
SUM PRODUCTIVE HOURS	: .1619598E+03	% GROSS UTILIZATION MACH	: .9999419E+02

PROCESS NO. 2 : SKIDDING

TIME BEGIN OF PROCESS	: .2803200E+01	AVERAGE INVENTORY	: .6186050E+05
TIME END OF PROCESS	: .1567840E+03	MAXIMUM INVENTORY	: .1216110E+06
DURATION OF PROCESS	: .1539808E+03	MINIMUM INVENTORY	: .0000000E+00
TIME INVENTORY TOO LOW	: .0000000E+00	STD.DEV.INVENTORY	: .3376338E+05
TIME INVENTORY TOO HIGH	: .0000000E+00	# OF OBSERVATIONS INV.	: .4508000E+04
% INVENTORY DOWNTIME	: .0000000E+00	SUM UNITS PROCESSED	: .2556640E+06
TOTAL # OF MACHINES	: 1	SUM COST OF PROCESS	: .1543550E+05
SUM SCHEDULED HOURS	: .1539808E+03	COST PER UNIT	: .6037418E-01
SUM MACH.BREAKDOWN HOURS	: .9500000E+01	COST PER SCHEDULED HOUR	: .1002431E+03
SUM PRODUCTIVE HOURS	: .1421970E+03		
% NET UTILIZATION MACH.	: .9234727E+02		
% GROSS UTILIZATION MACH	: .9851688E+02		

MACHINE TYPE 5 : TJ CLAMBUNK SKID

TOTAL # OF MACHINES	: 1	COST PER MACHINE	: .1543550E+05
SUM SCHEDULED HOURS	: .1539808E+03	COST PER SCHEDULED HOUR	: .1002431E+03
SUM MACH.BREAKDOWN HOURS	: .9500000E+01	% NET UTILIZATION MACH.	: .9234727E+02
SUM PRODUCTIVE HOURS	: .1421970E+03	% GROSS UTILIZATION MACH	: .9851688E+02

PROCESS NO.11 : DISTRIBUTION

TIME BEGIN OF PROCESS	: .4882385E+01	AVERAGE INVENTORY	: .3560324E+03
TIME END OF PROCESS	: .1586829E+03	MAXIMUM INVENTORY	: .4929597E+04
DURATION OF PROCESS	: .1538005E+03	MINIMUM INVENTORY	: .0000000E+00
TIME INVENTORY TOO LOW	: .0000000E+00	STD.DEV.INVENTORY	: .7710981E+03
TIME INVENTORY TOO HIGH	: .0000000E+00	# OF OBSERVATIONS INV.	: .2023000E+04
% INVENTORY DOWNTIME	: .0000000E+00	SUM UNITS PROCESSED	: .2556640E+06
TOTAL # OF MACHINES	: 0	SUM COST OF PROCESS	: .0000000E+00
SUM SCHEDULED HOURS	: .0000000E+00	COST PER UNIT	: .0000000E+00
SUM MACH.BREAKDOWN HOURS	: .0000000E+00	COST PER SCHEDULED HOUR	: .0000000E+00
SUM PRODUCTIVE HOURS	: .0000000E+00		
% NET UTILIZATION MACH.	: .0000000E+00		
% GROSS UTILIZATION MACH	: .0000000E+00		

PROCESS NO. 3 : SWINGING

===============

TIME BEGIN OF PROCESS	: .3688900E+01	AVERAGE INVENTORY	: .1616534E+04
TIME END OF PROCESS	: .1588901E+03	MAXIMUM INVENTORY	: .4006000E+04
DURATION OF PROCESS	: .1552012E+03	MINIMUM INVENTORY	: .0000000E+00
TIME INVENTORY TOO LOW	: .0000000E+00	STD.DEV.INVENTORY	: .8845016E+03
TIME INVENTORY TOO HIGH	: .2995998E+01	# OF OBSERVATIONS INV.	: .4310000E+03
% INVENTORY DOWNTIME	: .1930397E+01	SUM UNITS PROCESSED	: .2556640E+06
TOTAL # OF MACHINES	: 1	SUM COST OF PROCESS	: .7457829E+04
SUM SCHEDULED HOURS	: .1552012E+03	COST PER UNIT	: .2917043E-01
SUM MACH.BREAKDOWN HOURS	: .5000000E+01	COST PER SCHEDULED HOUR	: .4805266E+02
SUM PRODUCTIVE HOURS	: .1404496E+03		
% NET UTILIZATION MACH.	: .9049523E+02		
% GROSS UTILIZATION MACH	: .9371686E+02		

MACHINE TYPE 8 : SET-OUT TRUCK

TOTAL # OF MACHINES	: 1	COST PER MACHINE	: .7457829E+04
SUM SCHEDULED HOURS	: .1552012E+03	COST PER SCHEDULED HOUR	: .4805266E+02
SUM MACH.BREAKDOWN HOURS	: .5000000E+01	% NET UTILIZATION MACH.	: .9049523E+02
SUM PRODUCTIVE HOURS	: .1404496E+03	% GROSS UTILIZATION MACH	: .9371686E+02

