BEST
AVAILABLE COPY
The views and conclusions contained in this document are those of the author and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the Advanced Research Projects Agency or the U. S. Government.
The purpose of this research project is to apply the systems approach to the tunneling process and to the materials handling function in particular. This report deals primarily with the computer simulation model. A review of simulation methods and languages is presented first to provide background for the choice of a language and the development of the model.

The description of the computer model occupies a large portion of the report. The principles of simulation of each of the unit operations is discussed with the muck generation and the materials handling subsystems receiving the bulk of the attention. The method of testing out the program is also discussed. A guide to the use of the computer program is provided including a flow diagram of the computer logic, a section defining important variables, and a listing of the program.
14. KEY WORDS

<table>
<thead>
<tr>
<th>KEY WORDS</th>
<th>LINK A</th>
<th>LINK B</th>
<th>LINK C</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tunneling</td>
<td>8</td>
<td>10</td>
<td>8</td>
</tr>
<tr>
<td>Materials Handling</td>
<td>3</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>Systems Analysis</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Computer Model</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

INSTRUCTIONS

1. ORIGINATING ACTIVITY: Enter the name and address of the contractor, subcontractor, grantee, Department of Defense activity or other organization (corporate author) issuing the report.

2a. REPORT SECURITY CLASSIFICATION: Enter the overall security classification of the report. Indicate whether "Restricted Data" is included. Marking is to be in accordance with appropriate security regulations.

2b. GROUP: Automatic downgrading is specified in DoD Directive 5200.10 and Armed Forces Industrial Manual. Enter the group number. Also, when applicable, show that optional markings have been used for Group 3 and Group 4 as authorized.

3. REPORT TITLE: Enter the complete report title in all capital letters. Titles in all cases should be unclassified. If a meaningful title cannot be selected without classification, show title classification in all capitals in parentheses immediately following the title.

4. DESCRIPTIVE NOTES: If appropriate, enter the type of report, e.g., interim, progress, summary, annual, or final. Give the inclusive dates when a specific reporting period is covered.

5. AUTHOR(S): Enter the name(s) of author(s) as shown on or in the report. Enter last name, first name, middle initial. If military, show rank and branch of service. The name of the principal author is an absolute minimum requirement.

6. REPORT DATE: Enter the date of the report as day, month, year, or month, year. If more than one date appears on the report, use date of publication.

7a. TOTAL NUMBER OF PAGES: The total page count should follow normal pagination procedures, i.e., enter the number of pages containing information.

7b. NUMBER OF REFERENCES: Enter the total number of references cited in the report.

8a. CONTRACT OR GRANT NUMBER: If applicable, enter the applicable number of the contract or grant under which the report was written.

8b. & 8d. PROJECT NUMBER: Enter the appropriate military department identification, such as project number, subproject number, system numbers, task number, etc.

9a. ORIGINATOR'S REPORT NUMBER(S): Enter the official report number by which the document will be identified and controlled by the originating activity. This number must be unique to this report.

9b. OTHER REPORT NUMBER(S): If the report has been assigned any other report numbers (either by the originator or by the sponsor), also enter this number(s).

10. AVAILABILITY/LIMITATION NOTICES: Enter any limitations on further dissemination of the report, other than those imposed by security classification, using standard statements such as:

   (1) "Qualified requesters may obtain copies of this report from DDC."

   (2) "Foreign announcement and dissemination of this report by DDC is not authorized."

   (3) "U.S. Government agencies may obtain copies of this report directly from DDC. Other qualified DDC users shall request through"

   (4) "U.S. military agencies may obtain copies of this report directly from DDC. Other qualified users shall request through"

   (5) "All distribution of this report is controlled. Qualified DDC users shall request through"

If the report has been furnished to the Office of Technical Services, Department of Commerce, for sale to the public, indicate this fact and enter the price, if known.

11. SUPPLEMENTARY NOTES: Use for additional explanatory notes.

12. SPONSORING MILITARY ACTIVITY: Enter the name of the departmental project office or laboratory sponsoring (paying for) the research and development. Include address.

13. ABSTRACT: Enter an abstract giving a brief and factual summary of the document indicative of the report, even though it may also appear elsewhere in the body of the technical report. If additional space is required, a continuation sheet shall be attached.

   It is highly desirable that the abstract of classified reports be unclassified. Each paragraph of the abstract shall end with an indication of the military security classification of the information in the paragraph, represented as (T). (S), (C), or (U).

   There is no limitation on the length of the abstract. However, the suggested length is from 150 to 225 words.

14. KEY WORDS: Key words are technically meaningful terms or short phrases that characterize a report and may be used as index entries for cataloging the report. Key words must be selected so that no security classification is required. Identifiers, such as equipment model designation, trade name, military project code name, geographic location, may be used as key words but will be followed by an indication of technical context. The assignment of links, rules, and weights is optional.
ARPA Order Number: 1579
Contract Number: H0210011

Program Code Number: 1F10
Principal Investigator and Telephone Number: Jan M. Mutmansky, 801-581-6386

Name of Contractor: University of Utah
Project Scientist or Engineer and Telephone Number: Same

Effective Date of Contract: December 30, 1970
Short Title of Work: Computer Simulation of Unit Operations for Rapid Excavation Systems

Contract Expiration Date: December 29, 1971

Amount of Contract: $33,080

FINAL REPORT
December 30, 1970 - December 29, 1971

This research was supported by the Advanced Research Projects Agency of the Department of Defense and was monitored by the U. S. Bureau of Mines under Contract No. H0210011.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>SUMMARY</td>
<td>1</td>
</tr>
<tr>
<td>ACKNOWLEDGEMENTS</td>
<td>3</td>
</tr>
<tr>
<td>SIMULATION BACKGROUND</td>
<td>4</td>
</tr>
<tr>
<td>Definition of Simulation</td>
<td>4</td>
</tr>
<tr>
<td>Types of Simulation Models</td>
<td>5</td>
</tr>
<tr>
<td>Stochastic Simulation</td>
<td>6</td>
</tr>
<tr>
<td>Deterministic Simulation</td>
<td>7</td>
</tr>
<tr>
<td>Choice of the Simulation Model</td>
<td>9</td>
</tr>
<tr>
<td>DESCRIPTION OF THE MODEL</td>
<td>12</td>
</tr>
<tr>
<td>Model Objectives</td>
<td>12</td>
</tr>
<tr>
<td>Simulation Concepts Used</td>
<td>13</td>
</tr>
<tr>
<td>General Concepts</td>
<td>14</td>
</tr>
<tr>
<td>Muck Generation Subsystem</td>
<td>20</td>
</tr>
<tr>
<td>Materials Handling Subsystem</td>
<td>23</td>
</tr>
<tr>
<td>(1) Cyclic Systems</td>
<td>23</td>
</tr>
<tr>
<td>(2) Continuous Systems</td>
<td>33</td>
</tr>
<tr>
<td>Tunnel Support Subsystem</td>
<td>35</td>
</tr>
<tr>
<td>Environmental Control Subsystem</td>
<td>37</td>
</tr>
<tr>
<td>TESTING OF THE MODEL</td>
<td>39</td>
</tr>
<tr>
<td>Data Collection</td>
<td>39</td>
</tr>
<tr>
<td>Testing Procedure</td>
<td>41</td>
</tr>
<tr>
<td>Present Status of the Program</td>
<td>42</td>
</tr>
<tr>
<td>REFERENCES</td>
<td>44</td>
</tr>
<tr>
<td>APPENDIX A -- USER'S GUIDE TO THE COMPUTER PROGRAM</td>
<td>46</td>
</tr>
<tr>
<td>Definition of Important Variables in the Program</td>
<td>46</td>
</tr>
<tr>
<td>Computer Logic Diagrams</td>
<td>57</td>
</tr>
<tr>
<td>The Computer Program</td>
<td>30</td>
</tr>
<tr>
<td>Figures</td>
<td>Title</td>
</tr>
<tr>
<td>---------</td>
<td>-------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>1</td>
<td>Method of Introducing Probability Functions Into the Computer Program</td>
</tr>
<tr>
<td>2</td>
<td>Simple Examples of Updating in an Event-Oriented Simulation Model</td>
</tr>
<tr>
<td>3</td>
<td>Method of Representing the Tunnel Profile</td>
</tr>
<tr>
<td>4</td>
<td>Diagram of Two Adjacent Switches</td>
</tr>
<tr>
<td>5</td>
<td>Method of Inputting Data From the Characteristic Curve Relating Speed and Tractive Effort</td>
</tr>
<tr>
<td>6</td>
<td>Illustration of the Effective Cross-Section for Continuous Materials Handling Systems</td>
</tr>
</tbody>
</table>
SUMMARY

The prime purpose of this research project is to apply the systems approach and the technique of computer programming in an attempt to improve the process of tunneling by rapid excavation methods. One specific objective is the optimization of the materials handling function for tunneling systems. This report contains information on the methods of simulation on the digital computer, the development of the computer model from the basic concepts, the guide to the use of the program, and the computer program itself.

The review of simulation methods contains a basic definition of simulation and a description of the most useful types of models for digital computers. Primary attention was given to the stochastic and deterministic conceptual models which are used in the simulation of tunneling systems and to methods of updating the computer simulation time variable. In addition, a discussion of computer languages for possible use in a simulation model is presented. The FORTRAN language was chosen for use in the model based primarily upon its wide acceptance and its familiarity to potential users.

The description of the computer model contains a synopsis of the objectives of the model as well as an outline of the concepts used in the simulation program. The discussion of the specific concepts applied is divided into sections dealing with the individual unit operations: muck generation, materials handling, roof support, and environmental control. An additional section deals with the general concepts used.
throughout the program. The most detailed discussions deal with the muck generation and the materials handling sub-systems which are the most complex unit operations to be modeled in the program. The materials handling methods receive the greatest analysis since they are the most complex from a systems standpoint and are the most difficult to simulate.

The attempts at testing the computer program are described in a separate section of the report. At present, the program has been debugged and the logic and behavior of the model during simulation have been studied using data obtained primarily in the field. No attempts have yet been made to check the accuracy of the computer model as this phase of the testing program is scheduled in the near future.

The final section of this report is a users' guide to the computer program, a list and description of the most important variables contained in the program, and a listing of the computer program.
ACKNOWLEDGEMENTS

The investigators would like to sincerely thank the following people and organizations who made contributions to this project. They provided a significant amount of assistance and contact with the real world which is necessary in a research endeavor of this type.

The White Pine Copper Company provided basic operating data on the boring operation at the White Pine Mine. Mr. Joseph L. Patrick, Vice President, Research and Technological Services; Dr. Clifford C. Hanninen, Director of Rock Breaking Research; and Mr. William H. Lane, Superintendent of Boring Machines were especially helpful in this regard and also in providing initial guidance on the project.

Mr. Victor L. Stevens, Mining Consultant, Salt Lake City, provided valuable information on haulage practices in tunneling jobs with which he has been associated.

Jay-Dee Contractors, Inc.; the Metropolitan Sanitary District of Greater Chicago; Peter Keiwit Sons' Company; Climax Molybdenum Company; the White Pine Copper Company; and the S. A. Healy Company all contributed to the project by taking part in arranging for or permitting the investigators to visit their existing tunneling operations in order to get a feel for the problems that exist in the rapid excavation field.
The term "simulation" is used quite frequently in modern technical literature as the methods of computer modeling have become more widely applied and accepted. This section of the report provides a brief outline of the simulation methods and languages and defines the simulation terms used throughout the report.

Definition of Simulation

Simulation has been defined in numerous ways, but a definition that appears in a book by Pritsker and Balintfy (9)* appears to be most applicable here. They have said that "Simulation is the use of a model to study a problem." This simple yet concise definition describes very well the approach used in this project. Our problem is to improve or optimize the unit operations of a rapid excavation system, especially the materials handling subsystem. The model used will be a mathematical one, constructed for use on a digital computer.

The process of modeling is normally carried out in a series of five steps generally referred to as the scientific method. The scientific method of making decisions is often referred to as the systems approach and generally consists of the following steps:

1) Definition and breakdown of the system
2) Construction of a model of the system

*The numbers in parentheses refer to the numbered publications in the REFERENCES section.
3) Testing of the model
4) Solution of the problem
5) Implementation of the solution

This year's work is involved primarily with the first two steps above, beginning with the definition of the problem and continuing through the construction of the computer model of the system.

Types of Simulation Models

In order to simulate any particular process or system, some type of a model is required. Several types of models exist, but only three general types are extensively used. These are the physical models, the analog models, and the conceptual models.

A physical model is a physical model or replica of a system, generally scaled down to a size which is more easily handled than the full-size system. The usual reason for using a physical model is economy of operation. The model can be used to simulate the operation of the actual system without incurring the cost of the full-scale system. Physical models are seldom used in systems analysis but can often be used in other fields of engineering such as in aeronautical evaluation of aircraft design. A physical model is easy to "understand" since looks like the object that it represents or models.

The second class of simulation models are the analog variety. An analog model is a system, such as an electrical or hydraulic circuit, which can be constructed to relate
to another system in such a manner that the behavior of the model can solve problems in the analogous system we are interested in. A typical example of this type of model is the electrical network analyzers used to solve problems related to mine ventilation circuits. Analog models are useful only in certain types of problems, but provide rapid, convenient answers in situations where they apply.

The conceptual models, often called logical or mathematical models, are the prevalent model type and are put to use on a wide variety of problems. For this type of model, the components of the system are represented by mathematical formulas, probability distributions, or numerical data which is used to model the system. A mathematical model is normally written in a computer language so that the massive chore of performing the simulation may be done by computer. Most of the mathematical simulation models fall into the class known as the Monte Carlo methods. In these methods, the general approach is to run and rerun the simulation process as a statistical experiment, measuring the results in order to learn something about the process simulated. The Monte Carlo methods are subdivided into the stochastic and deterministic models.

**Stochastic Simulation.** Stochastic or probabilistic simulation models are used in situations where the elements of the model are probabilistic or random in nature, i.e.,
the elements of the model cannot be predicted with certainty. A stochastic model operates with the probability distributions of each element in the model and empirically determines just what will happen in a particular system by modeling the system under specific sets of conditions. By studying the responses which occur due to changing the controllable variables, the system can be optimized. The principal advantage of this class of model is that it may be used to solve many problems which cannot even be approached using conventional theoretical methods.

**Deterministic Simulation.** Deterministic simulation has been described by Hammersley and Handscomb (1) as an attempt to "exploit the strength of theoretical mathematics while avoiding its associated weakness by replacing theory by experiment whenever the former fails." Deterministic simulation is used to model processes which are governed at least in part by specific laws or rules and which will yield predictable results. For this reason, deterministic simulation has been used to simulate such activities as truck haulage (8), rail haulage (7), and the operation of bucket wheel excavators (14). In these applications, physical laws were used to determine accelerations, speeds, distances, power consumptions, etc., as a function of the operating characteristic curves for the equipment used. Normal practice in a model of this sort is to calculate the required variables at equal intervals of time in an interactive fashion. At each iteration, the theoretical
laws can be used to calculate the desired variables, thus using the power of the digital computer to eliminate the necessity for extensive mathematical development. The model can be used to study an activity based on a theoretical basis and possibly optimize the activity by interpreting the outcome of the simulation experiments.

Deterministic simulators are sometimes further subdivided into event-oriented and time-oriented models. The time-oriented model is perhaps more widely used than the other and often is the easiest to program. In this type of model, a specific increment of time is chosen previous to each computer run. The program updates the simulation by that time increment and calculates all the variables of record at the new time. The calculations are repeated at each incrementation in the time variable. By using the proper logic, any variable can be accurately determined in the simulation if the concepts for simulating that variable are valid.

An event-oriented deterministic simulator is a simulator which does not update its time variable by a constant value but instead, updates the time variable only when specific predetermined events occur in the simulation. The events chosen to result in updating are generally the completion of activities after which decisions must be made.
The principal advantage of this method is that all the variables of record may not need updating during a particular time span. By using the knowledge about specific events in the process to be modeled, only those variables of record which require updating are calculated by the computer. A disadvantage of the method is that it may require more programming work than the time-oriented model for the same system. The choice between the emphasis on time increments or events will depend upon the system to be modeled. It may not be obvious which is the most advantageous before the program is initiated.

Choice of a Simulation Language

One of the first important tasks involved in constructing a computer model is the choice of a medium, i.e., a computer language, in which to write the model. There are numerous computer languages to choose from, including general languages and those specifically designed for application to simulation.

Several general simulation languages are available for use such as GPSS (General Purpose System Simulator) and SIMSCRIPT. These languages are designed to handle variations of standard simulation problems which are often encountered. GPSS, for example, is best suited to problems related to scheduling or to systems involving queueing while SIMSCRIPT is most applicable to inventory and similar problems. Several other languages are available which are designed to study situations of a more specific nature. DYNAMO and
SIMULATE are languages which are used to simulate economic systems. More complete descriptions of these programs can be obtained in the computer language manuals and in books on computer simulation (6).

One language which merits special attention here is GASP II. This is a FORTRAN-based language which is widely applicable and which has numerous advantages. The originators of the language outlined these advantages in their manual on GASP II (9). The most important advantages are related to GASP's base in a common computer language. As a result, the user does not have to learn a new language or obtain a new compiler for his present machine. Thus, two of the major problems related to using a simulation language are eliminated. In addition to these points, GASP is a versatile tool which will have appeal in many simulation analyses.

Another possible language for use in simulation is a general purpose language such as FORTRAN. While this language was not designed for specific use as a simulation language, it is widely used as such and has several advantages as a simulation language. The advantages that GASP II offers to simulation can also be obtained from FORTRAN. Thus, FORTRAN is advantageous since it is widely understood and does not require a special compiler. FORTRAN does present some problems for simulation. These include the lengthy input-output formatting and the lack of inherent debugging aids. However, these disadvantages will not be serious ones if the programmer is quite familiar with the
language and will not effect the program users.

With these facts in mind, the choice of FORTRAN was made for the simulation model being constructed. The main factors affecting this decision are its wide use and its ease of transfer from one machine to another. Most of the important simulation work done in the mining and construction industry has been performed by FORTRAN programs to date. In addition, nearly every digital computer has FORTRAN capability and this will enable the model to be used on the maximum number of computers. To further minimize transfer problems, the authors of the model have attempted to follow USA Standard FORTRAN IV as published by the United States of America Standards Institute (13). This will minimize the machine-dependent statements which will require changing when the program is used on other machines.
DESCRIPTION OF THE MODEL

The computer model presented here is a Monte Carlo type model written in the FORTRAN language using both deterministic and stochastic simulation methods to model the overall tunneling system. The program is written in an event-oriented manner with program updating being accomplished after specific jobs or events are completed. Most of the unit operation submodels are written in stochastic form although the materials handling subsystem contains much in the way of deterministic calculations. Emphasis has been placed upon supplying a number of options within the program to make the program applicable to various types or forms of rapid excavation systems. This portion of the report deals with the model objectives, the description of the simulation concepts, the logic used and the outline of program organization.

Model Objectives

The primary goal of this model is to simulate the common methods of driving a tunnel with a boring machine. To accomplish this goal, it is necessary to think in terms of a general computer program which contains a number of options which allow a user to vary the simulation of the unit operations and the way that they interact during the tunneling process. Primary attention is paid in this model to the materials handling process as this is one unit operation which promises to yield results from a systems evaluation. This conclusion is based upon observations about the
materials handling function creating a bottleneck in the operation (2,3,4,10) and due to the fact that more control may be exercised over the design and operation of the materials handling process than over the other unit operations. For this reason, the most significant programming time and attention was devoted to the modeling of the materials handling function.