PROCESS NO. 4 : DELIMB & DEBARK

===============

TIME BEGIN OF PROCESS	: .4882385E+01	AVERAGE INVENTORY	: .2006120E+04
TIME END OF PROCESS	: .1590361E+03	MAXIMUM INVENTORY	: .9010000E+04
DURATION OF PROCESS	: .1541537E+03	MINIMUM INVENTORY	: .0000000E+00
TIME INVENTORY TOO LOW	: .0000000E+00	STD.DEV.INVENTORY	: .2605731E+04
TIME INVENTORY TOO HIGH	: .0000000E+00	# OF OBSERVATIONS INV.	: .9183000E+04
% INVENTORY DOWNTIME	: .0000000E+00	SUM UNITS PROCESSED	: .1547534E+06
TOTAL # OF MACHINES	: 1	SUM COST OF PROCESS	: .1054747E+05
SUM SCHEDULED HOURS	: .1541537E+03	COST PER UNIT	: .6815664E-01
SUM MACH.BREAKDOWN HOURS	: .3200000E+02	COST PER SCHEDULED HOUR	: .6842178E+02
SUM PRODUCTIVE HOURS	: .5766267E+02		
% NET UTILIZATION MACH.	: .3740596E+02		
% GROSS UTILIZATION MACH	: .5816446E+02		

MACHINE TYPE 11 : CHAIN-FLAIL DEB.

TOTAL # OF MACHINES : 1	COST PER MACHINE : .1054747E+05
SUM SCHEDULED HOURS : .1541537E+03	COST PER SCHEDULED HOUR : .6842178E+02
SUM MACH.BREAKDOWN HOURS: .3200000E+02	% NET UTILIZATION MACH. : .3740596E+02
SUM PRODUCTIVE HOURS : .5766267E+02	% GROSS UTILIZATION MACH: .5816446E+02

PROCESS NO.13 : LOG FTRAPO

TIME BEGIN OF PROCESS : .4882385E+01	AVERAGE INVENTORY : .9966492E+03
TIME END OF PROCESS : .1676927E+03	MAXIMUM INVENTORY : .3699874E+04
DURATION OF PROCESS : .1628103E+03	MINIMUM INVENTORY : .0000000E+00
TIME INVENTORY TOO LOW : .0000000E+00	STD.DEV.INVENTORY : .6292474E+03
TIME INVENTORY TOO HIGH : .0000000E+00	# OF OBSERVATIONS INV. : .2500000E+03
% INVENTORY DOWNTIME : .0000000E+00	SUM UNITS PROCESSED : .1009106E+06
TOTAL # OF MACHINES : 8	SUM COST OF PROCESS : .1719054E+05
SUM SCHEDULED HOURS : .1302483E+04	COST PER UNIT : .1703542E+00
SUM MACH.BREAKDOWN HOURS: .0000000E+00	COST PER SCHEDULED HOUR : .1319829E+02
SUM PRODUCTIVE HOURS : .8075062E+03	
% NET UTILIZATION MACH. : .6199746E+02	
% GROSS UTILIZATION MACH: .6199746E+02	

MACHINE TYPE 41 : LOG TRAILER

TOTAL # OF MACHINES : 6	COST PER MACHINE : .3948423E+03
SUM SCHEDULED HOURS : .9768621E+03	COST PER SCHEDULED HOUR : .2425167E+01
SUM MACH.BREAKDOWN HOURS: .0000000E+00	% NET UTILIZATION MACH. : .5112352E+02
SUM PRODUCTIVE HOURS : .4994063E+03	% GROSS UTILIZATION MACH: .5112352E+02

MACHINE TYPE 42 : LOG TRAKTOR

TOTAL # OF MACHINES : 2	COST PER MACHINE : .7410744E+04
SUM SCHEDULED HOURS : .3256207E+03	COST PER SCHEDULED HOUR : .4551765E+02
SUM MACH.BREAKDOWN HOURS: .0000000E+00	% NET UTILIZATION MACH. : .9461929E+02
SUM PRODUCTIVE HOURS : .3081000E+03	% GROSS UTILIZATION MACH: .9461929E+02

PROCESS NO.12 : CHIPPING

TIME BEGIN OF PROCESS	: .4888785E+01	AVERAGE INVENTORY	: .6150024E+04
TIME END OF PROCESS	: .1756062E+03	MAXIMUM INVENTORY	: .1211100E+05
DURATION OF PROCESS	: .1707174E+03	MINIMUM INVENTORY	: .0000000E+00
TIME INVENTORY TOO LOW	: .0000000E+00	STD.DEV.INVENTORY	: .3280347E+04
TIME INVENTORY TOO HIGH	: .0000000E+00	# OF OBSERVATIONS INV.	: .1809800E+05
% INVENTORY DOWNTIME	: .0000000E+00	SUM UNITS PROCESSED	: .1547534E+06
TOTAL # OF MACHINES	: 10	SUM COST OF PROCESS	: .3700039E+05
SUM SCHEDULED HOURS	: .1707174E+04	COST PER UNIT	: .2390926E+00
SUM MACH.BREAKDOWN HOURS	: .1600000E+02	COST PER SCHEDULED HOUR	: .2167348E+02
SUM PRODUCTIVE HOURS	: .1374853E+04		
% NET UTILIZATION MACH.	: .8053386E+02		
% GROSS UTILIZATION MACH	: .8147108E+02		