To meet the basic objective of studying primarily the materials handling process, models for both cyclic and continuous handling methods have been provided so that either type may be studied. The cyclic systems have been programmed in a fashion which will allow either a track or a rubber-tired haulage system to be modeled providing that the characteristic curves of the driving mechanism are available. For continuous systems, similar accommodations have been provided so that either belt or hydraulic conveyors may be simulated.

Outline of Simulation Concepts Used

The outline of the logic and concepts used in the simulation model will deal first with the general principles or concepts used throughout the program. Afterwards, those concepts which apply primarily to the individual operations will be discussed. For purposes of outlining these specific concepts, the tunneling process will be divided into the following unit operations:

1) muck generation
2) materials handling
3) roof support

4) environmental control

Each of these unit operations will be discussed separately even though they may not be programmed in separate units in the program itself.

**General Concepts.** The first of the discussions on general concepts should perhaps be centered around the method of introducing the necessary probability functions into the program. For versatility and ease of input, all the probability functions which are used in the program are introduced as piecewise linear cumulative probability functions which are sometimes also referred to as cumulative frequency polygons or ogives (11). Figure 1 illustrates the method for reading the cumulative probability functions into the program. Several things should be mentioned here regarding these functions:

1) Neither the abscissa nor the ordinate values must be evenly spaced.

2) The first ordinate value, shown in Figure 1 as CP (1), must equal zero.

3) The final ordinate value, shown in Figure 1 as CP(NPOINT), must equal one.

4) The ordinate and abscissa values are read into the program as pairs and must be arranged in terms of increasing ordinate or cumulative probability values.
Figure 1 - Method of Introducing Probability Functions
Into the Computer Program
5) The number of abscissa and ordinate values read in may be up to 13. If more are necessary, the dimensions of the necessary variables may be easily changed to provide the additional storage space.

Should the user decide that a constant value is to be read into the program for a particular variable instead of a distribution of values, he may do so under the framework of the above method. The procedure that should be used is to read in two ordinate and abscissa values; the first ordinate value should be zero and the second should be one while both abscissa values should be equal to the constant desired for that variable. For example, if the user wished to read in a constant value of 10.5 for a specific variable, he would read in the following values for the cumulative frequency polygon:

\[
\begin{align*}
CP(1) &= 0.0 & T(1) &= 10.5 \\
CP(2) &= 1.0 & T(2) &= 10.5
\end{align*}
\]

The computer would then automatically assign a value of 10.5 to the variable in question every time it is called in the program.

Since the computer program described here is classified as an event-oriented model rather than a time-oriented model, a simplified explanation of an event-oriented model is presented to provide a basic background for users. The computer will store the clock time for all pertinent events in storage. In searching for the activity which should be updated next, the computer will go to the activity with the
shortest clock time. An example of how this would work is illustrated in Figure 2. In Figure 2(a), the status of the five activities assumed to exist in the problem are shown in Gantt chart fashion. The tick marks shown are indicative of specific events such as the completion of certain jobs or tasks. Since Activity 2 exhibits the shortest clock time, the computer must deal with or act upon Activity 2 before it proceeds to the activity with the next shortest clock time. If it is possible to update Activity 2 beyond its present clock time, then this is done as shown in Figure 2(b) and the computer then focuses attention on the new activity which has the shortest clock time, Activity 3.

If the situation requires that the updating of Activity 2 is restrained by another activity, then the start of another cycle of Activity 2 may not begin immediately. A very simple example is presented in Figure 2(c) where the start of Activity 2 is assumed to be restrained by Activity 3 (and only Activity 3). This type of situation may arise because of manpower, space, sequencing, or other limitations. In any case, in this situation Activity 2 must wait until Activity 3 is completed before it can be reinitiated. Thus, the wait is indicated by a dotted rather than a solid line. After Activity 3 is completed, Activity 2 is simulated to completion and Activity 3 then has the shortest clock time and is considered for updating next. In reality, the simulation of an activity may be restrained by numerous other activities of different types. However, the general
Figure 2 - Simple Examples of Updating in an Event-Oriented Simulation Model.
principle of focusing on the activity with the shortest clock time will apply no matter how complex the logic, providing that the model has been properly programmed.

The assignment and utilization of manpower is another consideration which applies throughout the computer model. The method of allocation of manpower was aimed at maximum versatility in the number of men assigned to a particular job. For each task in the tunneling process, an upper and lower limit on the number of workmen assigned is read into the program. The lower limit will reflect the minimum number of men required to safely carry out a task. The upper limit will generally be determined by space, productivity, safety, or other practical limitations of the activity. The computer program will always assign at least the minimum number of men to a job before it is initiated and will assign as many men as it can subject to availability and upper limit restrictions. As more men are assigned to a job, the time to accomplish the job is reduced proportionally. This policy is based upon the assumption that the upper and lower limits of manpower are reasonable and that all men are gainfully occupied on any particular job. As each job is completed, the men assigned to that job are reassigned to other jobs if it is possible. When several jobs require manpower simultaneously, the largest job in terms of manhours required is assigned men first.

On additional general topic of discussion here is the options available for outputting information from the computer. At the termination of each simulation run, a listing
of summary statistics is printed routinely. In order to allow users to determine just what is taking place in the computer program, a log of operations which outputs information on each significant event in the simulation as it occurs can also be optionally implemented. If the log of operations is not desirable or necessary, the user may suppress this series of output statements and the computer will print only the simulation summary statistics.

**Muck Generation Subsystem.** The muck generation subsystem includes all the activities taking place at the face of the tunnel concerned with the operation of the tunneling device. Thus, the muck generation subsystem is concerned primarily with the rate of advance, the inspection and repair and replacement of bits, and the repair and maintenance of the tunneling device. The bits are one of the most important of the considerations in the generation of muck, particularly in large tunnels driven in hard rock. Each bit on the face of the mole must be numbered for the purposes of the computer program. This can be done as shown in Figure 2 of a previous report (5) or in any other suitable manner. After numbering each bit, a time-to-failure probability function is assigned to each bit with the probability being expressed in terms of the feet of advance. In addition, another distribution for the replacement manhours required is assigned to each bit location in order to differentiate between bits in terms of the replacement time required. A separate time-to-failure and repair time
distribution is provided to simulate repair work on the bits which does not require replacement, e.g., welding or other repair work on the bit housings. When a bit reaches the point of failure, it is not replaced immediately but is replaced at the first inspection after the failure has occurred, i.e., at the first opportunity for the failure to be discovered.

The inspection of the bits are assumed in the program to be completed in conjunction with the resetting of the jacks after completion of a normal stroke or on any occasion in which the machine is down for other purposes. It is assumed to be made normally after any integer number of cycles, i.e., after the jacks have been reset a predetermined number of times. If the bits are found to be in condition for more boring, the boring is reinitiated. If failed or worn bits are detected, the replacement operation is simulated before the boring is continued. Some tolerance, inputted in terms of feet of advance, is allowed in the program so that worn bits do not have to be replaced the instant their generated lifetime is assumed to end.

The muck generation subsystem also includes provision for repairs and maintenance which must be performed on the tunneling machine. Those repairs which result in the shutdown of the system are compiled into a time-to-failure distribution. A distribution of manhours required for these repairs is also provided to complete the simulation of this part of the process. In all cases of simulating repairs
associated with the mole, the tunneling machine is assumed to be down in the model and as many crewmen as possible under the circumstances are assigned to the repair action in order to expedite the boring operation. All of the above processes are simulated in a relatively straightforward stochastic manner. This is accomplished by placing each event (bit failure, mole failure, etc.) in an event matrix and testing at each update time to see if any action is required. In this manner, all events in the muck generation subsystem are handled in the same matrix and are scanned at the same time in the program.

The final important element in this subsystem is the rate of generation of muck during the operation of the mole. This process is accomplished in the program through the advance rate distribution and the geometry of the face. The advance rate potential of the tunneling device in feet per hour can be formed into a probability distribution. A random sample from this distribution is chosen to obtain an advance rate which applies for the advance of one stroke length of the machine. This advance rate is then combined with the tunnel cross-section to determine the muck flow rate. An instantaneous advance rate would have been more precise but the result in terms of the simulation would have been negligible, i.e., the long-term production of the machine does not appear to be sensitive to this variable. In the computer program, the simulation of the muck generation subsystem is carried out in the main program and in SUBROUTINE MUCK.
Materials Handling Subsystem. The materials handling subsystem was the most complex portion of the overall model to program. This situation existed as a result of the emphasis placed upon the materials handling process in the model and the physical complexity of some of the muck handling systems. Simulation of both cyclic and continuous systems have been provided for in the model. The computer model is designed in such a manner that the haulage distance is increased as the tunnel is advanced. This is accomplished by keeping track of the advance and increasing the haulage length each time a predetermined advance, DELTH, is attained. This also results in changes in the inby end of the haulage system which must be reflected within the model.

(1) Cyclic Systems. Materials handling using cyclic systems are the most complex methods from a systems standpoint. The cyclic materials handling systems were modeled primarily with single-track haulage systems in mind but a haulage system using rubber-tired vehicles can be accommodated using the same model since the simulation program is designed with this in mind. The initial concern of the cyclic materials handling model to be discussed here is the method of introducing the tunnel grade characteristics into the program. This is accomplished by dividing the tunnel into sections with each section having a constant grade. In case of a tunnel with continuously varying grade, the tunnel profile may have to be approximated by the assumed linear grade segments. The segments are read into the program
in order proceeding from the dumping point and continuing to the face of the tunnel as shown in Figure 3 where a tunnel profile with five sections is illustrated. In all cases, the distances are measured along the center line of the tunnel and changes in azimuth are ignored and assumed to be of little or no consequence in the movement of the haulage devices as they traverse the tunnel. For programming reasons, the sections outside the portal are counted separately from the sections within the tunnel. The program will accommodate a tunnel profile with 100 sections without alteration.

The switches, or switchpoints in the case of rubber-tired vehicles, are assumed to be evenly spaced along the tunnel route. For rubber-tired vehicles, a bored tunnel is not an ideal roadbed and thus it is not usually possible for the vehicles to pass anywhere except where special passing points have been blasted out of the tunnel. For this reason, the simulation model is assumed to be able to model this type of haulage system with passing points at equal intervals along the tunnel. The cyclic materials handling sub-model simulates the movement of the vehicles on a switch-to-switch basis in SUBROUTINE TRANS. For example, assume that a train is waiting on the inbound side of Switch B of Figure 4 on one of its empty trips to the face of the tunnel. When the track is cleared, SUBROUTINE TRANS controls the movement of the empty train by calling SUBROUTINE MOTION which simulates the motion of the train from Switch B to Switch A. In order to obtain clearance to use the section
Figure 3 - Method of Representing the Tunnel Profile
Figure 4 - Diagram of Two Adjacent Switches
of track between the two switches, the track section must
be clear and the empty train must have priority as deter-
mined by the decision or control function in SUBROUTINE TRANS.
The decision as to which train has priority to a particular
section of track is made on a first-in-first-out basis.
Adjacent switches, such as Switch A and Switch B, are al-
ways considered together in determining this priority. For
example, if an inbound train reaches Switch B before an
outbound train reaches Switch A, then the inbound train has
the priority for the use of the connecting track and it com-
pletes its movement to Switch A before the outbound train
can initiate its move from Switch A to Switch B. By con-
sidering all the switches simultaneously, SUBROUTINE TRANS
can control the operation of all the trains in an event-
oriented fashion while SUBROUTINE MOTION simulates the ac-
tual switch-to-switch movements.

SUBROUTINE MOTION handles the motion of the train in
an event-oriented deterministic fashion based upon the physi-
cal laws of motion. One of the first publications dealing
with this basic simulation method for haulage systems was
introduced by Nelson (7). For this application, his basic
deterministic approach has been changed to one which does
not make use of equal time increments but instead concen-
trates upon specific events in the movement of the train as
its travel is simulated. The basic physical law used is
Newton's second law of motion which for the case of a rolling
vehicle (12) can be written as:
\[ a = \frac{(T - F_f - F_g)G}{W_L + W_c + W_m} \]

where:
- \( T \) = tractive effort of the driving wheels in pounds
- \( F_f \) = force required to overcome friction in pounds
- \( F_g \) = force required to overcome the gravity component in pounds
- \( W_L \) = weight of the locomotive in pounds
- \( W_c \) = weight of the cars in pounds
- \( W_m \) = weight of the muck in pounds
- \( a \) = acceleration in feet per second per second
- \( g \) = acceleration of gravity, 32.2 feet per second per second

Since the tractive effort does not remain constant for changes in the speed of the tractive unit, some method of applying the formula above must be used so that the changes in the speed and the tractive effort are reflected in the program. To accomplish this, the characteristic curve of the tractive unit which relates its speed and tractive effort must be made available for use in the computer model. A number of selected points along this characteristic curve are read into the computer program as shown in Figure 5 for a hypothetical two-speed locomotive unit. The program then assumes that the characteristic curve is linear between succeeding points so that the effect is an approximation of the actual curve by a piecewise linear function defined by the points selected for input. The degree of simulation
accuracy required in the runs will dictate the number and spacing of the points selected. At present, the proper variables in the computer program are dimensioned to allow reading in up to 30 points along this characteristic curve.

The use of the tractive effort-speed curve in SUBROUTINE MOTION is carried out on an iterative basis using certain specified events to indicate the need for recalculation of the variables of motion. Normally this is done based upon the assumed linear segments of the characteristic curve as follows. A train (or other vehicle) which is starting from rest is assumed to do so at the average tractive effort value for the first assumed linear segment along the curve in Figure 5, i.e., at a tractive effort value of \([\text{TE}(1)+\text{TE}(2)]/2\). An acceleration is calculated based upon this tractive effort and the train moves until the acceleration results in the train achieving the speed at the end of the first linear segment, \(S(2)\). When this occurs, a new average tractive effort value, \([\text{TE}(2)+\text{TE}(3)]/2\), is applied for the period of time required for the train's speed to reach \(S(3)\), and so on. This iterative method continues until the train reaches its maximum allowable speed or until it reaches a new grade section in the tunnel. At the maximum speed, the train's speed is not permitted to accelerate any further and it continues with a constant velocity. When a change in grade occurs, this changes the gravity force component and thus the acceleration is automatically recalculated within the program even though the
Figure 5 - Method of Inputting Data From the Characteristic
train has not speeded up to the next input point on the tractive effort-speed curve. To perform this calculation, the computer will interpolate to determine the current value of the tractive effort and average this value with the next higher tractive effort value read in along the curve. This average will then be used to calculate the initial acceleration on the new grade. It is assumed in this method that the mass of the train is a point mass located at the locomotive unit. This assumption will not effect the simulation significantly unless the tunnel profile is changing rapidly and considerably in grade, a situation which does not occur in rapid excavation tunneling jobs.

The dumping, loading, and switchout times for the cyclic materials handling systems are handled separately. The loading times are determined by the interaction of the muck generation and the materials handling systems. The cars of a train are loaded by the action of the mole as it advances into the face. Thus, the loading time for each car is a stochastic function which is dependent on the rate of advance which is generated in the program for the tunneling device. The dumping time of each train is also determined stochastically to allow for the variations which will certainly occur in the process. Thus, a dumping time cumulative probability function must be read into the computer as illustrated in Figure 1. The switchout time mentioned above is the name given here to
the time required for an empty and a loaded train to switch out under the gantry conveyor using the switch normally located directly behind the conveyor. This process may be deterministically simulated under ideal circumstances. However, operators often use incoming trips to haul the tunnel supplies and these must be unloaded when the train reaches the face area. Thus, it is necessary to use a probabilistic approach on the switchout time in order to reflect the variations in time due to the necessity of unloading the supplies at the face. This can be done by utilizing a bimodal distribution, the first or shortest mode reflecting switchout times where no supplies are unloaded and the second mode related to times necessary to complete the switchout operation when the unloading time is included in the switchout time. When unloading of supplies is not a problem, a unimodal distribution may be suitable for this variable.

At the start of a simulation run, the trains are positioned behind the tunneling device in such a manner that they are spaced one switch apart. This setup places the trains in as favorable a state of readiness as can be achieved in the tunnel. This initial setup scheme was chosen since it was felt that the trains would be in a ready state during a normal startup of a tunneling operation, e.g., at the beginning of the first shift of the simulation.
Continuous Systems. The simulation of a continuous materials handling system is simple in comparison with the cyclic systems. To model the actual transport of the muck, the concept of effective cross-section is defined as the area occupied by the broken muck in a cross-section of the material flow when the materials handling method is operating at its maximum capacity. The value would be a constant for any system and would be independent of the flow velocity and the material density. The effective cross-section for a belt conveyor and those for a hydraulic or pneumatic system would differ as shown in Figure 6. In all cases, however, when the effective cross-section is multiplied by the velocity of transport and the proper density value, the result should be the maximum mass flow rate of the muck for the specific materials handling system used. Care should be taken in expressing the value of the density as the effective cross-section of the belt is based upon the profile of broken rock while those for the systems using pipe are based upon solid material. Once the muck has entered the flowstream, the actual transport can be easily simulated. This can be modeled deterministically based upon the flow velocity and the length of the haulage system.

One of the most important considerations in the materials handling subsystem for continuous systems is
THE EFFECTIVE CROSS-SECTION OF A CONVEYOR IS THE CROSS-SECTIONAL AREA OF THE MUCK WHEN THE CONVEYOR IS OPERATING AT ITS MAXIMUM CAPACITY (INDICATED BY THE CROSS-HATCHED AREA ABOVE)

THE EFFECTIVE CROSS-SECTION OF A HYDRAULIC OR PNEUMATIC CONVEYOR IS THE CROSS-SECTIONAL AREA OCCUPIED BY THE MUCK WHEN THE CONVEYOR IS OPERATING AT ITS MAXIMUM CAPACITY (INDICATED BY THE SHADED AREA ABOVE)

Figure 6 - Illustration of the Effective Cross-Section for Continuous Materials Handling Systems
the interaction with the muck generation subsystem. The primary function is to regulate the flow of muck into the materials handling subsystem. If the muck generation rate is greater than that which can be handled by the materials handling subsystem, then the rate of muck generation is slowed to permit the materials handling subsystem to accommodate the muck. This would, of course, slow the advance of the overall system. When the materials handling device can handle the flow of muck, the muck generation subsystem can then be allowed to operate in an unconstrained manner.

At the other end of the materials handling subsystem where the muck is dumped, another possibility for interruptions in the flow of muck occurs. This can arise because of an interaction with another transport system, because of the condition of a holding device, or due to numerous other factors which can affect the flow of material from the tunnel. Because of the varied nature of the possibilities which may be encountered on this end, no specific delay has been programmed. However, if a specific type of delay is expected to occur at the discharge point of the continuous materials handling system, this can be added to the model in the manner which will correctly affect the simulation of this characteristic of the system.

**Tunnel Support Subsystem.** The support function for tunneling is quite variable because of the nature of the
geologic materials through which tunnels are driven. Many excavations are provided with support in a fashion which may be considered to be cyclic, i.e., a cycle of jobs is carried out to advance the support by one "set." In the computer program, the simulation of such a cyclic method is carried out by assigning a probability distribution to the number of manhours required to advance the support through a single cycle of support work. This makes the interrelationship between the muck generation and the support subsystems an easy one to handle in the model. The time to advance the tunneling device the length of one set can be compared to the time required to complete one cycle of support and tunnel advance can be limited to the speed of the slower process. This procedure will permit the support subsystem to keep up and provide the support which is required to safely advance the tunnel.

Other methods of providing support in a tunnel are much less cyclic in nature and vary significantly from the methods suggested above. Examples of this type of support methods include roofbolting and guniting. For methods which are not cyclic in nature, the simulation must be handled differently. This can be done, however, within the framework of the cyclic support methods outlined above by shortening the length of a "set" to a value which is short compared to the stroke of the tunneling machine. In this manner the simulation will approach the installation of support which occurs continuously rather than one which causes the support to be advanced
in spurts. As an illustration, the action of installing roof bolts may be modeled by inputting the probabilistic number of manhours required to advance the support a relatively short distance along the tunnel e.g., one foot. As the support is advanced, the advance of the mole can be checked to insure that it does not exceed the advance of the available support exactly as was done for the cyclic systems. Since the support is not advanced in long increments, however, the model is realistic in relation to the actual system.

**Environmental Control Subsystem.** The primary tasks in providing an adequate environment throughout the tunnel normally involve extension of the ventilation system and maintaining a water supply if used on the cutting head to aid in dust abatement. The process of supplying these auxiliary needs will normally be performed at specific intervals of tunnel advance. The installation of the ventilation tubing is normally undertaken at intervals of advance equal to the length of the tubing sections. The simulation of the installation is performed stochastically by providing a probability function for the number of manhours required to install one length of the ventilation tubing. Provision has been made for allowing the tunnel to advance by more than one length of the tubing before the installation of the tubing must be undertaken. A similar method is applied to the process of maintaining the supply of water at the face. A
A separate probability distribution for the demands of this system is read into the computer for each run.

Additional auxiliary services may be necessary at the face which may or may not be directly related to the environmental control function. These may include such functions as the advancing of the track, the extension of the sump lines, or other jobs which must be carried out on a periodic basis. These processes may be simulated within the environmental control subsystem just as those functions directly connected to the environment in the tunnel. A third periodic process of this type can be simulated by using the probability distribution already provided within this subsystem. Other functions of similar nature can be handled if necessary by providing additional distributions and using the framework of logic inherent in the environmental control subsystem.
TESTING OF THE MODEL

The testing of the computer model was only partially completed at the end of the project year. The initial testing phase concerned with checking the logic of the program and its macro behavior was accomplished using data obtained mainly in the field. However, more exhaustive evaluation and development was scheduled for the second year of the project and is yet to be undertaken.

Data Collection

In the testing of the computer program, as much data as possible from the field was used to supply the computer program. In the muck generation subsystem, data obtained through the courtesy of the White Pine Copper Company was used in the simulation. The bit life distributions were compiled from actual bit records kept by the mine personnel during the period of experience with their Robbins machine. The bit lives available were formed into a histogram for each bit on the head of the machine. The histograms were formed from the raw data and then converted into cumulative frequency diagrams by a computer program written for that purpose. The repair times for each of the bits were not determined from actual data but were instead estimated by company officials. The repair times for each of the bits on the machine were individually assumed to be constant values but higher constant repair times were assigned to
bits near the periphery of the cutting head where working conditions were more difficult due to space restrictions. The time-to-failure and repair distributions for the tunneling device were determined by reconstructing the operating record from shift reports and obtaining the individual times between failure and the number of manhours required to complete each repair. These were then formed by computer into the necessary distributions for use in the computer program. The data for testing of the cyclic materials handling subsystem was not hard to gather, although actual field data was not available for some of the variables. A tunnel profile with many grade changes was hypothesized for use in the test. Trains corresponding to present practice were assembled for the simulation. Three two-speed diesel locomotives with a weight of fifteen tons were selected. Eight fifteen-ton cars with an empty weight of three tons were chosen for each train. The distribution of the weight loaded in each of the trains was assumed to be normal with a standard deviation equal to 5% of the mean value. A bimodal switchout time distribution was hypothesized to indicate a practice of unloading supplies from the incoming trains. The distribution of dumping time was estimated from one contractor's experience on a previous tunneling project.

Data for the tunnel support and the environmental control subsystems was obtained from available records on the White Pine system. The individual samples were collected by studying the shift reports and extrapolating as best as
possible the number of manhours spent during specific activities involving each of the subsystems. By collecting information on a large number of occurrences, distributions of the manhours required for specific advances of these two subsystems were formed.

Testing Procedure

The initial test of the program was made with the idea of eliminating the programming problems in the model, i.e., eliminating the bugs and errors in logic in the model. This was accomplished simply by attempting to run the program and check the validity of the results. The most complex portion of the program was the materials handling subsystem and this subsystem was the most difficult to debug. When the obvious debugging problems were out of the way, the program was then checked to be certain it was operating logically and outputting data in the log of operations which agreed with calculations made by hand. This procedure probably did not result in testing all the possible branches of the program even though an attempt was made to cover as much of the logic as possible. After several problems were eliminated, the program seemed to be at least superficially correct and free of obvious bugs.

No attempt was made to test the accuracy of the simulation model in terms of the overall results as this step in the testing procedure was planned for the second year of the project. The testing of the accuracy of the model was to be
undertaken using the data obtained at White Pine as input to a simulation run which would model the tunneling operation for about one month's time. The results of the simulation in terms of the tunnel advance and the times spent in the various unit operations would then be compared with the actual values of these variables obtained from tunneling records for the time period in question. Attempts could then be made to adjust or improve the computer program in areas where its performance was concluded to be unsuitable.

**Present Status of the Program**

Since the development of the program is not complete at the present time, users should recognize that parts of the model may still be in rather unfinished form in the program. In particular, the program may still contain bugs which have not been detected. In addition, options which would make the program more versatile and useful may not be included due to the limited period of use of the model. As an example, it was hoped to expand the program to include the logic for systems using both cyclic and continuous materials handling systems, the cyclic system being applied to the handling of supplies while the continuous system was applied to the handling of muck. Such logic does not presently exist in the model. These inadequacies are to be taken care of during the latter stages of development and use of the program. At present, however, the program is still in a state of
development and testing and should not be considered a finished product.

One of the most important aspects of the testing of the model which has not been completed is the testing of the accuracy of the program in modeling actual tunneling situations. For this reason, the fact that the model will complete a run and output data is not sufficient reason to have complete confidence in the results. Inaccuracies may be caused by bugs in the program or by the assumptions of the model not being valid for all or some of the conditions under which the model is to be applied. Users should note these warnings before making use of the program.
REFERENCES


APPENDIX A

USERS' GUIDE TO THE COMPUTER PROGRAM

Definition of Important Variables in the Program

This section contains the definition of the most important variables and the units in which they are expressed in their usage in the program. The variables which must be input into the program are also defined in this list. To prepare an input data deck, a user must refer to the main program and SUBROUTINE TRANS where all the data is input. The data prepared for the main program should appear first in the data deck while the data for SUBROUTINE TRANS follows.

All the input variables are defined in this list, which is alphabetized for convenience in locating specific variable names. Users may refer to the program for the order and format information on variables and then to this list for the definition.

ACCFC -- available accelerating force, tons

ACCMAX -- maximum acceleration rate allowed in the tunnel, feet per second per second

ACCR -- acceleration rate of a train, feet per second per second

ACT(I) -- the reduced time to complete activity \( i \) after redistributing the manpower.

ACTIM(I) -- the time required to complete the \( i \)th activity

ADRT -- the tunnel advance rate, feet per hour

AFT -- feet of advance required to load one train

AVAMH -- manhours available for the support function

AVATF -- available tractive effort of a locomotive at its current speed, pounds
BINSMH -- manhours required to inspect the bits and regrip the mole

CAPMH -- the capacity of the continuous materials handling system, tons per minute

CCTM -- the time in minutes required to generate one trainload of muck

CF(I,J) -- the \( i^{th} \) ordinate value read in from the \( i^{th} \) cumulative probability curve in the main program

\[ I = 1 \text{ to } NBITS \] correspond to the probability distributions for the time-to-failure of the bits in feet or operating hours.

\[ I = NBITS + 1 \text{ to } 2\times NBITS \] correspond to the probability distributions for the manhours required to replace the bits.

\[ I = 2\times NBITS + 1 \] corresponds to the probability distribution for the time between bit repairs, hours.

\[ I = 2\times NBITS + 2 \] corresponds to the probability distribution for the time between mole repairs, hours.

\[ I = 2\times NBITS + 3 \] corresponds to the probability distribution for the time between repairs of the third (optional) equipment, hours.

\[ I = 2\times NBITS + 4 \] corresponds to the probability distribution for the advance rate, feet per hour.

\[ I = 2\times NBITS + 5 \] corresponds to the probability distribution for the manhours for repair of the bits.

\[ I = 2\times NBITS + 6 \] corresponds to the probability distribution for the manhours required for repair of the mole.

\[ I = 2\times NBITS + 7 \] corresponds to the probability distribution for the manhours required to repair of the third (optional) equipment.

Altogether, \( 2\times NBITS + 7 \) probability distributions are read into the main program.

CFD(I) -- the \( i^{th} \) ordinate value read in from the cumulative probability curve for the dumping time.

CFL(I) -- the \( i^{th} \) ordinate value read in from the cumulative probability curve for weight of the muck in a car.

CFR(I) -- the \( i^{th} \) value of the cumulative probability read from the support requirement function.

CFS(I) -- the \( i^{th} \) ordinate value read in from the cumulative probability curve for switching time.

CROSEC -- cross-sectional area of the tunnel, square feet.

CT(I) -- the \( i^{th} \) abscissa value read in from the cumulative probability curve for dumping time, minutes.
CTIME -- clock time from the start of the simulation, minutes

CTLOC(I) -- total or clock time of the _ith_ locomotive in minutes

CTM(I) -- the _ith_ value of the ACT(I) array if arranged in ascending order

D1 -- the distance in feet from one stop to the next stop of a train excluding the distance required to stop

D2 -- distance in feet to the end of the present grade section

D(I) -- horizontal length of section _i_ of the tunnel profile in feet

DECEL -- maximum deceleration rate allowed in the tunnel, feet per second per second

DELTCH -- increment added to the tunnel length as the face advances, feet

DISTR(I) -- distance traveled by the _ith_ locomotive in feet

DISW -- distance between two switching points in feet

DMS -- current distance between the switch closest the face and the next switching point, feet

DS(I) -- distance from the dumping station to the _ith_ switch

DSTOP(I) -- distance required for the _ith_ locomotive to stop, feet

FCAR -- the friction coefficient of each mine car in pounds per ton

FLOCO(I) -- friction coefficient of locomotive _i_ in pounds per ton

FRFC -- force required to overcome the frictional resistance, pounds

FTA(I) -- the _ith_ abscissa value read from the support requirement curve, manhours per foot of advance

G(I) -- present grade of section _i_ of the tunnel profile

GAMMA -- specific weight of the muck in the solid, pounds per cubic foot

GFC -- force required to overcome the grade resistance, pounds

GLEFT -- distance in feet remaining to be traveled in the track section
HAUL -- the current haulage length in feet

HRPSH -- working hours per shift, i.e., the total shift time minus travel and other idle time

ICYCLE -- the variable which indicates the type of material handling system
ICYCLE = 0 indicates a continuous system

IDE(I) -- queueing number of the i_th locomotive while waiting empty at the dumping station to enter the tunnel; IDE(I) = 0 means the i_th locomotive is not in the queue

IDEOS -- the variable which indicates that the simulation is to terminate; IDEOS = 1 indicates the termination

IDL(I) -- queueing number of the i_th locomotive as it waits to dump its muck at the dumping station; IDL(I) = 0 means the i_th locomotive is not in the queue

IDLOAD -- indicates whether any trains were loaded or not; IDLOAD = 1 indicates trains have been loaded

IL -- the number of the locomotive which has the shortest clock time but which is awaiting the movement of another locomotive

ILC -- the number of the locomotive which has the same clock time as that of the main program

ILS -- controls the input statements in SUBROUTINE TRANS; ILS = 0 means no simulation is performed

ILWTID -- the variable which indicates the beginning of the simulation; ILWTID = 1 indicates the beginning

IMAN -- number of men currently available

INLC -- the number of the loaded locomotive at the loading point

INSPM -- the number of men required to inspect the bits

IR -- the subscript used to obtain the repair manhours for ITEM

ITEM -- the number of the unit which has the shortest life

KK -- the next lower speed point on the characteristic curve

KMAX -- number of points on the characteristic curves of the locomotive at which input data will be read
KOUNT -- the number of bits which need to be replaced

LC(I) -- the queuing number of the trains in the ILC list

LCLAS -- number of points read in from the cumulation frequency function for the weight of muck in one muck car

LIL -- the variable which retains the numbers of the locomotives which were in the previous IL list

LL(I) -- the switch on which the ith locomotive is located

LLW(I) -- the number of the locomotive in the ith spot in the LIL queue

LOAD(I) -- indicates the status of the ith train
    LOAD(I) = 0 indicates the train is empty
    LOAD(I) = 1 indicates the train is loaded

LOGPRT -- print option variable
    LOGPRT = 0 indicates that the complete log of operations is printed
    LOGPRT # 0 indicates that only the summary of the simulation is printed

LS(I) -- the variable which indicates the status of the ith switch
    LS(I) = 0 indicates the switch is empty
    LS(I) = 1 indicates the switch contains an empty train
    LS(I) = 2 indicates the switch contains a loaded train
    LS(I) = 3 indicates the switch contains both an empty and a loaded train

LW(I) -- the number of the ith locomotive in the clock time queue

LWTID -- indicates whether or not there is an empty train at the loading point; LWTID = 0 indicates no empty train

MAD -- number of men available to be reassigned when a repair activity is completed

MAN(I,J) -- the variable which stores the upper and lower limits on the number of men assigned to each activity
    I = 1 corresponds to the lower limit
    I = 2 corresponds to the upper limit
    J = 1 to NBITS corresponds to the limits of manpower for the replacement of the bits
    J = NBITS + 1 corresponds to the limits of manpower for the repair of the bits
    J = NBITS + 2 corresponds to the limits of manpower for the repair of the mole
    J = NBITS + 3 corresponds to the limits of manpower for the repair of the third (optional) equipment
MANAW(I) -- the number of men assigned to the i\text{th} job

MAXSHT -- maximum number of shifts that the simulation is to be run

MH -- variable which indicates which option was employed in reading in the muck generation cumulative frequency curves
MH = 0 indicates the abscissa values are in terms of hours
MH \neq 0 indicates the values are in terms of the feet of advance

MM -- grade section number which train NL is presently traversing

ML -- number of the locomotive currently being moved

MNBITL -- lower limit on the number of men required to repair bits

MNBITU -- upper limit on the number of men required to repair bits

MOTM -- the time in minutes required for the hauling of the muck generated by TEMSTR

MREST -- cumulative number of men who spent idle time during the computer run

MSS(I) -- number of the locomotive occupying the i\text{th} switch

MTB -- number of men who are reassigned when a repair activity is completed

NACF -- the number of events to be simulated in the muck generation subsystem in addition to the events related to bit replacement

NBITS -- the number of bits

NCARS -- number of muck cars assigned to each train

NCF -- total number of cumulative frequency diagrams read into the muck generation subsystem

NCLAS(I) -- the number of points read in for the i\text{th} cumulative probability function of the muck generation subsystem

THIRD1 -- cumulative time spent in doing the third event, minutes

NCREW -- the number of men in the crew

NDC -- number of points read in from the cumulative frequency function for the dumping time

NEVENT -- the number of separate repair activities currently being performed
NL -- locomotive number presently being simulated
NLDL -- number of loaded trains waiting at the dumping station to dump
NLDE -- number of empty trains at the dumping station
NLDL -- the number of loaded trains at the dumping point
NLOCO -- number of locomotives
NRBG -- the number of points read in from the cumulative probability curve for the support function
NS -- the switch from which locomotive NL is moved
NCS -- number of points read in from the cumulative frequency function for the time to switch trains behind the mole
NSCF -- the number of time-between-repair cumulative probability functions read into the muck generation subsystem
NSDP -- number of sections of the haulage profile between the dumping point and the tunnel mouth read into the program
NSECS -- number of sections of the haulage profile within the tunnel read into the program (after input, NSECS is the number of sections in the tunnel profile at the time of simulation)
NSW -- number of switching points currently in the haulage system
NSHIFT -- the number of shifts simulated so far in the current run
OTRD -- distance in feet that the train overtravels
PWT -- the time the continuous materials handling system can operate before a breakdown, minutes
RADIUS -- radius of the tunnel, feet
REQMH -- required manhours of support work for one foot of advance
REQTF -- required tractive effort, pounds
RESTMH -- cumulative number of idle manhours
S(I,J) -- speed of the jth locomotive at the jth point on its characteristic curve
SAFT -- cumulative length of advance since the last value of DELTH was added to Haul
SCCTM -- the cumulative time in minutes to advance by TEMSTR

SGL(I) -- distance in feet from the ith switch to the inby end of the track section on which the switch exists

SLEFT -- distance in feet to the next switch point

SP -- former speed of the train, feet per second

SPEED(I) -- velocity of the ith locomotive, feet per minute

SSCC -- incremental time in minutes that a train waits for the completion of another event

ST(I) -- the ith abscissa value read in from the cumulative probability curve for switching time, minutes

STROKE -- stroke of the mole, feet

SWTTIM -- the cumulative delay time in minutes due to the support subsystem

T(I,J) -- tractive effort of the ith locomotive at the jth point on its characteristic curve

T1 -- the time in seconds required to travel the distance D1

T2 -- time in seconds to reach the end of the present grade section

TBELT1 -- operating time of the continuous materials handling system, minutes

TBELT2 -- delay time due to the continuous materials handling system, minutes

TBELT3 -- downtime of the continuous materials handling system, minutes

TBIT1 -- cumulative working time of the bits, minutes

TBIT2 -- cumulative idle time of the bits, minutes

TBIT3 -- cumulative time the bits are under repair, minutes

TBIT4 -- cumulative time the bits are under replacement, minutes

TBIT5 -- cumulative time the bits are under inspection, minutes

TDUMP(I) -- dumping time in minutes of the ith locomotive during the last dumping cycle

TEMPWT -- the weight in tons of the portion of the material remaining to be loaded in the current train
TEMSTR — portion of the stroke which remains to be completed

TFT — the number of feet the mole can advance before being stopped

TFTA — the incremental number of feet the mole is to be advanced

THIRD1 — time the third (extra) subsystem spends working, minutes

THIRD2 — time the third (extra) subsystem spends in waiting, minutes

THIRD3 — time the third (extra) subsystem undergoes repair, minutes

TIMAX — maximum clock time in minutes that the simulation is to be run

TIME(I) — time required in minutes for the i-th locomotive to get from one switch to the next minus the value of TPA(S(I)) or TSTOP(I)

TLOAD(I) — loading time in minutes of the i-th locomotive when it was last loaded, minutes

TLOC1(I) — cumulative time the i-th locomotive spends in the loading process, minutes

TLOC2(I) — cumulative time the i-th locomotive spends in the dumping process, minutes