MACHINE TYPE 37 : CHIPPER

TOTAL # OF MACHINES	: 1	COST PER MACHINE	: .8816044E+04
SUM SCHEDULED HOURS	: .1707174E+03	COST PER SCHEDULED HOUR	: .5164116E+02
SUM MACH.BREAKDOWN HOURS	: .1000000E+02	% NET UTILIZATION MACH.	: .1649224E+02
SUM PRODUCTIVE HOURS	: .2815513E+02	% GROSS UTILIZATION MACH	: .2234988E+02

MACHINE TYPE 39 : CHIP TRAILER

TOTAL # OF MACHINES	: 6	COST PER MACHINE	: .9401497E+03
SUM SCHEDULED HOURS	: .1024304E+04	COST PER SCHEDULED HOUR	: .5507052E+01
SUM MACH.BREAKDOWN HOURS	: .0000000E+00	% NET UTILIZATION MACH.	: .8635597E+02
SUM PRODUCTIVE HOURS	: .8845480E+03	% GROSS UTILIZATION MACH	: .8635597E+02

MACHINE TYPE 40 : CHIP TRAKTOR

TOTAL # OF MACHINES	: 3	COST PER MACHINE	: .7514484E+04
SUM SCHEDULED HOURS	: .5121522E+03	COST PER SCHEDULED HOUR	: .4401709E+02
SUM MACH.BREAKDOWN HOURS	: .6000000E+01	% NET UTILIZATION MACH.	: .9023684E+02
SUM PRODUCTIVE HOURS	: .4621500E+03	% GROSS UTILIZATION MACH	: .9140836E+02

LOADING DEVICES
===============

```
TOTAL # OF MACHINES      :              2  SUM OF UNITS HARVESTED  : .2556640E+06
SUM SCHEDULED HOURS      : .3512124E+03  SUM COST LOADER DEVICES : .1740970E+05
SUM MACH.BREAKDOWN HOURS: .1350000E+02  COST PER UNIT            : .6809600E-01
SUM PRODUCTIVE HOURS     : .7431316E+02  COST PER SCHEDULED HOUR : .9914057E+02
% NET UTILIZATION MACH.  : .2115904E+02
% GROSS UTILIZATION MACH: .2500286E+02
```

MACHINE TYPE 32 : CAT 225 LOG LOADER

```
TOTAL # OF MACHINES      :              1  COST PER MACHINE          : .8808517E+04
SUM SCHEDULED HOURS      : .1756062E+03  COST PER SCHEDULED HOUR   : .5016063E+02
SUM MACH.BREAKDOWN HOURS: .1000000E+01  % NET UTILIZATION MACH.   : .1975968E+02
SUM PRODUCTIVE HOURS     : .3469923E+02  % GROSS UTILIZATION MACH: .2032914E+02
```

MACHINE TYPE 33 : CAT 225 W/SLASHER

```
TOTAL # OF MACHINES      :              1  COST PER MACHINE          : .9011740E+04
SUM SCHEDULED HOURS      : .1756062E+03  COST PER SCHEDULED HOUR   : .5131790E+02
SUM MACH.BREAKDOWN HOURS: .1250000E+02  % NET UTILIZATION MACH.   : .2255839E+02
SUM PRODUCTIVE HOURS     : .3961393E+02  % GROSS UTILIZATION MACH: .2967659E+02
```

COMPLETE HARVESTING SYSTEM STATISTICS
=====================================

```
COMPUTER TIME START SIMULATION   DATE: 05-31-87    TIME: 21:53:43
COMPUTER TIME END SIMULATION     DATE: 05-31-87    TIME: 22:33:49

SIMULATION RUN  1 OF  1.
BEGIN OF HARVESTING      : .0000000E+00  END OF HARVESTING         : .1756062E+03
TOTAL # OF MACHINES      :             25  SUM OF UNITS HARVESTED    : .2556640E+06
SUM SCHEDULED HOURS      : .4000175E+04  SUM COST OF SYSTEM        : .1191275E+06
SUM MACH.BREAKDOWN HOURS: .9000000E+02  COST PER UNIT             : .4659532E+00
SUM PRODUCTIVE HOURS     : .2758942E+04  COST PER SYSTEM HOUR      : .6783785E+03
% NET UTILIZATION MACH.  : .6897053E+02
% GROSS UTILIZATION MACH: .7122043E+02
```

------------------------- END OF RUN # 1 OF 1.-------------------------