TLOC3(I) — cumulative time the i-th locomotive spends in motion, minutes

TLOC4(I) — cumulative time the i-th locomotive spends waiting, minutes

TMH — manhours required to advance by TFTA

TMOLE1 — cumulative working time of the mole, minutes

TMOLE2 — cumulative idle time of the mole, minutes

TMOLE3 — cumulative time the mole is under repair, minutes

TNL — maximum length of advance of the tunnel in feet for the simulation run

TOLIT — the tolerance placed upon the repair starting times in minutes; i.e., when one repair action is initiated, the potential repairs are checked and are also initiated if they are within the tolerance time of requiring repair
TPASS(I) — time required for the $i$th locomotive to travel through a switch without stopping, minutes

TPM — the muck generation rate in tons per minute

TSEC — the current length of the $i$th section which has been driven and added to the variable HAUL

TSTOP(I) — time required for the $i$th locomotive to decelerate and stop on a switch, minutes

TSUPPT — cumulative time expended for support activities, minutes

TSW — the time in minutes required to switch out the loaded train at the loading point

TTM — the time in minutes that the mole can advance before being stopped

TUNNEL — the length of tunnel bored to the present, in feet from the portal

TV(I,J) — the $j$th time or other abscissa value read in from the $i$th cumulative probability curve in the main program in units of feet or operating hours (for a definition of the meanings of each of the values of 1, see the variable CF(I,J))

VELMAX — maximum velocity allowed in the tunnel, feet per second

WAITIM — cumulative idle time of the muck generation sub-system in minutes

WTCAR — weight in tons of each muck car while empty

WTD — cumulative weight of muck dumped, tons

WTG — cumulative weight of muck generated, tons

WTIM — the time in minutes to move an empty train to the loading point

WTL(I) — the $i$th abscissa value read in from the cumulative probability curve for weight of muck in a car, tons

WTLDG — the weight in tons of the load to be generated by TFTA

WTLOAD(I) — weight of the muck in the $i$th train in tons

WTLOC(I) — weight of locomotive $i$ in tons

WTMUCK — the weight of muck in tons to be loaded in one train

WTTRN(I) — weight in tons of the $i$th locomotive and its empty cars
WWTM -- incremental time in minutes that a train waits for the completion of another event

XX(I) -- the abscissa value as determined from SUBROUTINE CALCUM
Computer Logic Diagrams

The Computer logic diagrams of the main program and two of the seven subroutines, SUBROUTINE MOTION and SUBROUTINE TRANS, appear on the following pages. The remainder of the subroutines are not represented in this section since they perform relatively simple functions for which the logic diagrams were considered unnecessary. The diagrams presented are not intended to be a detailed flowchart of all the calculations and manipulations that take place in the computer program. Instead, they are meant to convey the macro logic of the simulation and way that it fits together in the model. Most of the variables which appear in the logic diagram are identified in the previous section of this Appendix. In the logic diagrams, two types of offpage connectors are used. The connectors appear as small circles with numbers or letters enclosed. Connectors containing numbers indicate the actual program statement at which the connection is to be made. This gives the reader one extra bit of help in following the program using the logic diagram. The connectors containing letters are those for which no exact statement number to which the program proceeds could be named.
MAIN PROGRAM

START

READ DATA FOR MAIN PROGRAM

INITIALIZE VARIABLES

CALL CALCUM

CALL TRANS

CALL MUCK

CALL SUPPRT

SCCTM = 0.0

TFTA = TFT

TFT

CALL MUCK

TEMPWT = WTMUCK

TEMSTR = STROKE

FIND SHORTEST LIFE

DETERMINE NEVENT, TTM, TFT, TMH

CALCULATE TTM, TMH, WTLG, TFTA

CALL SUPPRT

SCCTM = 0.0

760
MAIN PROGRAM (CONT'D)

760

ICYCLE

YES

FTAD = TEMSTR

EQ.0

NO

CALL TRANS

LWTID = 0

YES

CALL TRANS

DETERMINE CTIME, TBIT2, TMOLE2, THIRD2

LWTID = 1

WTLDG-TEMPWT

WTLDG = WTLDG - TEMPWT

TEMPWT = 0.0

CALL TRANS

ILWTID = ILWTID + 1

IDLOAD = 0

CALL TRANS

TEMPWT = TEMPWT - WTLDG

FIND LOCO JJ AT NSW

COUNT TLOC1(JJ), CTLOC(JJ)

TEMPWT = 0.0

ILWTID = ILWTID + 1

IDLOAD = 0

CALL TRANS

520

IDEOS

- EQ. 1

CALL MUCK

TEMPWT = WTMUCK

520

YES

IDEOS

- EQ. 1

CALL MUCK

CALL MUCK

TEMPWT = WTMUCK

413

SHFTMH - TMH

CALCULATE SHFTMH

NSHIFT = NSHIFT + 1

412

0
MAIN PROGRAM (CONT'D)

412

CALCULATE SHFTMH, TUNNEL, TBIT1, TMOLE1, THIRD1, CTIME, AVAMH

REQMH - AVAMH

0 -

CALCULATE TSUPPT
TFT = TFT - TEMSR
TEMSTR = STROK'

LOGPRT - EQ. 0

YES

PRINT BORING INFORMATION

NO

DETERMINE BNSMH, CTIME, TMOLE2, THIRD2

LOGPRT - EQ. 0

YES

PRINT BIT INSPECTION DATA

NO

DETERMINE KOUNT
MAIN PROGRAM (CONT'D)

A

KOUNT + REPLACE BITS

0 -

LOGPRT =EQ. 0

YES

NO

PRINT BIT REPLACEMENT DATA

XX(I) = XX(I) - STROKE

KOUNT +

0 -

CALCULATE XX(I)

CALCULATE ADRT

ICYLE =EQ. 0

NO

YES

CALL MUCK

CALCULATE TTM, TMH

967
MAIN PROGRAM (CONT'D)

CALL TRANS

DETERMINE CTIME, THIRD2, TMOLE2, TBIT2

W0L06 = WTLDG - TEMPWT

TEMPWT = 0.0

ILWTID = ILWTID + 1

IDLOAD = 0

CALL TRANS

TEMPWT = TEMPWT - WTLDG

FIND LOGO, JJ, AT NSW

FIND TLOCK, JJ, CTIME (JJ)

TEMPWT = WTMUCK

ZD

| TEMPWT - WTMUCK |

CALCULATE WTMUCK

SCCTM = 0.0

END
MAIN PROGRAM (CONT'D)

FIND SHFTMH, TUNNEL, TBIT1
TMOLE1, THIRD1, CTIME

CALL SUPPRRT

CALCULATE AVAMH

REQMH - AVAMH

FIND SWTTIM, SHFTMH, NSHIFT

CALCULATE TSUPPT, ADRT

ICYCLE = EQ. 0

CALL MUCK

YES

LOGPRT = EQ. 0

PRINT BORING
INFORMATION

NO

FIND ITEM

NEVENT = EQ. 1

CREATE IXR ARRAY

371
MAIN PROGRAM (CONT'D)

DISTRIBUTE MANPOWER

SHFTMH \( \geq \) XX(IR) 

YES

CALCULATE SHFTMH
NSHIFT = NSHIFT + 1

NO

SHFTMH \( \lt \) XX(IR)

FIND SHFTMH, WORK(I), ACTIM(I)
ACT(I), CTM(I)

PRINT REPAIR DATA

LOGPRF 0

+ -

FIND CTIME, TBIT3, TMOLE3
THIRD3, TBIT2, TMOLE2, THIRD2
XX(I) = XX(I) - XX(ITEM)
CALCULATE XX(ITEM)

ITEM = EQ MOLE

YES

INSPECT BITS
temstr = STROKE

NO

TEMSTR = TEMSTR - TFT

B
MAIN PROGRAM (CONT'D)

```
B

TNL-TUNNEL  -  PRINT: TERMINATED BY TNL
          +  520

MAXSHT-NSHIFT  -  PRINT: TERMINATED BY MAXSHT
          +  520

TIMAX-CTIME  -  PRINT: TERMINATED BY TIMAX
          +  520

999

PRINT SUMMARY
DATA

IDEOS = 1

CALL TRANS

STOP
```

SUBROUTINE MOTION

START

LOAD(NL) EQ. 0 NO

YES

REVERSE G(I), D(I), DS(I)

FIND SGL(I), MSS(I)

INITIALIZE TIME(NL), TSTOP(NL)

TPASS(NL), DSTOP(NL)

LOAD(NL) EQ. 0 NO

YES

REVERSE NS

FIND SLEFT, GLEFT, MM

CALCULATE FRFC, GFC, REQTF

SPEED(NL) + 111

0--

120

CALCULATE AVATF, ACCFC, ACCR

ACC R + 117

115

151

STOP

116

SPEED(NL) YES

DETERMINE T1, D1
SUBROUTINE MOTION (CONT'D)

ADEC = -ACCR

FIND KK

CALCULATE TD, DP

FIND NEW VALUE OF ACCR

ACCR = ACCMAX

DETERMINE D1, T1

STOP
SUBROUTINE MOTION (CONT'D)

117

ACCR = ACCMAX

+ 0 -

FIND D1, T1
FIND D2, T2

D1 - D2

+ 0 -

SPEED(NL) = S(NL, KK+1)

151

CALCULATE DSTOP(NL)
DISTR(NL), TIME(NL)
GLEFT, SLEFT

SLEFT - 0.5

0 +

GLEFT - 0.5

0 +

MM = MM + 1
CALCULATE GFC, REQTF

111
SUBROUTINE MOTION (CONT'D)

LOADING(NL) EQ. 0

RETURN

LOADING(NL) NO

REVERSE DS(I)

FIND NEW SPEED(NL) = SPEED(NU) = VELMAX

ACCR

DETERMINE SLEFT, OTRD

CALCULATE T1, DI

TPASS(NL), TSTOP(NL)

FIND NEW SPEED(NL)

CALCULATE T1, DI

TPASS(NL), TSTOP(NL)

< TSTOP(NL)

LOADING(NL) EQ. 0

RETURN

G(I), 0(I), NS

YES

G(I), 0(I), NS

NO

FIND NEW SPEED(NL) = SPEED(INL) = VELMAX

SLEFT = SLEFT

DSTOP(NL) = SLEFT

SLEFT = SLEFT

111

116

120

154

154

69
SUBROUTINE TRANS

START

ICYLE      YES
    EQ.0

IDEOS      NO
    EQ.1

ILDC=0, ILC=0
LC(I)=0

ILS
    EQ.0

LWTID
    EQ.0

NL=1

CALL LOADNG

IDLOAD=1
COUNT SAFT, SCCTM
CTLOC(NL), W TG,
WAIT TIMES

CALL MOTION

COUNT CTLOC(NL)
TLOC3(NL)
CHANGE LL(NL), LS(NL)

LOGPRT
PRINT MOTION DATA

READ INPUT DATA

INITIALIZE VARIABLES
COUNT NSW
LOCATE TRAINS

CALL LOADNG

TLOC3(NL)
CHANGE LL(NL), LS(NL)

LOGPRT
PRINT MOTION DATA

LIL=0
FIND SMALLEST
CTLOC(I)

LIL
    EQ.0

LIL=LIL-1
REMOVE
LLW(NL)

PRINT SUMMARY

IDEOS=1

RETURN

NL=1

CALL LOADNG

IDLOAD=1
COUNT SAFT, SCCTM
CTLOC(NL), W TG,
WAIT TIMES

CALL MOTION

COUNT CTLOC(NL)
TLOC3(NL)
CHANGE LL(NL), LS(NL)

LOGPRT
PRINT MOTION DATA

READ INPUT DATA

INITIALIZE VARIABLES
COUNT NSW
LOCATE TRAINS

CALL LOADNG

TLOC3(NL)
CHANGE LL(NL), LS(NL)

LOGPRT
PRINT MOTION DATA

LIL=0
FIND SMALLEST
CTLOC(I)

LIL
    EQ.0

LIL=LIL-1
REMOVE
LLW(NL)

PRINT SUMMARY

IDEOS=1

RETURN
SUBROUTINE TRANS (CONT'D)

READ INPUT DATA

INITIALIZE VARIABLES

CALCULATE PWT, SPWT

CALCULATE CAPMH TPM, MOTM

CALCULATE DIFTM, MOTM, CTIME, SHFTMH, NSHIFT

IF MOTM ≠ MOTM - SPWT

SPWT = SPWT - MOTM

FIND TBELT1

MOTM = MOTM - SPWT

FIND TBELT1

SPWT = SPWT - MOTM

FIND TBELT1

SPWT = 0.0

CALCULATE DOWNMH

FIND TBELT3

FIND NEW SPWT

PRINT SUMMARY OF CONTINUOUS HANDLING SYSTEM
SUBROUTINE TRANS (CONT'D)

LOAD(NL) EQ. 1
YES

NS-EQ. 1
YES

NS-EQ. NSW
YES

NS-EQ NSW
YES

LOAD(NL) EQ. 1
YES

NS-EQ. 1
YES

NS-EQ. NSW
YES

LOAD(NL) EQ. 1
YES

NS-EQ. 1
YES

NS-EQ NSW
YES

LOAD(NL) EQ. 1
YES

NS-EQ. 1
YES

NS-EQ NSW
YES

LOAD(NL) EQ. 1
YES

NS-EQ. 1
YES

NS-EQ NSW
YES

LOAD(NL) EQ. 1
YES

NS-EQ. 1
YES

NS-EQ NSW
YES

LOAD(NL) EQ. 1
YES

NS-EQ. 1
YES

NS-EQ NSW
YES

LOAD(NL) EQ. 1
YES

NS-EQ. 1
YES

NS-EQ NSW
YES

LOAD(NL) EQ. 1
YES

NS-EQ. 1
YES

NS-EQ NSW
YES

LOAD(NL) EQ. 1
YES

NS-EQ. 1
YES

NS-EQ NSW
YES

LOAD(NL) EQ. 1
YES

NS-EQ. 1
YES

NS-EQ NSW
YES

LOAD(NL) EQ. 1
YES

NS-EQ. 1
YES

NS-EQ NSW
YES

LOAD(NL) EQ. 1
YES

NS-EQ. 1
YES

NS-EQ NSW
YES
SUBROUTINE TRANS (CONT'D)

COUNT ILC
CTLOC(II)
TLOC4(II)

FIND
CTLOC(IL),
TLOC4(IL)

CALL DUMP

COUNT CTLOC(NL),
TLOC2(NL)
CHANGE NLDL,
NLDE, IDL(NL),
IDE(NL)

LS(NS) =
LS(NS) + 2

YES
LS(NS) =
LS(NS) - 1

LS(NS) =
LS(NS) + 2

LS(NS) =
LS(NS) + 1

COUNT LIL
REMOVE IL,
LW(IL)

FIND
CTLOC(IL),
TLOC4(IL)

YES

COUNT TL0C4(JJ)

COUNT LIL

NO

LS(NS) =
LS(NS) - 1

LS(NS) =
LS(NS) + 2

YES

LS(NS) =
LS(NS) + 1

NOTIMAX

PRINT: TERMINATED

YES

701

ideos = 1

G

NO

CTLOC(II) >= TIMAX

YES
SUBROUTINE TRANS (CONT'D)

156

LS(NS-1) - EQ. 1 OR 3
NO
CALL MOTION

CHANGE LL(NL), LS(NS)
FIND CTLOC(NL), TLOC3(NL)

PRINT MOTION DATA
LOGPRT
O
165

ILC = ILC + 1

FIND LOCQII AT LS(NS-1)

150

YES
ILC - EQ. 0

LC(I) - EQ. II
NO
YES

DETERMINE SSQC, CTLOC(NL), TLOC4(NL)

ILC = ILC + 1

DETERMINE ILC, CTLOC(JJ), TLOC4(JJ)

340

YES
ILC - EQ. 0
NO

340
SUBROUTINE TRANS (CONT'D)

CALL MOTION

COUNT CTLOC(NL)
TLOC3(NL)
CHANGE LL(NL),
LS(NS)

PRINT MOTION
DATA

FIND CTLOC(NL)
DISTR(NL), TLOC3(NL)
CHANGE NLDL, IDL(NL)

COUNT CTLOC(JJ)
TLOC4(JJ)

CHANGE IL TO LIL
COUNT CTLOC(JJ)
TLOC4(JJ)
SUBROUTINE TRANS (CONT'D)

```plaintext
IDLOAD -EQ. 0

ILWTID -EQ. 1 AND NL LT NLOCO
  YES

CTTM = CTIME + SCCTM
FIND ILC
  YES

ILC -EQ. 0
  YES

340

ILC .GE. NLOCO
  NO

FIND NL FOR SMALLEST CTLOC(I), I ≠ ILC

200

DETERMINE CTLOC(LIL), TLOC4(LIL)

300

IL -EQ. 0
  NO

NLDL -EQ. 1
  YES

CALL DUMP

300

FIND CTLOC(NL), TLOC2(NL)
CHANGE NLDL, IDL(NL), NLDE, IDE(NL), LS(NS)
```
SUBROUTINE TRANS (CONT'D)

DETERMINE WTIM

IF IL = 0

DETERMINE CTLOC(NL), DISTR(NL), TLOC3(NL)

IF SAFT ≥ DELTH

INCREMENT HAUL, TSEC, DMS

IF TSEC ≥ DD

CHANGE NSECS, D(NSECS)

IF DMS- DISW

CHANGE NSW, DS

CALL LOADING

FIND SCCTM, CTLOC(NL), TLOC1(NL)

SAFT, WTG

CHANGE LS(NS)

IF IDLOAD = 1

CALCULATE WWTM

IF IL = 1

DETERMINE CTLOC(NL) TLOC4(NL)

IF WWTM ≥ 0

FIND CTLOC(NL) TLOC4(NL)

IF IL = 0

DETERMINE CTLOC(NL) TLOC4(NL)

IF IL = 1

DETERMINE CTLOC(JJ) TLOC4(JJ)

IF IL = 0

DETERMINE CTLOC(JJ) TLOC4(JJ)

IF IL = 1

DETERMINE CTLOC(JJ) TLOC4(JJ)

IF IL = 0

DETERMINE CTLOC(JJ) TLOC4(JJ)

IF IL = 1

DETERMINE CTLOC(JJ) TLOC4(JJ)
SUBROUTINE TRANS (CONT'D)

129

FIND CTLOC(NL), DISTR(NL), TLOC3(NL)

IFDLC.NE.1

150

IL=IL+1

FIND SMALLEST CTLOC(I)

IF I.LT.IL

200

IDLC=0

CALCULATE SSCC

SSCC

Determine CTLOC(NL) TLOC4(NL)

+ OR -

Determine TLOC4(JJ) CTLOC(JJ)

CHANGE IL TO ILC

340
SUBROUTINE TRANS. (CONT'D)

125

- NLDE
  - EQ. 1

- IDE(NL)
  - EQ. 1

- ILC
  - EQ. 0

- LS(NS+1)
  - EQ. 0 OR 1

- ILC
  - EQ. 0

- CHANGE NLDE, IDE(I)

150

- CALL MOTION

340

- PRINT MOTION DATA

165

- FIND CTLOC(JJ), TLOC4(JJ)

300

- CHANGE IL TO ILC

- DETERMINE TLOC4(JJ), CTLOC(JJ)
The Computer Program

The computer model which is presented on the following pages consists of a main program and seven subroutines. The jobs performed by the individual subroutines are explained by the comment cards located at the beginning of each subroutine deck. The program was written for the Univac 1108 in standard FORTRAN IV and should be rather easy to transfer to other machines since few machine dependent statements were used. One aspect of the program which may need attention is the random number generator. The Univac 1108 used at the University of Utah uses the function RAND(N) to assign a random number uniformly distributed between zero and one to any variable. For example, the statement Y = RAND(N) will result in a random number between zero and one being assigned to the variable Y. Users wishing to use the program will have to check the random number function for their machines and, if necessary, replace all the statements calling random numbers with statements specific to their own machines.
DIMENSION IXKK(20), IXA(20)
DIMENSION MAN(120, 2), XX(120), IX(120), IXR(120), XIXR(20),
SMANAW(20), ACT(20), LOG(20), WORK(20), CTM(20), JON(20),
\* MUT(5), ACTM(20)
COMMON WTMUCK, WTM, NLOC0, TMUCK, KMAX, NCAHS, LCLAS, NUC, ACCMAX,
\* VELMAX, UCECL, NSN, S1, AFTH, NSEC, NSW, WTCAR, FCAK, TPS, ADKT,
\* CROSEC, GAMMA, CTIME, LWID, TTM, TNL, TIMAX, NRBG, ILS,
\* CFI(15), FTA(13), NCLAS(120), LL(10), CTLLOC(10), TLOC(10), TLOC(10),
\* FLOCO(10), G(100), D(100), OS(10), WTLOAD(10), SPEED(10), LOAD(10),
\* DISTR(10), TIML(15), TSTOP(10), IPASS(10), WTPRN(10), CFL(13), CFU(13),
\* CLT(13), NTL(13), DSTOP(10), ST(10), CFSL(13), TLOAD(10), TDUMP(10),
\* CF(10, 100, 13), TV(120, 13), T(10, 30), S(10, 30), LS(20, 30), ILWTD
\* , IDLOAD, SCCTM, TBIT2, TMOLE2, THIRD2, DELTH, IDEOS, TNL,
\* , FTAD, TMOLE1, TBIT1, THIRD1, NCREW, SHFTMH, HRPSH, NSHIFT, IMAN, ICYCLE

READ IN CONTROL CARDS

94 FORMAT ( 15)
READ 93 (CFI(I), FTA(I), I=1, NRBG)
93 FORMAT (2F10, 5)
READ 95 (RADIUS, GAMMA)
95 FORMAT (2F10, 3)
CROSEC = 3.14159*RADIUS*RADIUS
READ 96, MBITS, NACF, NCREW, ICYCLE
96 FORMAT (4IU)
READ 97, TIMAX, TNL, MAXSHT
97 FORMAT (2F15, 4, I10)
READ 98, LOGPRT, NH
98 FORMAT (2I5)
READ 99, CTIME, TUNNEL, HRPSH, TOLIT
99 FORMAT (4F10, 3)
READ 100, BINSMH, STROKE, INSPM
100 FORMAT (2F10, 3, 15)
NSCF=NBITS*2 + NACF
NCF=NBITS*2 + NACF*2-1

INITIALIZE THE VARIABLES

DO 101 I=1, NCF
DO 102 JJ=1, 2
102 MAN(I, JJ)=0
DO 103 J=1, 13
CF(I, J)=0.0
103 TV(I, J)=0.0
MREST=0
RLSTMH=0.0
WAITH=0.0
NSHIFT=0
```fortran
TB1T1=0.0
TB1T2=0.0
TB1T3=0.0
TB1T4=0.0
TB1T5=0.0
TB1LE1=0.0
TB1LE2=0.0
TB1LE3=0.0
TB1RD1=0.0
TB1RD2=0.0
TB1RD3=0.0
ID=10
ILS=0
ILNTH=0
IDEOS=0
SWTTIM=0.0
TSUPPT=0.0

READ IN CUMULATIVE PROBABILITY FUNCTIONS
   DO 103 I=1,NCF
      READ 104+ NCLAS(I)
   104 FORMAT ( I10)
      N0=NCLAS(I)
   DO 110 J=1,NJ
      READ 105, CF(I,J), TV(I,J)
   105 FORMAT ( 2F10.5)
   103 CONTINUE

READ IN THE UPPER AND LOWER LIMITS OF NUMBER OF MEN
   REQUIRED FOR EACH ACTIVITY
   MAN(I,1)=LOWER LIMIT
   MAN(I,2)=UPPER LIMIT

READ 106, MNBITL,MNBITU
   NI=MNBIT
   DO 107 I=1,N1
      MAN(I,1)=MNBITL
   107 MAN(I,2)=MNBITU
   106 FORMAT ( 2I10 )
   N2=N1+2+1
   N3=NSCF-1
   READ 109 ((MAN(I,J), J=1,2), I=N2,N3)
   109 FORMAT ( 1215)

CALL SUBROUTINE CALCUM TO OBTAIN ABSCISSA VALUES CORRESPONDING
   TO CF(I) = RAND
```
DO 150 I=1,NCF
   ICF=I
   CALL CALCUM(ICF,X)
150 XX(I)=X

COMPUTE MANHOURS AND NUMBER OF MEN AVAILABLE FOR THE SHIFT
   NSHIFT=NSHIFT+1
   SHIFTMH=HRPSH*NCREW
   MAN=NCREW

IF WANTED, PRINT LOG OF OPERATIONS

IF (LOGPRT) 995,160,995
160 PRINT 161
161 FORMAT (1H1,'*****LOG OF OPERATIONS*****')
162 FORMAT (1H0,' C]LK TIME','5X,'COMPLETED EVENT','3X','MAN HOUR'
     '3X,'TUNNEL LENGTH')

SEARCH FOR THE SHORTEST LIFE OF THE UNIT IN THE SYSTEM

995 IF (ILS .NE. 0) GO TO 89
   CALL TRANS
89 CONTINUE
   ILS=ILS+1
   ALRT=XX(NSCF)
   XX(NSCF)=XX(NSCF)/60.0
   IF (ICYCLE .EQ. 0) GO TO 829
   CALL MUCK

   TEMPWT=TMUCK
629 TEMSTR=STROKE
999 IN=NSCF-1
   IL1=NBIT5*2+2
   M=NBIT5*2+1
   KK=M
   DO 170 L=1,IL1,IN
170 IF (XX(I) .LT. XX(M)) M=I
   ITEM=M
   COUNT NUMBER OF EVENTS HAVING THE SAME LIFE

   ICOUNT=0
   LLL=ITEM-1
   MM=ITEM+1
   SX=XX(ITEM)
IF ( ITM .EQ. KK ) GO TO 314
DO 311 I=KK,LLL
XI=XX(I)
DIF=ABS(SX-XI)
IF ( DIF .GT. TOLIT ) GO TO 311
ICOUNT=ICOUNT+1
II=ICOUNT+1
IX(II)=I
311 CONTINUE
IF ( ITM .EQ. IN ) GO TO 315
DO 314 J=MM,IN
XI=XX(J)
DIF=ABS(SX-XI)
IF ( DIF .GT. TOLIT ) GO TO 314
ICOUNT=ICOUNT+1
II=ICOUNT+1
IX(II)=J
312 CONTINUE
315 CONTINUE
NEVENT=ICOUNT+1
IX(1)=ITEM
C
C COMPUTE THE TIME FOR ADVANCING AND THE DISTANCE TO BE ADVANCED
C
IF ( MH ) 355,356,355
356 TTM=XX(ITEM)
TFT=XX(NSCF)*XX(ITEM)
GO TO 357
355 TFT=XX(ITEM)
TMT=XX(ITEM)/XX(NSCF)
357 TMM=TTM*NCREW/60.0
C
C COMPARE TFT WITH STROKE
C
487 IF ( TFT-TEMSTR ) 350,352,352
C
ADVANCE BY TEMSTR ( CASE OF TEMSTR .LT.TFT )
C
352 TTM=TEMSTR/XX(NSCF)
TMM=TMM*NCREW/60.0
WLDG=CLOSEG*TEMSTR *GAMMA/2000.0
TFTA=TEMSTR
C
CALL SUPPRT(TFTA,REQMH)
C
SCCTM=0.0
IF (ICYCLE .NE. 0) GO TO 760
FTAD=TEMSTR
C
CALL TRANS
GO TO 413

760 IF ( LS(NSW) .EQ. 0 .OR. LS(NSW) .EQ. 1) GO TO 751
    LWTID=1
    GO TO 752

751 LW TID=0

CALL TRANS

CTIME=CTIME+WTIM
TMOLE2=TMOLE2+WTIM
T2IT2=T2IT2+WTIM
THIRD2=THIRD2+WTIM

752 IF ( WTL DG-TEMPWT) 753, 754, 755

753 TEMPWT=TEMPWT-WTL DG
    DO 756 I=1, NLOCO

756 IF ( LL(I) .EQ. NSW) GO TO 757
    PINT 758

758 FORMAT ( 1HO, 'LOADING WAS ATTEMPTED WITHOUT EMPTY TRAIN AT LOADING POINT')
    STOP

757 JJ=I
    C TLOC(JJ)=C TLOC(JJ)+TMUCK
    TLOC1(JJ)=TLOC1(JJ)+TMUCK
    GO TO 413

754 TEMPWT=W TMUCK
    ILWT ID=ILWT ID+1
    ILOAD=I

CALL TRANS

IF ( IDEOS .EQ. 1) GO TO 520

CALL MUCK

TEMPWT=W TMUCK
GO TO 413

755 WTL DG=W TL DG-TEMPWT
    TEMPWT=W TMUCK
    ILWT ID=ILWT ID+1
    ILOAD=I

CALL TRANS

IF ( IDEOS .EQ. 1) GO TO 520

CALL MUCK

TEMPWT=W TMUCK
GO TO 760
CHECK TO SEE IF THE ADVANCE CAN BE COMPLETED IN THE SHIFT

413 IF (SHFTMH-TMH) 411,412,412
411 SHFTMH=SHFTMH+NCREW*HRPSH
NSHIFT=NSHIFT+1
GO TO 413
412 SHFTMH=SHFTMH-TMH
TUNNEL=TUNNEL+TEMSTR
TBIT1=TBIT1+TTM
TMOLE1=TMOLE1+TTM
THIRD1=THIRD1+TTM
CTIME=CTIME+TTM
AVAMH=TTM*2.0/60.0
IF (REQMH-AVAMH) 835,835,836
836 WTTIM=((REQMH-AVAMH)/2.0)*60.0
SWTTIM=SWTTIM+WTTIM
CTIME=CTIME+WTTIM
TBIT2=TBIT2+WTTIM
TMOLE2=TMOLE2+WTTIM
THIRD2=THIRD2+WTTIM
MREST=MREST+(NCREW-2)
RESTMH=RESTMH+WTTIM*(NCREW-2)/60.0
WAITM=WAITM+WTTIM
WTTMH=WTTIM*NCREW/60.0
890 IF (SHFTMH=WTTMH) 830,832,832
830 SHFTMH=SHFTMH+NCREW*HRPSH
NSHIFT=NSHIFT+1
GO TO 890
832 SHFTMH=SHFTMH-WTTMH
TSUPPT=TSUPPT+TTM+WTTIM
GO TO 891
835 TSUPPT=TSUPPT+TTM
891 CONTINUE
TFT=TFT-TEMSTR
TEMSTR=STROKE
IF (LOGPRT) 481,480,481
480 PRINT 162
PRINT 362,CTIME,TMH,TUNNEL

INSPECTION

481 IF (IMAN-INSPM) 450,451,451
450 MREST=IMAN+MREST
RESTMH=SHFTMH+RESTMH
IF (IMAN.EQ.0) GO TO 892
WAITM=WAITM+(SHFTMH/IMAN)*60.0
CTIME=CTIME+(SHFTMH/IMAN)*60.0
892 NSHIFT=NSHIFT+1
SHFTMH=HRPSH*NCREW
IMAN=NCREW
451 IF (SHFTMH=BINSMH) 452,453,453
452 SHFTMH=SHFTMH+NCREW*HRPSH
NSHIFT=NSHIFT+1

SHIFTMH=SHFTMH+INSMH
TBINS=(INSMH/INSPM)*60.0
TBIT5=TBIT5+TINS
CTIME=CTIME+TBINS
TMOLE2=TMOLE2+TINS
THIRD2=THIRD2+TINS
MREST=(NCREW-INSPM)+MREST
RESTMH=RESTMH+(NCREW-INSPM)*TINS/60.0

PRINT 162
PRINT 464,CTIME, BINSMH, TUNNEL
FORMAT (1H , F10.3, 5X, 'BIT INSPECTION ', 3X,F10.3,3X, F10.3)

COUNT NUMBER OF BITS TO BE REPLACED, IF ANY

IF ( ID .EQ. 1) GO TO 668

GO TO 609

DO 415 I=1,NBITS

XX(I)=XX(I)*XX(NSCF)

IF (XX(I).GT. TOLSTK) GO TO 415
KOUNT=KOUNT+1
K=KOUNT
IBIT(K)=I

CONTINUE

REPLACE BITS, IF ANY

IF ( KOUNT .EQ. 0 ) GO TO 430

DO 420 I=1,KOUNT
ITEM=IBIT(I)

IF ( IMAN-MAN(ITEM,1)) 421,422,423

IF ( IMAN-MAN(ITEM,2)) 422,422,424

IMAN=IMAN+MANWK

MREST=IMAN+MREST

RESTMH=RESTMH+SHFTMH

IF ( IMAN .EQ. 0) GO TO 893

WAITM=WAITM+(SHFTMH/IMAN)*60.0

CTIME=CTIME+(SHFTMH/IMAN)*60.0

NSHIFT=NSHIFT+1

SHFTMH=HRPSH*NCREW

IMAN=NCREW

MANWK=MAN(ITEM,2)
IMAN=IMAN-MANWK
GO TO 425
422 MANWK=IMAN
IMAN=0
425 ITEMH=ITEM+NBITS
BMH=XX(ITEMH)
429 IF (SHFTMH .LT. BMH) GO TO 428
ACTM=(BMH/MANWK)*60.0
TUIT4=TBIT4+ACTM
CTIME=CTIME+ACTM
TMOLE2=TMOLE2+ACTM
THIRD2=THIRD2+ACTM
SHFTMH=SHFTMH-BMH
IF (LOGPRT) 420,485,420
485 PRINT 162
PRINT 486,CTIME,BMH,TUNNEL,ITEM
486 FORMAT (1H,F10.3,5X,'BIT REPLACING ',3X,F10.3,3X,F10.3,
4X,3X,'BIT NO.',I3,2X,'REPLACED')
422 CONTINUE

SUBTRACT STROKE FROM BIT LIFE
430 CONTINUE
IF (ID .NE. 1) GO TO 670
AA=(STROKE-TEMSTR)+TFT
GO TO 671
670 AA=STROKE
671 DO 431 I=1,NBITS
IF (MH .NE. 0) GO TO 432
AA=AA/XX(NSCF)
GO TO 431
432 AA=AA
431 XX(I)=XX(I)-AA

REPLACE BIT LIFE OF BIT REMOVED WITH A NEW LIFE
435 IF (KOUNT .EQ. 0) GO TO 438
DO 435 I=1,KOUNT
ITEM=IRIT(I)
ITEMH=ITEM+NBITS
CALL CALCUM(ITEMH,X)
XX(ITEMH)=X
CALL CALCUM(ITEM,X)
435 XX(ITEM)=X
438 IF (ID .NE. 1) GO TO 577
ID=ID+1
TEMSTR=STROKE
GO TO 998
577 ICF=NSCF
CALL CALCUM (ICF,X)

XX(NSCF)=X
AURT=XX(NSCF)
XX(NSCF)=XX(NSCF)/60.0
IF (ICYCLE .EQ. 0) GO TO 838

CALL MUCK

838 TTM=TFT/XX(NSCF)
TMH=TTM*NCREW/60.0
IF (TUNL=TUNNEL) 950,950,951
951 IF (MAXSHT=NSHFT) 952,952,953
953 IF (TIMAX=CTIME) 954,954,987
950 PRINT 507
GO TO 520
952 PRINT 512
GO TO 520
954 PRINT 513
GO TO 520

COMPLETED INSPECTION AND REPLACING OF BITS
ADVANCE BY TFT (CASE OF TEM STR .GE. TFT)

350 CONTINUE
TFTA=TFT

IF (TFT) 771,361,771
771 WTLDG=CROSEC*TFT*GAMMA/2000.0
SCCTM=0.0
IF (ICYCLE .NE. 0) GO TO 710
FTAD=TFT

CALL TRANS

GO TO 369
710 IF (LS(NSW) .EQ. 0 .OR. LS(NSW) .EQ. 1) GO TO 701
LWTID=1
GO TO 702
701 LWTID=0

CALL TRANS

CTIME=CTIME+WTIM
TMOLE2=TMOLE2+WTIM
TBIT2=TBIT2+WTIM
THIRD2=THIRD2+WTIM
702 IF (WTLDG=TEMPWT) 703,704,705
703 TEMPWT=TEMPWT-WTLDG
DO 706 I=1,NLOCO
706 IF (LL(I) .EQ. NSW) GO TO 707
PRINT 708
708 FORMAT ( 1HO,'LOADING WAS ATTEMPTED WITHOUT EMPTY TRAIN AT LOADING POINT')
STOP
707 JJ=I
   CTLOC(JJ)=CTLOC(JJ)+TMUCK
   TLOC1(JJ)=TLOC1(JJ)+TMUCK
   GO TO 369
704 TEMPWT=0.0
   ILWTID=ILWTID+1
   IDLOAD=0
C
   CALL TRANS
C
   IF ( IDEOS .EQ. 1) GO TO 520
C
   CALL MUCK
C
   TEMPWT=WTMUCK
   GO TO 369
705 WTLDG=WTLDG-TEMPWT
   TEMPWT=0.0
   ILWTID=ILWTID+1
   IDLOAD=0
C
   CALL TRANS
C
   IF ( IDEOS .EQ. 1) GO TO 520
C
   CALL MUCK
C
   TEMPWT=WTMUCK
   GO TO 710
C
   CALL SUPPRT(TFTA,REQMH)
C
   AVAMH=TTM*2.0 /60.0
   IF ( REQMH-AVAMH) 865,865,866
   WTTIME=((REQMH-AVAMH)/2.0)*60.0
CTIME = CTIME + WTTIM
SWTTIM = SWTTIM + WTTIM
TBIT2 = TBIT2 + WTTIM
TMOLE2 = TMOLE2 + WTTIM
THIRD2 = THIRD2 + WTTIM
MREST = MREST + (NCREW - 2)
RESTMH = RESTMH + WTTIM * (NCREW - 2) / 60.0
WTTMH = WTTIM * NCREW / 60.0

894 IF (SHFTMH = WTTMH) 860, 862, 862
860 SHFTMH = SHFTMH + NCREW * HRPSH
NSHIFT = NSHIFT + 1
GO TO 894
862 SHFTMH = SHFTMH - WTTMH
TSUPPT = TSUPPT + TTM + WTTIM
GO TO 895
865 TSUPPT = TSUPPT + TTM
895 CONTINUE

CALL CALCUM (NSCF, X)

XX(NSCF) = X
ADRT = XX(NSCF)
XX(NSCF) = XX(NSCF) / 60.0
IF (ICYCLE .EQ. 0) 60 TO 837

CALL MUCK

361 DO 370 K = 1, NEVENT
M = NBITSP + 2 + 1
N = NSCF - 1
JCOUNT = 0
ITEM = IX(K)
DO 365 J = M, N
JCOUNT = JCOUNT + 1
365 IF (ITEM .EQ. J) GO TO 366
PRINT 367, ITEM
366 IR = NSCF + JCOUNT
370 IXR(K) = IR

START REPAIRS
MAKE AN ARRAY OF IXR(I) IN DECREASING ORDER
IF ( NEVENT .EQ. 1) GO TO 371
DO 372 I=1,NEVENT
IR=IXR(I)
372 XIXR(I)=XX(IR)
NA=NEVENT-1
DO 373 J=1,NA
M=J
MA=J+1
DO 374 K=MA,NEVENT
IF (IXR(K) .GT. XIXR(M)) M=K
TEMP=IXR(J)
XIXR(J)=XIXR(M)
373 XIXR(M)=TEMP
MKL=IR(1)
DO 840 KL=2,NEVENT
IR=IR(KL)
IF (XX(MKL)-XX(IR)) 840,371,840
840 CONTINUE
DO 375 I=1,NEVENT
IR=IR(I)
DO 376 J=1,NEVENT
IF (XIXR(J)-XX(IR)) 376,377,376
376 CONTINUE
377 IXXR(J)=IR
375 IXX(J)=IR+(NSCF-NBITS*2)
DO 645 I=1,NEVENT
IX(I)=IXX(I)
645 IXXR(I)=IXR(I)

C

371 ISUM1=0
ISUM2=0

C To distribute men to crews, the bigger jobs are assigned manpower first

C

DO 380 I=1,NEVENT
ITEM=IX(I)
ISUM1=ISUM1+MAN(ITEM,1)
380 ISUM2=ISUM2+MAN(ITEM,2)
DO 500 KK=1,NEVENT
IR=IR(KK)
ITEM=IX(KK)
I=KK
IF (IMAN-ISUM1) 381,382,383
383 IF ( IMAN-ISUM2) 381,384,384
382 MANAW(I)=MAN(ITEM,1)
GO TO 395
384 MANAW(I)=MAN(ITEM,2)
GO TO 395
381 IF (IMAN-MAN(ITEM,1)) 390,391,392
392 IF (IMAN-MAN(ITEM,2)) 391,391,393
390 MREST=MREST+IMAN
RESTMH = RESTMH + SHFTMH
IF (IMAN .EQ. 0) GO TO 846
WAITM = WAITM + (SHFTMH / IMAN) * 60.0
CTIME = CTIME + (SHFTMH / IMAN) * 60.0
846 NSHIFT = NSHIFT + 1
SHFTMH = NCREW * NRPSPH
IMAN = NCREW
393 MANAW(I) = MAN(ITEM + 2)
GO TO 395
391 MANAW(I) = IMAN
GO TO 395
395 IMAN = IMAN - MANAW(I)

C C IF A JOB CANNOT BE HANDLED IN SHIFT, IT IS EXTENDED INTO
C THE NEXT SHIFT
C
C IF (SHFTMH .GE. XX(IR)) GO TO 396
397 SHFTMH = SHFTMH + NCREW * NRPSPH
NSHIFT = NSHIFT + 1
IMAN = NCREW - MANAW(I)
IF (SHFTMH .LT. XX(IR)) GO TO 397
396 SHFTMH = SHFTMH - XX(IR)
WORK(I) = XX(IR)
500 ACTIM(I) = (WORK(I) / MANAW(I)) * 60.0

C C ARRANGE ACTIM(I) IN ASCENDING ORDER
C
C IF (NEVENT .EQ. 1) GO TO 550
DO 557 I = 1, NEVENT
557 ACT(I) = ACTIM(I)
NA = NEVENT - 1
DO 551 J = 1, NA
M = J
MA = J + 1
DO 552 I = MA, NEVENT
552 IF (ACT(I) .LT. ACT(M)) M = I
TEMP = ACT(J)
ACT(J) = ACT(M)
551 ACT(M) = TEMP
DO 558 K = 1, NEVENT
DO 559 I = 1, NEVENT
IF (ACTIM(I) - ACT(K)) 559, 558, 559
559 CONTINUE
558 LOG(K) = I
GO TO 553
550 ACT(I) = ACTIM(I)
LOG(I) = 1
553 IF (NEVENT .EQ. 1) GO TO 630

C C REDUCE THE VALUES OF ACTIM(I) BY REDISTRIBUTING THE MANPOWER
C AFTER ACTIM(I) IS ACHIEVED
J=1
DO 600 L=NEVENT,2,-1
I=L

620 M=LOG(I)
IR=IXR(M)
ITEM=IR-(NSCF-NBITS*2)
IF (MAN(Item,2) .EQ. MANAW(M)) GO TO 471
MAD=MAN(Item,2)-MANAW(M)

610 IF (MAD-MANAW(J)) 601,602,602

602 MTB=MANAW(J)
IF (J .NE. 1) GO TO 603
DN=WORK(M)-ACT(J)*MANAW(M)/60.0
DD=MANAW(M)+MTB
GO TO 604

603 DN=DN-(ACT(J)-ACT(J-1))*DD/60.0
DD=DD+MTB

604 ACT(I)=ACT(J)+(DN/DD)*60.0
J=J+1
IF (I-J) 605,606,607

605 PRINT 608
608 FORMAT (1X,'RUN STOPPED BY AN ERROR, SEE STATEMENT 605 IN MAIN')
STOP

607 MAD=MAD-MTB
IF (I .NE. NEVENT) GO TO 609
IF (ACT(I)-ACT(I-1)) 600,610,610

609 IL=I+1
DO 611 K=IL,NEVENT

611 IF (ACT(K),GT,ACT(I)) GO TO 612
IF (ACT(I)-ACT(I-1)) 600,610,610

612 I=K
GO TO 620

610 IF (ACT(I)-ACT(I-1)) 613,471,471

613 PRINT 614
614 FORMAT (1X,'RUN STOPPED BY AN ERROR, SEE STATEMENT 606 IN MAIN')
STOP

C THE CASE OF MAD .GE. MANAW(J) IS COMPLETED
C THE CASE OF MAD .LT. MANAW(J) IS BEGUN
C

601 MTB=MAD
MANAW(J)=MANAW(J)-MTB
IF (J .NE. 1) GO TO 621
DN=WORK(M)-ACT(J)*MANAW(M)/60.0
DD=MANAW(M)+MTB
GO TO 622

621 DN=DN-(ACT(J)-ACT(J-1))*DD/60.0
DD=DD+MTB

622 ACT(I)=ACT(J)+(DN/DD)*60.0
IF (I .NE. NEVENT) GO TO 623
IF (ACT(I)-ACT(I-1)) 600,471,471

623 IL=I+1
DO 624 K=IL,NEVENT

624 IF (ACT(K),GT,ACT(I)) GO TO 625
IF ( I-(J+1)) 626,627,628

626 PRINT 629
629 FORMAT (1X,'RUN STOPPED BY AN ERROR. SEE STATEMENT 626 IN MAIN.')
   STOP
627 GO TO 471
628 IF (ACT(I)-ACT(I-1)) 600,471,471
625 I=K
   GO TO 620
600 CONTINUE

COMPLETE THE REDUCTION OF THE ACTIM(I) VALUES
REARRANGE ACT(I) IN ASCENDING ORDER

471 CONTINUE
   DO 631 L=1,NEVENT
51   IM=1
   DO 632 K=1,NEVENT
52      DO 633 IK=1,NEVENT
53         MM=JON(IK)
   633 IF (IM .EQ. MM) IM=IM+1
      M=IM
   DO 634 I=1,NEVENT
      DO 635 N=1,NEVENT
90         JJ=JON(N)
   635 IF ( JJ .EQ. I) GO TO 634
   636 CONTINUE
   CTM(K)=ACT(M)
   JON(K)=M
   CONTINUE
   GO TO 472
   630 CONTINUE
   CTM(1)=ACT(1)
   JON(1)=LOG(1)
472 IF ( LOGPRT) 473,470,473
470 PRINT 162
473 DO 455 K=1,NEVENT
   IK=JON(K)
   M=LOG(IK)
   IR=IXK(M)
   J=IR-NSCF
   TCT=CTIME+CTM(K)
   RJOB=WORK(M)
   GO TO (456,457,858),J
   456 TBIT3=TBIT3+CTM(K)
      IF ( LOGPRT) 455,275,455
275 PRINT 459, TCI, RJOB, TUNNEL
459 FORMAT (1H,' BIT REPAIR ',3X, F10.3,3X,F10.3)
   GO TO 455
457 TMOLE3=TMOLE3+CTM(K)
IF ( LOGPRT) 455,276,455
276 PRINT 400, TCT, RJOB, TUNNEL
400 FORMAT (1H, F10.3, 5X, 'MOLE REPAIR ', 3X, F10.3, 3X, F10.3)
GO TO 455
858 THIRD3=THIRD3+CTM(K)
IF ( LOGPRT) 455,277,455
277 PRINT 461, TCT, RJOB, TUNNEL
461 FORMAT (1H, F10.3, 5X, ‘THIRD REPAIR ’, 3X, F10.3, 3X, F10.3)
455 CONTINUE

C DETERMINE CLOCK TIME AND WAITING TIME
C
CTIME=CTIME+CTM(NEVENT)
IF (NEVENT .GE. 1) GO TO 404
NA=NEVENT-1
BIG=CTM(NEVENT)
DO 400 I=1,NA
WTM=BIG-CTM(I)
IK=JON(I)
M=LOG(IK)
MREST=MREST+MANAW(M)
RESTMH=RESTMH+(WTM*MANAW(M))/60.0
IR=IXR(M)
J=IR-NSCF
GO TO (401,402,403), J
401 TMOLE2=TMOLE2+WTM
THIRD2=THIRD2+WTM
GO TO 400
402 TBIT2=TBIT2+WTM
THIRD2=THIRD2+WTM
GO TO 400
403 TBIT2=TBIT2+WTM
TMOLE2=TMOLE2+WTM
400 CONTINUE
GO TO 405
404 MREST=MREST+(NCREW-MANAW(1))
RESTMH=RESTMH+(NCREW-MANAW(1))*CTM(1))/60.0
IR=IXR(1)
J=IR-NSCF
GO TO (406,407,408), J
406 TMOLE2=TMOLE2+CTM(1)
THIRD2=THIRD2+CTM(1)
GO TO 405
407 TBIT2=TBIT2+CTM(1)
THIRD2=THIRD2+CTM(1)
GO TO 405
408 TBIT2=TBIT2+CTM(1)
TMOLE2=TMOLE2+CTM(1)
405 CONTINUE
GO TO 405

C SUBTRACT XX(ITEM) FROM XX(I)
C END OF REPAIRS
ITEM = IX(1)
M = NBITS * 2 + 1
N = NSCF - 1
DO 490 I = M, N
490 XX(I) = XX(I) - XX(ITEM)
C
C REPLACE XX(ITEM) WITH NEW XX(ITEM)
C
DO 495 J = 1, NEVENT
ITEM = IX(J)
C
CALL CALCUM(ITEM, X)
C
XX(ITEM) = X
IR = IXR(J)
C
CALL CALCUM(IR, X)
C
495 XX(IR) = X
C
PERFORM A BIT INSPECTION IF THE MOLE IS DOWN FOR OTHER REPAIRS
C
DO 555 I = 1, NEVENT
IR = IXR(I)
IDR = IR - NSCF
555 IF (IDR .EQ. 2) GO TO 666
GO TO 607
666 I = 1
IF (ID .EQ. 1) GO TO 481
667 TEMSTR = TEMSTR - TFT
998 CONTINUE
C
C COMPLETED THE CASE OF TEMSTR GT. TFT
C CHECK TERMINATION VARIABLES
C
IF (TNL - TUNNEL) 505, 505, 506
506 IF (MAXSHIFT - NSHIFT) 510, 510, 515
515 IF (TIME - CTIME) 511, 511, 999
505 PRINT 507
507 FORMAT (1H1, 'SIMULATION WAS TERMINATED BY THE MAXIMUM ADVANCE OF THE TUNNEL!//')
GO TO 520
510 PRINT 512
512 FORMAT (1H1, 'SIMULATION WAS TERMINATED BY THE MAXIMUM NUMBER OF SHIFTS!//')
GO TO 520
511 PRINT 513
513 FORMAT (1H1, 'SIMULATION WAS TERMINATED BY THE MAXIMUM CLOCK TIME $!//')
520 PRINT 521
521 FORMAT (5X, '**********SIMULATION SUMMARY DATA**********!//')
PRINT THE SUMMARY OF THE SIMULATION

CTIME=CTIME/60.0
WAITM=WAITM/60.0
TBIT5=TBIT5/60.0
TBIT1=TBIT1/60.0
TBIT2=TBIT2/60.0
TBIT3=TBIT3/60.0
TBIT4=TBIT4/60.0
TMOLE1=TMOLE1/60.0
TMOLE2=TMOLE2/60.0
TMOLE3=TMOLE3/60.0
THIRD1=THIRD1/60.0
THIRD2=THIRD2/60.0
THIRD3=THIRD3/60.0
TSUPPT=TSUPPT/60.0
SWTITM=SWTITM/60.0

PRINT 525
525 FORMAT (1X,'CLOCK TIME',3X,'NO. SHIFT',3X,'TUNNEL LENGTH',3X,
& '* MH ON REST',3X,'IDLE TIME',2X,'BIT INSPI HR' //)
PRINT 526
526 FORMAT (CTIME,NSHIFT,TUNNEL,RESTM+WAITM,TBIT5
PRINT 526
PRINT 527
527 FORMAT (13X,'HR WORKED',3X,'HR WAITED',3X,'DOWN TIME',3X,
& '* REPLACING HR' //)
PRINT 528
528 FORMAT (TBIT1,TBIT2,TBIT3,TBIT4
PRINT 528
528 FORMAT (10H BIT , 3F12.3, 3X, F12.3)
PRINT 530
530 FORMAT (TMOLE1,TMOLE2,TMOLE3
PRINT 531
531 FORMAT (10H MOLE , 3F12.3)
PRINT 531
531 FORMAT (THIRD1,THIRD2,THIRD3
PRINT 531
531 FORMAT (10H THIRD , 3F12.3)
PRINT 531
531 FORMAT (TSUPPT,SWTITM
PRINT 531
531 FORMAT (TOTAL SUPPORTING TIME=', F12.3,' ', TOTAL TIME DELAY
& BY THE SLOWER SUPPORTING SYSTEM=', F12.3)
IDEOS=1

CALL TRANS

STOP
END

COMPIILATION: NO DIAGNOSTICS.
SUBROUTINE CALCUM(ICF,X)

COMMON WTMUCK,WTM,NCARS,LCLAS,ACCMAK,
*VELMAX,DECEL,NSC,D1,HAUL,NS,WTCA,FCAR,TPM,ADRT,
*CROSEC,GAMMA,CTIME,LWTID,TTM,NL,MAX,NRBG,ILS,
*CFR(13),FTA(13),NCLAS(120),LL(10),TLOC1(10),TLOC1(10),WLOAD(10),SPEED(10),LOAD(10),
*FLCO(10),G(100),D(100),DS(10),WLOAD(10),SPEED(10),LOAD(10),
*DISTR(10),TML(10),TSTOP(10),IPASS(10),WTTRN(10),CFL(13),CFD(13),
*CF(13),WTL(10),STOP(10),ST(13),CF(13),TLOAD(10),TDUMP(10),
*CF(120,13),TV(12C,13),T(10,30),S(10,30),LS(20),ILWTI
*ULOAD,SCCTM,TRIT2,TH3LE2,TH3RD2,DELTH,IDEOS,TLT
*FTAD,TMOL,TRIT,T3RD1,NCREW,SHFTMH,HRP,NSHIFT,IMAN,ICYCLE

SUBROUTINE CALCUM DETERMINES AN ABSCISSA VALUE ON A CUMULATIVE
PROBABILITY CURVE CORRESPONDING TO A GIVEN RANDOM NUMBER
BETWEEN 0.0 AND 1.0

I=ICF
Y=RAND(N)
JN=NCLAS(I)
DO 100 J=1,JN
IF (CF(I,J) .LE. Y .AND. CF(I,J+1) .GE. Y) GO TO 102
100 CONTINUE
PRINT 1'J3,I,Y,JN
103 FORMAT (1X,'I Y JN AT 103 IN CALCUM',I5,F10.8,I10)
PRINT 111
111 FORMAT (1X,'RUN TERMINATED BY ERROR IN SEARCHING THROUGH THE CF(I,
*J))
IF ( JN .GE. 100 ) GO TO 113
PRINT 112 ( CF(I,J), J=1,JN)
112 FORMAT (3F18.8)
113 STOP
102 X=TV(I,J)+(TV(I,J+1)-TV(I,J))*(Y-CF(I,J))/(CF(I,J+1)-CF(I,J))
RETURN
END
SUBROUTINE SUPPRT ( TFTA, REQMH )

COMMON WTMUCK, WTIM, NLUCO, TMUCK, KMAX, NCARS, LCLAS, NDC, ACCMAX,
$VELMAX, JECEL, NSC, D1, AFT, HAUL, SECS, NSW, WTCAR, FCAK, TPM, ADRT,
$CHOSLC, GAMMA, CTIME, LWIID, TTM, TNL, TIMAX, NNRG, ILS,
$CFR(13), FTA(13), CLAS(120), LL(10), TLOC(10), TLOC1(10), TLOC(10),
$FLOCO(10), G(100), D(100), DS(10), TLOAD(10), SPEED(10), LOAD(10),
$DISTR(10), TIML(10), TSTOP(10), TPASS(10), WTTRN(10), CFL(13), CFD(13),
$CT(13), TL(13), DSTOP(10), ST(13), CS(13), TLOAD(10), TLOAD(10),
$CF(120, 13), TV(120, 13), T(10, 30), S(10, 30), LS(20), ILWTID
$IDLOAD, SCCTM, TBIT1, TMOLE2, THI3RD2, SELTH, IDEOS, TNL
$FTAD, TMOLE1, TBIT1, THI3RD1, NCREW, SHFTMH, HRPSH, NSHIFT, IMAN, ICYCLE

SUBROUTINE SUPPRT DETERMINES THE MANHOURS OF TIME REQUIRED TO
COMPLETE THE SUPPORT WORK FOR ONE UNIT OF ADVANCE OF THE TUNNEL

SPMH=0.0
REQMH=0.0
NFTR=INT(TFTA+0.4)
IF ( NFTR .EQ. 0) GO TO 100
DO 10 I=1, NFTR
Y=RANU(N)
DO 20 J=1, NRUG
20 IF ( CFR(J) .LE. Y .AND. CFR(J+1) .GE. Y ) GO TO 30
PRINT 21
21 FORMAT (1X, 'CFR(I) - FTA(I) DIAGRAM HAS INCORRECT INPUT DATA')
STOP
30 SPMH=FTA(J)+(FTA(J+1)-FTA(J))*(Y-CFR(J))/(CFR(J+1)-CFR(J))
10 REQMH=REQMH+SPMH
100 RETURN

END

COMPILATION: NO DIAGNOSTICS.
SUBROUTINE MOTION(NL,NS)
COMMON WTMUCK,WTM,MLCO,TMUCK,KMAX,NCARS,LCLAS,NDC,ACCMAX,
*VELMAX,DFCEL,NSD,AFT,HAUL,NSECS,NSW,WTCA,FCAR,TPM,ADRT,
*CROSEC,GAHMAX,CTIME,LWID,TTM,TNL,MAX,MIN,SRID,RLS,
*CRL(15),FTA(15),NCLAS(120),LL(10),CTLOC(10),TLUC1(10),WTLOC1(10),
*FLOC0(10),G(100),O(100),DS(10),WTLOAD(10),SPEED(10),LOAD(10),
*DISTR(10),TIME(10),TPASS(10),WTTRN(10),CFL(13),CFD(13),
*CT(13),WTL(13),DSTOP(10),ST(13),CFS(13),TLOAD(10),TDUMP(10),
*CF(120,13),TV(120,13),T(10,30),S(10,30),LS(20),ILWTID,
*ILOAD,SCCTM,TBT2,TMOLE2,THIRD2,DELTH,IDEOS,TNL,
*FTAD,TMOLE1,TBIT1,THIRD1,NCREW,SHFTMH,HRPSH,NSHIFT,IMAN,ICYCLE

SUBROUTINE MOTION MOVES A TRAIN FROM ONE SWITCH TO THE NEXT

DIMENSION SGL(IO),MSS(IO),GREV(IO),DREV(IO),DSREV(IO)

REVERSE PERTINENT VARIABLES FOR LOADED TRAINS

IF ( LOAD(NL) .EQ. 0) GO TO 100

NNN=0
IF ( D(NSECS) .LT. DEPTH) 815,816,816

815 NSECS=NSECS-1

816 DO 101 I=1,NSECS
    IREV=NSECS-(I-1)
    GREV(IREV)=-G(I)
101 DREV(IREV)=D(I)

DO 99 I=1,NSECS
    G(I)=GREV(I)
99 D(I)=DREV(I)

DO 102 I=1,NSW
    J=NSW-(I-1)
102 DSREV(I)=HAUL-DS(J)

DO 98 I=1,NSW
98 D(S(I))=DSREV(I)

100 SBT=0.0
    SGL(1)=D(1)
    MSS(1)=1
    SGL(NSW)=0.0
    MSS(NSW)=0
    NS=NSW-1
    DO 103 I=2,NS
    DO 104 J=1,NSECS
        SST=SST+D(J)        IF (SST-NS(I)) 104,106,107
103 SST=SST+D(J)
104 SST=SST+D(J)
106 SGL(1)=D(J+1)
    MSS(1)=J+1
    GO TO 103
107 SGL(1)=SST-DS(I)
    MSS(1)=J
GO TO 103
104 CONTINUE
103 CONTINUE
TIME(NL)=0.0
TSTOP(NL)=0.0
TPASS(NL)=0.0
DSTOP(NL)=0.0

START LOCO IN MOTION

IF ( LOAD(NL) .EQ. 0) GO TO 86
JL=NSw=(NS-1)

SLEFT=NS(NS+1)-DS(NS)
GLEFT=GL(IS)
MM=MSS(NS)
FRFC=( FLOC0(NL)*WLOC(NL)+WTLOAD(NL)+WTCAR*NCARS)*FCAR
IF ( LOAD(NL) .EQ. 1) GO TO 77
WTRN(NL)=WLOC(NL)+NCARS*WTCAR
77 GFC=( 2.0*G(MM))*WTRN(NL)
REQTF=FRFC+GFC
IF ( SPEED(NL) ) 110,110,111
110 AVATF=(T(NL,1)+T(NL,2))/2.0
111 ACCFC=AVATF-REQTF
ACCR=ACFC*3.2/(WTRN(NL)*20.0,0)
IF ( ACCR ) 115,116,117
115 TD=0.0
DD=0.0
IF ( SPEED(NL) ) 130,130,131
130 IF ( LOAD(NL) .EQ. 1) GO TO 132
PRINT 133, NL,MM,MM,ACCW,WLOC(NL),WTLOAD(NL),WTCAR*NCARS,FCAR,
GTRN(NL)*FLOC0(NL),FRFC,GFC,REQTF,AVATF,ACCFC,T(NL,1),T(NL,2)
133 FORMAT (1HO,'LOC0 NO.=',I3,'WITH LOAD CANNOT NEGOTIATE SECTION

NUMBR ','I4 ','3X,'GRADE=','F10.4,3X,'ACCR=','F12.4 ',
'3F10.2,15,3F10.2 ' ,7F12.3)
STOP
132 PRINT 134, NL,MM,MM,ACCW,WLOC(NL),WTLOAD(NL),WTCAR*NCARS,FCAR,
GTRN(NL)*FLOC0(NL),FRFC,GFC,REQTF,AVATF,ACCFC,T(NL,1),T(NL,2)
134 FORMAT (1HO,'LOC0 =',I3,'WITH LOAD CANNOT NEGOTIATE SECTION NUMBR

','I4 ','3X,'REVERSED GRADE=','F10.4,3X,'ACCR=','F12.4 ',
'3F10.2,15,3F10.2 ' ,7F12.3)
STOP
131 IF ( GLEFT=SLEFT) 135,135,130
135 IF ( SPEED(NL) .EQ. (NL,2)) 136,136,137
136 ADEC=-ACCR
TDD=SPEED(NL)/ADEC
TD=TDD+DD
IF ( DD=GLEFT) 130,130,138
13A DTR=DD=GLEFT
TD=TD-TDD
DO=DD-TDD
SP=SPEED(NL)
ASA=SPEED(NL)*SPEED(NL)-2.0*ADEC*DTR
IF (ASA) 261,261,262
261 PRINT 263,ASA
263 FORMAT (1X,*VARIABLE ASA IN MOTION HAS NEGATIVE SIGN*, F10.3)
STOP
262 CONTINUE
SPEED(NL)=SQR(SPEED(NL)*SPEED(NL)-2.0*ADEC*DTR)
TT1=(SF-SPEED(NL))/ADEC
TD1=DTR
D1=DD+TD1
T1=TD+TT1
GO TO 151
137 ADEC=ACCR
SP=SPEED(NL)
N7=KMAX-1
DO 139 I=1,N7
55 IF (SPEED(NL)=S(NL,1)) 139,139,50
55 IF (SPEED(NL)=S(NL,1+1)) 140,140,139
139 CONTINUE
PRINT 141, SPEED(NL),NL
141 FORMAT (1H0,*SPEED=*,F10.3,*OF LOCO NUMBER*,13,*COULD NOT BE FOUND IN ITS CHARACTERISTIC CURVE*)
STOP
142 KK=I
TTD=(SPEED(NL)-S(NL,KK))/ADEC
TD=(SPEED(NL)*SPEED(NL)-S(NL,KK)*S(NL,KK))/(2.0*ADEC)
TU=TD+TTD
Du=DD+TD
IF (DD=GLDF) 142, 138, 138
142 SP=SPEED(NL)
AVA1=T(NL,KK)+(T(NL,KK+1)-T(NL,KK))*(SPEED(NL)-S(NL,KK))/
(((S(NL,KK+1)-S(NL,KK))))
SPEED(NL)=S(NL,KK)
AVATF=(AVA1+T(NL,KK))/2.0
ACCFC=AVATF-RLQTF
ACCR=ACCCFC*32.2/(WTTR(NL)*20UJ,0)
IF (ACCR) 135, 143, 144
144 IF (ACCMAX=ACCR) 990,991,991
99 ACCR=ACCMAX
991 SPEED(NL)=S(NL,KK)+ACCR*(S(NL,KK+1)-S(NL,KK))/(ADEC+ACCR)
TD1=(SPEED(NL)-S(NL,KK))/ADEC
DU1=(SPEED(NL)*SPEED(NL)-S(NL,KK)*S(NL,KK))/(2.0*ADEC)
TU=TD+TD1
Du=DD+DU1
143 TD2=(GLDF-DD)/SPEED(NL)
DU2=(GLDF-DD)
D1=DD+DU2
T1=TD+TD2
GO TO 151
CALCULATE THE CASE OF ZERO ACCELERATION

116 IF ( SPEED(NL)) 130,130,145
145 D1=GLEFT
T1=GLEFT/SPEED(NL)
SP=SPEED(NL)
GO TO 151

CALCULATE THE CASE OF POSITIVE ACCELERATION

117 IF ( ACCR=ACCMAX) 146,146,147
147 ACCR=ACCMAX
146 N7=KMAX-1
148 CONTINUE
149 CONTINUE
PRINT 149, NL, SPEED(NL)
149 FORMAT (1HO,'LOCO NUMBER',*,I3,' HAD BAD INPUT DATA OF SPEED=',F10.2)
STOP
150 KK=I
T1=(S(NL,KK+1)-SPEED(NL))/ACCR
D1=T1*(SPEED(NL)+(T1*(ACCR/2.0)))
D2=GLEFT
V=V=SPEED(NL)*SPEED(NL)+2.0*ACCR*D2
T2=(SQRT(V-V-SPEED(NL)))/ACCR
IF ((T1-D2) 351,152,152
152 D1=D2
T1=T2
SP=SPEED(NL)
SPEED(NL)=SP+T1*ACCR
GO TO 151
351 SP=SPEED(NL)
SPEED(NL)=S(NL,KK+1)
151 DSTOP(NL)=SPEED(NL)*SPEED(NL)/(2.0*DECEL)
153 DISTR(NL)=DISTR(NL)+D1
TIME(NL)=TIME(NL)+T1
GLEFT=GLEFT-D1
SLEFT=SLEFT-D1
IF (SLEFT<0.5) 154,154,507
507 IF (GLEFT<0.5) 156,156,157
156 MM=MM+1
GLEFT=D(MM)
GFC=(20.0*G(MM))*WTTRV(NL)
RLQT=FRFC+GFC
151 CONTINUE
517 IF (SLEFT=GLEFT) 501,501,500
501 IF (USTOP(NL)=SLEFT) 502,503,154
502 IF (SPEED(NL)=VELMAX) 506,505,505
505 D1=SLEFT-USTOP(NL)
SPEED(NL)=VELMAX
**T**1 = D1 / SPEED(NL)
**TIME**(NL) = **TIME**(NL) + T1
**DISTR**(NL) = **DISTR**(NL) + D1

**STOP**(NL) = SPEED(NL) / DEC
**TPASS**(NL) = **DSTOP**(NL) / SPEED(NL)
GO TO 405

506 **GL**EFT = SLEFT
GO TO 500

507 **CONT**INUE
IF (SPEED(NL) = VELMAX) 158, 216, 216

216 SPEED(NL) = VELMAX
GO TO 116

158 DO 159 K = 1, KMAX
IF (SPEED(NL) - S(NL + K)) 159, 52, 52
52 IF (SPEED(NL) - S(NL + K + 1)) 160, 159, 159

159 **CONT**INUE
PRINT 161, NL, SPEED(NL), (S(NL + K)), K = 1, KMAX

161 FORMAT (2H0, 'LOCO NUMBER', I3, ' AT SPEED =', F10.3, 'HAD BAD INPUT DATA AND STOPPED/', ' ', F12.3)
STOP

162 **K**K = K
AVAI = T(NL, K) + (T(NL, K + 1) - T(NL, K)) * (SPEED(NL) - S(NL, K)) / (S(NL, K + 1) - S(NL, K))
AVATF = (AVAI + T(NL, K + 1)) / 2.0
GO TO 120

154 SLEFT = DI + SLEFT
OTRD = (DI + **DSTOP**(NL)) - SLEFT
IF (ACCR) 400, 401, 400

401 DI = DI - OTRD
**DISTR**(NL) = **DISTR**(NL) - OTRD
**TIME**(NL) = **TIME**(NL) - T1
T1 = D1 / SPEED(NL)
**TIME**(NL) = **TIME**(NL) + T1
**STOP**(NL) = SPEED(NL) / DEC
**TPASS**(NL) = **DSTOP**(NL) / SPEED(NL)
GO TO 405

406 V2 = (2.0 * SLEFT + SP * SP / ACCR) * (ACCR * DEC / (ACCR + DEC))
IF (V2) 411, 412, 412

411 PRINT 413, V2, SP, SPEED(NL), ACCR, SLEFT, D1, **DSTOP**(NL)

417 FORMAT (1X, 'V2 IN MOTION HAS NEGATIVE VALUE', ' ', 2X, 'F7.10, 3)
STOP

412 SPEED(NL) = SQRT(V2)
**DISTR**(NL) = **DISTR**(NL) - D1
**TIME**(NL) = **TIME**(NL) - T1
**DSTOP**(NL) = SPEED(NL) * SPEED(NL) / (2.0 * DEC)
D1 = (SPEED(NL) * SPEED(NL) - SP * SP) / (2.0 * ACCR)
**DISTR**(NL) = **DISTR**(NL) + D1
T1 = (SPEED(NL) - SP) / ACCR
**TIME**(NL) = **TIME**(NL) + T1

155 **STOP**(NL) = SPEED(NL) / DEC
**TPASS**(NL) = **DSTOP**(NL) / SPEED(NL)
GO TO 405

405 IF (LOAD(NL) .EQ. 0) GO TO 455
NS=NSW-NS+1
DO 861 I=1,NSLCS
IREV=NSLCS-(I-1)
GREV(I)=G(IREV)
861 DREV(IREV)=D(I)
DO 862 I=1,NSLCS
G(I)=GREV(I)
862 D(I)=DREV(I)
DO 863 I=1,NSW
J=NSW-(I-1)
863 DREV(I)=HAUL=DS(J)
DO 864 I=1,NSW
864 D(I)=DREV(I)

IF ( NSW .EQ. 3 ) GO TO 455
NSEC=NSEC+1
455 RETURN
END

SUBROUTINE LOADNG(NL)
COMMON TMUCK,WTIM,NLUCO,TMUCK,KMAX,NCARS,LCLAS,NDC,AMMAX,
VELMAX,DECEL,NSC,D1,AFT,HAUL,NSC,NW,WTAC,FCAR,SPS,ADRT,
CROSEC,GAMMA,CTIME,LWTID,TTM,TNL,PLMAX,SRF,ILS,
CFR(13),FTA(13),NCLAS(120),LL(10),CTLOC(10),TLOC1(10),TLOC2(10),
FLUCO(10),G1(100),D(100),DS(10),WTL(10),SPEED(10),LOAD(10),
DIST(10),TIME(10),TSOP(10),TSTOPD(10),TSTOP(10),CFL(13),CFD(13),
C(13),DLF(13),DSTOPD(10),DSTOP(10),ST(13),CFS(13),TLOAD(10),TDUMP(10),
LC(120),T(120),T(10),T(10,3),S(10,3),S(10,30),LS(20),ILWTID,
TLOAD,SCCTM,TRIT2,TMOL2,THRD2,DELTH,TEOS,TNL,
TAD,TMOL1,TRIT1,THRD1,NCLLY,SHFTH,HRPSH,NSHIFT,IMAN,ICYCLE

SUBROUTINE LOADNG SIMULATES THE LOADING OF THE TRAIN

TLOAD(NL)=0.0
WTL(10)=0.0
UIT=U(0)
NL=NSC-1
DO 60 K=1,NL
60 IF ( CFS(K) .LT. U .AND. CFS(K+1) .GE. U ) GO TO 61
PRINT 62
62 FORMAT ( 2X, 'STOPPED BY AN ERROR IN SEARCHING THROUGH CFS(I)' )
STOP
61 TLLD=ST(K)+(ST(K+1)-ST(K))**((Y-CFS(K))/(CFS(K+1)-CFS(K)))
TLOAD(NL)=TMUCK+TLLD
WTL(10)=WTMUCK
WTMUCK(10)=WTL(10)+NL-1
RETURN
END
SUBROUTINE DUMP(NL)
COMMON WTWUCK,WTIM,NLUCO,TMUCK,KMAX,NCARS,LCLAS,NUC,ACCMAX,
$VELMAX,DECEL,NSC,11,AFT,HAUL,NSEC,NSW,WTCA,FCAR,TPM,AURT,
$CRSSEC,GAMMA,CME,LWTID,LTM,TLN,TIMAX,NRH,ILS,
$CFR(13),FTA(13),NCLAS(120),LL(10),CTLOC(10),TLOC1(10),TLOC(10),
$FLOCO(10),G(100),D1(100),DS(10),WTLOAD(10),SPEED(10),LOAD(10),
$DISTR(10),TIME(10),TSTOP(10),PASS(10),WTTRN(10),CF(13),CFD(13),
$CT(13),WT(13),DSTOP(10),S(13),CFS(13),TLOAD(10),TDUMP(10),
$CF(120,13),TV(120,13),T(10,30),S(10,30),LS(20),LWTID
*IDLOAD,SCCTM,TRIT2,TMOLE2,THIRD2,DELTH,IDEOS,TNL
*FTA,TMOLE1,TRIT1,THIRD1,NCRLW,SHFTMH,HRP,NSHIFT,IMAN,ICYCLE

SUBROUTINE DUMP SIMULATES THE DUMPING OF THE TRAIN

Y=RAND(N)
NS=NCC-1
DO 100 J=1,NS
100 IF ( CF(J), LT, Y AND, CFD(J+1), GE, Y) GO TO 110
PRINT 120,Y,(CF(J),J=2,NDC)
120 FORMAT (1H5,1IN DUMPING CYCLE RAND=,'F10.6',2X,'CANNOT BE FOUND'
*/'',3F10.6)
STOP
110 TDUMP(NL)=CT(J)+(CT(J+1)-CT(J))*((Y-CFD(J))/(CFD(J+1)-CFD(J)))
LOAD(NL)=0
RETURN
END

COMPIILATION: NO DIAGNOSTICS.
SUBROUTINE MUCK

COMMON WTMUCK, WTM, NLCO, TMUCK, KMAX, NCARS, LCLAS, NUC, ACCMAX,
VELMAX, DECEL, NSC, D1, AFT, HAUL, NSECS, NSW, WTCAR, FCAR, TPM, ADRT,
CROSEC, GAMMA, CTIME, LWID, TTM, TNL, TIMAX, NRGB, ILS,
CFR(13), FTA(13), NCLAS(120), LL(10), TLOC1(10), TLOC2(10), WLOC(10),
FILE(10), G(100), D(100), DS(10), WLOAD(10), SPEED(10), LOAD(10),
DIST(10), TIME(10), TSTOP(10), IPASS(10), WTRN(10), CFL(13), CFU(13),
CT(13), WTL(13), NSTOP(10), ST(10), CFS(13), TLOAD(10), TDUMP(10),
CF(120, 13), TV(120, 13), T(10, 30), S(10, 30), LS(20), ILWTID,
1DLOAD, SCCTM, TRIT2, TMOLE2, THRT2, DELTH, IDEOS, TNLT,
FTAD, TMOL1, TRIT1, THRT1, NCRLW, SHFTM, HRPSH, NSHIFT, IMAN, ICYCLE

SUBROUTINE MUCK DETERMINES THE WEIGHT OF THE MUCK LOADED INTO THE CARS

AFT=0.0
TMUCK=0.0
WTMUCK=0.0
NSC=LCLAS=1
DO 10 I=1, NCARS
Y=RANUN(N)
DO 20 J=1, NSC
1 IF (CFL(J) .LT. Y .AND. CFL(J+1) .GE. Y ) GO TO 30
PRINT 25, Y
25 FORMAT ( 1HO, " IN THE LOADING CYCLE RAND=" , F10.6, 'X, 'CANNOT BE FOU
AND IN CFL(I)" )
PRINT 26 ( CFL(K), K=1, LCLAS)
26 FORMAT ( F12.6)
STOP

WTM=WTM(J)+(WTL(J)+WTL(J))*((Y-CFL(J))/(CFL(J+1)-CFL(J)))
10 WTMUCK=WTMUCK+WTM
TPM=ADR+(CROSEC*GAMMA/(60.0*2.000,0)
TMUCK=WTMUCK/TPM
AFT=ADR+TMUCK/30.0
RETURN
END

COMPILED: NO DIAGNOSTICS.
SUBROUTINE TRANS

DIMENSION CFMS(15), TMS(15), CFMD(15), TMD(15)

SUBROUTINE TRANS CONTROLS THE SIMULATION OF THE MATERIALS HANDLING SUBSYSTEM

READ IN CONTROL CARDS

IF ( ICYCLE .NE. 0) GO TO 650
IF ( IDEOS .EQ. 1) GO TO 699
IF ( ILS .NE. 0) GO TO 652

READ IN INPUT DATA FOR CONTINUOUS MATERIAL HANDLING SYSTEM

READ 653, EFFCSA, FLUVEL, GMUCK

653 FORMAT (3F10.3)
READ 654, NCPTS

654 FORMAT (I5)
READ 655 (CFMS(I), TMS(I), I = 1, NCPTS)

655 FORMAT (6F10.3)
READ 654, NCST
READ 655 (CFMD(I), TMD(I), I = 1, NCST)

INITIALIZE VARIABLES TO ZERO

SPWT = 0.0
TBELT1 = 0.0
TBELT2 = 0.0
TBELT3 = 0.0
DMNTM = 0.0
SUMDLY = 0.0

CALCULATE PW T AND SPWT

Y = RANU(N)

DO 656 I = 1, NCPTS

656 IF (CFMS(I) .LT. Y .AND. CFMS(I+1) .GE. Y) GO TO 657
PRINT 658, Y
658 FORMAT (1X,'RAND= ', F12.6,3X,'CANNOT BE FOUND IN CFMS-1MS CURVE')
STOP
657 PWT=TIMS(I)+(TIMS(I+1)-TIMS(I)) *((Y-CFMS(I))/(CFMS(I+1)-CFMS(I)))
SPWT=PWT*60.0
IF ( ILS .EQ. 0) GO TO 701
PRINT 28
28 FORMAT ( ' ILS WAS NOT ZERO')
STOP
652 CAPMH=FFFCSA*FLUVEL*GHUCK/200.0
TPM=ADRT*CROSNC*GAMMA/(60.0*2.00.0)
MOTM=FADT*60.0/ADRT
IF ( CAPMH-TPM) 659, 660
COUNT TIME DELAYED AND MH WAITED DUE TO HANDLING SYSTEM
659 ADJTPM=CAPMH
AMOTM=MOTM*TPM/ADJTPM
DIFTM=AMOTM-MOTM
SUMDLY=SUMDLY+DIFTM
CTIME =CTIME+DIFTM
TBELT1=TBELT1+DIFTM
TBU1=TBU1+DIFTM
TMOLE1=TMOLE1+DIFTM
THIRD1=THIRD1+DIFTM
DIFTM=ADJTPM*NCREW/60.0
670 IF ( SHFTMH-DIFTM) 673, 676, 673
673 SHFTMH=SHFTMH*NCREW*HRS
NSHIFT=NSHIFT+1
GO TO 677
675 SHFTMH=SHFTMH*NCREW*HRS
NSHIFT=NSHIFT+1
GO TO 677
676 SPWT=0.0
C
MH FOR REPAIR
Y=RAND(N)
DO 663 J=1,NCST
663 IF ( CFMD(J) .LT. Y .AND. CFMD(J+1) .GE. Y) GO TO 664
PRINT 605, Y
665 FORMAT (1X,'RAND= ', F12.5,3X,'NEVER BE FOUND IN CFMD-TMD')
STOP
664 DNTM=TMD(J)+(TMD(J+1)-TMD(J)) *((Y-CFMD(J))/(CFMD(J+1)-CFMD(J)))
C
CALCULATE DOWNTIME
Determine TBELT1 and waiting times

CALCULATE PWT and SPWT

PRINT SUMMARY OF CONTINUOUS MATERIAL HANDLING SYSTEM

PRINT 673

FORMAT (1HO, ' SUMMARY OF CONTINUOUS SYSTEM')
TBELT1=TBELT1/60.0
TBELT2=TBELT2/60.0
TBELT3=TBELT3/60.0
SUMDLY=SUMDLY/60.0
PRINT 673

FORMAT (1HO, ' WORK TIME TIME DELAYED DOWN TIME')
PRINT 674, TBELT1, TBELT2, TBELT3

FORMAT (3F12.3)
PRINT 674, SUMDLY

FORMAT (1X, ' TOTAL TIME DELAYED BY THE CONTINUOUS MATERIAL HANDLING $ SYSTEM', F12.3)
GO TO 701
CYCLIC MATERIAL HANDLING SYSTEM

55 IF ( IDEOS .EQ. 1) GO TO 299
ILDC=0
ILC=0
DO 50 I=1,NLOCO
50 LC(I)=0
IF ( ILS .EQ. 0) GO TO 99
IF ( ILWTD .EQ. 1) GO TO 90
98 IF ( LWTD .EQ. 0) GO TO 301
WTIM=0
GO TO 301
99 READ 100,NLOCO,KMAX,NSEC,NSD,NDC,NCLAS,NSC
100 FORMAT (I5)
PRINT 101,NLOCO,NSEC,NDC
101 FORMAT (I11,15x,'NUMBER OF LOCOMOTIVES =',I5,'0',3x,
        'NUMBER OF GRADE SECTIONS =',I5,'0',3x,'NUMBER OF CARS PER TRAIN
        =',I5,7x)
READ IN ALL CHARACTERISTIC CURVES OF LOCOMOTIVES
READ 102 ((T(I,J),S(I,J), J=1,KMAX), I=1,NLOCO)
102 FORMAT (6F10.3)
READ IN THE CUMULATIVE FREQUENCY CURVES OF LOADING AND DUMPING
READ 103 ( CFL(I),WTL(I), I=1,LCLAS)
READ 103 ( CFS(I), ST(I), I=1,NSC)
READ 103 ( CFD(I), CT(I), I=1,NDC)
103 FORMAT (6F10.3)
READ IN THE WEIGHT AND FRICTION COEFFICIENT OF THE LOCOMOTIVES
READ 103 (WTLOC(I),FLUCO(I), I=1,NLOCO)
READ IN THE PROFILE OF THE TUNNEL
READ 103 (G(I),D(I), I=1,NSEC)
READ IN THE WEIGHT AND FRICTION COEFFICIENT OF THE CARS
READ 103 , WTCAR,FCAR
READ IN SPEED LIMITATIONS
READ 103 , ACCMAX,VELMAX,DECEL
READ IN THE DISTANCE BETWEEN SWITCHES, DENTH, AND THE NUMBER OF
SWITCHES BETWEEN THE DUMPING STATION AND THE PORTAL
LOAD 104, DISW, DELTH
104 FORMAT (2F19.3, 15)

C C C

INITIALIZE VARIABLES

DO 105 I=1,NLOC0
DISTR(I)=0.0
LOAD(I)=0
WTLOAD(I)=0.0
WTTRN(I)=0.0
TLOC1(I)=0.0
TLOC2(I)=0.0
TLOC3(I)=0.0
TLOC4(I)=0.0
TIME(I)=0.0
CTLOC(I)=0.0
SPEED(I)=0.0
TPASS(I)=0.0
TSTOP(I)=0.0
DSTOP(I)=0.0

105 CONTINUE
LIL=0
NLDE=0
NLDL=0
HAUL=0.0
TSEC=0.0
NSA=0
SAFT=0.0
WTU=0.0
WTG=0.0

C C C

LOCATE AND COUNT THE SWITCH POINTS

DO 106 J=1,NSLP
106 HAUL=HAUL+D(J)
TNLT=TNL+HAUL
NSW=2
DH=HAUL

109 IF (DH=DISW) 107,108,108
108 NSW=NSW+1
DH=DH-DISW
GO TO 109
107 NN=NSW-1
DMS=DH
IF (NN .EQ. 1) GO TO 110
110 DO 111 I=2,NN
111 DS(I)=DISW*(I-1)
110 DO (I)=0.0
DS(NSW)=HAUL
COUNT THE NUMBER OF SECTIONS AT THE START OF THE SIMULATION AND DETERMINE THE DISTANCE TO THE LAST SECTION

NSECS=NSWP+1
DU=D(NSECS)
D(NSECS)=0.0

LOCATE THE LOCOMOTIVES AT THE STARTING POINT

DO 199 I=1,NLOCO
   199 IDE(I)=0
      NLDE=0
      NLDL=0
      IF (NLOCO=NSW) 311,112,113
   311 NWLD=0
      N2=NSW-NLOCO +1
      DO 114 I=NSW,N2,-1
         J=NSW-I+1
         LS(I)=2
      114 LL(I)=J
      N1=N2+1
      DO 115 I=1,N1
         115 LS(I)=0
         GO TO 116
   112 NWLD=1
      IDE(NLOCO)=1
      NLDE=1
      DO 117 I=1,NSW
         J=NSW-I+1
         LS(I)=2
      117 LL(I)=J
      GO TO 116
   113 NWLD=NLOCO-NSW+1
      LMN=0
      DO 312 I=NSW,NLOCO
         LMN=LMN+1
      312 IDE(I)=LMN
      NLDE=NWLD
      DO 118 I=1,NSW
         J=NSW-I+1
         LS(I)=2
      118 LL(I)=J
      NEX=NSW+1
      DO 97 I=NEX,NLOCO
         97 LL(I)=1
   116 CONTINUE
   DO 120 I=1,NLOCO
      120 IUL(I)=0
      IF ( ILS .EQ. 0) GO TO 701
START THE SIMULATION WITH LOCOMOTIVE NUMBER 1 AT THE LOADING POINT

90 NL=1
CALL LOADING(NL)

IDLOAD=1
CCTM=TLOAD(NL)*2000.0*60.0/(CROTSEC*GAMMA*ADRT)
SCCTM=SCCTM+CCTM
TLOC1(NL)=TLOC1(NL)+TLOAD(NL)
CTLOC(NL)=CTLOC(NL)+TLOAD(NL)
SAFT=SAFT+AFT
WTG=WTG+WLOAD(NL)
NS=LL(NL)
LS(NS)=1

DETERMINE THE LOADING TIME FOR THE LOCOMOTIVE

DO 119 J=2,NLOC0
  TLOC4(J)=TLOAD(NL)
  119 CTLOC(J)=TLOAD(NL)

CALL MOTION(NL,NS)

CTLOC(NL)=CTLOC(NL)+TIME(NL)/60.0
TLOC3(NL)=TLOC3(NL)+TIME(NL)/60.0
IF (LOAD(NL),NE,1) GO TO 120
LS(NS)=LS(NS)+2
NS=NS+1
LL(NL)=NS
LS(NS)=LS(NS)+2
NS1=NS-1
NS2=NS
GO TO 121

120 LS(NS)=LS(NS)-1
NS=NS-1
LL(NL)=NS
LS(NS)=LS(NS)+1
NS1=NS+1
NS2=NS
GO TO 121

CONTINUE
IF (LOOPHR,NE,0) GO TO 301
PRINT 411
411 FORMAT (1HO, 'LOCO NO. CLOCK TIME MOVING SW NO. LOAD', 25X
$,'FROM TO' $)
PRINT 412, NL, CTLOC(NL), NS1, NS2, LOAD(NL)
412 FORMAT (4X,I3,4X,F10.2,5X,I2,4X,I2,7X,I1)
CHOOSE THE LOCOMOTIVE HAVING THE SMALLEST VALUE OF CTLOC(I)

301 IL=0
M=1
DO 122 J=2,NLOC0
122 IF(CTLOC(M) GT CTLOC(J)) M=J
NL=M

EMPTY TRAIN

IF ( LIL .EQ. 0) GO TO 200
DO 221 I=1,LIL
221 IF ( NL .EQ. LLW(I)) GO TO 222
GO TO 200
222 M=I
IF ( LLW(M) .EQ. LIL) GO TO 224
LLW(M)=0
M=M+1
DO 223 J=M1,LIL
K=J-1
223 LLW(K)=LLW(J)
224 LIL=LIL-1
230 IF ( LOAD(NL) .EQ. 1) GO TO 123
NS=LL(NL)
IF ( NS .EQ. NSW) GO TO 124
IF ( NS .EQ. 1) GO TO 125
IF ( LS(NS+1) .EQ. 2 OR LS(NS+1) .EQ. 3) GO TO 360
GO TO 361
360 IF ( ILC .EQ. 0) GO TO 129
NSS=NS+1
DO 362 I=1,NLOC0
362 IF ( LL(I) .EQ. NSS ) GO TO 363
PRINT 364, NSS
364 FORMAT ( ' NSS AT 364 IN TRANS', I15)
STOP
363 II=I
IF ( LOAD(II) .EQ. 0) GO TO 360
Jl=II+1
DO 365 K=J1,NLOC0
365 IF ( LL(K) .EQ. NSS) GO TO 367
PRINT 364, NSS
STOP
367 II=K
368 DO 368 I=1,ILC
368 IF ( LC(I) .EQ. II) GO TO 369
GO TO 129
369 ILC=ILC+1
LC(ILC)=NL
IDLC=1
GO TO 129
CTLOC(NL)=CTLOC(NL)+TPASS(NL)/60.0
DISTR(NL)=DISTR(NL)+DSTOP(NL)
TLOC3(NL)=TLOC3(NL)+TPASS(NL)/60.0

CALL MOTION(NL,NS)

IF (LOAD(NL) .EQ. 1) GO TO 127
LS(NS)=LS(NS)-2
NS=NS+1
LL(NL)=NS
LS(NS)=LS(NS)+2
NS1=NS-1
NS2=NS
GO TO 128

127 LS(NS)=LS(NS)-1
NS=NS-1
LL(NL)=NS
LS(NS)=LS(NS)+1
NS1=NS+1
NS2=NS
CONTINUE
CTLOC(NL)=CTLOC(NL)+TIME(NL)/60.0
TLOC3(NL)=TLOC3(NL)+TIME(NL)/60.0
IF (LOGPKT .NE. 0) GO TO 413
PRINT 411
PRINT 412, NL, CTLOC(NL), NS1, NS2, LOAD(NL)

413 IF (IL .EQ. 0) GO TO 300
IF (LIL .EQ. 0) GO TO 225
DO 192 1=1,IL
DO 228 J=1,LIL

229 IF (LW(I) .EQ. LLW(J)) GO TO 229
ATPS=0,J
GO TO 230
ATPS=FTPS

230 JJ=LW(I)
CTLOC(JJ)=CTLOC(JJ)+ATPS+TIME(NL)/60.0
192 TLOC4(JJ)=TLOC4(JJ)+ATPS+TIME(NL)/60.0
DO 231 1=1,IL
DO 232 J=1,LIL

232 IF (LW(I) .EQ. LLW(J)) GO TO 231
LIL=LIL+1
LLW(LIL)=LW(I)

231 CONTINUE
FTPS=TSTOP(NL)/60.0
DO 35 I=1,IL

35 LW(I)=0
IL=0
GO TO 300

225 DO 226 I=1,IL
JJ=LW(I)
CTLOC(JJ)=CTLOC(JJ)+TIME(NL)/60.0
226 TLOC4(JJ)=TLOC4(JJ)+TIME(NL)/60.0
LIL=IL
FIPS=TRIOP(NL)/60.0
DO 227 I=1,LIL
227 LLW(I)=LW(I)
   DO 36 I=1,LIL
36 LW(I)=0
   IL=0
   LW(I)=0
   GO TO 300
124 IF ( LS(NS),LE, 3,OK, LS(NS),LE, 1) GO TO 338
   DO 333 I=1,NLOC0
333 IF ( LL(I),LE, NS AND LOAD(I),LE, 1) GO TO 332
   PRINT 541
541 FORMAT ( 1X,'STOPPED AT 541 IN TRANS')
   STOP
332 INLC=1
   DO 333 I=1,ILC
333 IF ( LC(I),LE, INLC) GO TO 334
   GO TO 129
334 ILC=ILC+1
   LC(ILC)=NL
   ILC=1
   GO TO 129
339 TLLOC(NL)=TLLOC(NL)+TSTOP(NL)/60.0
   DISTR(NL)=DISTR(NL)+DSTOP(NL)
   TLLOC3(NL)=TLLOC3(NL)+TSTOP(NL)/60.0+TIME(NL)/60.0
   SPEED(NL)=0.0
   TPASS(NL)=0.0
   TSTOP(NL)=0.0
   DSTOP(NL)=0.0
   TIME(NL)=0.0
   CHECK THE TUNNEL LENGTH AND HAULAGE DISTANCE AND RELOCATE THE
   LAST SWITCH
IF ( SAFT ,LE, DELTH ) GO TO 180
   GO TO 189
180 SAFT=SAFT-DELTH
   HAUL=HAUL+DELTH
   TSEC=TSEC+DELTH
   DMS=DMS+DELTH
   IF ( TSEC-UD ) 181,182,182
181 TSEC=TSEC-DD
   D(NSECS)=DD
   NSECS=NSECS+1
   DD=D(NSECS)
182 D(NSECS)=TSEC
   IF ( DMS-DISW ) 183,184,184
184 DMS=DMS-DISW
DS(NSw)=HAUL-DMS
NSw=NSw+1
183 DS(NSw)=HAUL
IF ( HAUL = TNL) 189, 185, 185
185 PRINT 186
186 FORMAT (IHU, 'TUNNELLING WAS COMPLETED AND SIMULATION TERMINATED')
GO TO 299
189 CONTINUE
IF ( LWID .NL. 0) GO TO 700
WTM=CLLOC(NL)-CTIME
IF ( IL .EQ. 0) GO TO 701
DO 702 IL=1, IL
JU=LW(I)
LW(I)=0
CTC=CLLOC(NL)-CLLOC(JJ)
CLLOC(JJ)=CLLOC(NL)
702 TLOC(JJ)=TLOC(JJ)+CTC
IL=0
GO TO 701
703 IF ( IL .EQ. 1) GO TO 600
IL=0
CALL LOADING(NL)
CCTM=WLOAD(NL)*20*60*/600/(CROSEC*GAMMA*ADRT)
SCCTM=SCCTM+CCTM
WTG=WTG+WLOAD(NL)
CLLOC(NL)=CLLOC(NL)+TLOAD(NL)
TLOC1(NL)=TLOC1(NL)+TLOAD(NL)
LS(NS)=LS(NS)-1
SAFT=SAFT+AFT
IF ( IL .EQ. 0) GO TO 300
DO 191 IL=1, IL
JU=LW(I)
LW(I)=0
CLLOC(JJ)=CLLOC(JJ)+TLOAD(NL)
191 TLOC(JJ)=TLOC(JJ)+TLOAD(NL)
IL=0
GO TO 300
600 WTM=(CTIME+SCCTM)-CLLOC(NL)
IF ( WTM .GT. 601, 601, 602
601 IF ( IL .EQ. 0) GO TO 701
DO 931 IL=1, IL
JU=LW(I)
LW(I)=0
TLOC(JJ)=TLOC(JJ)+(CLLOC(NL)-CLLOC(JJ))
931 CLLOC(JJ)=CLLOC(JJ)+(CLLOC(NL)-CLLOC(JJ))
IL=0
GO TO 701
602 CLLOC(NL)=CLLOC(NL)+WTM
TLOC(NL)=TLOC(NL)+WTM
IF ( IL .EQ. 0) GO TO 300

DO 39 I=1,IL
JJ=LW(I)
LW(I)=0
CTLOC(JJ)=CTLOC(JJ)+WwTM
30 TLOC4(NL)=TLOC4(NL)+WwTM
IL=0
GO TO 300
129 CTLOC(NL)=CTLOC(NL)+TSTOP(NL)/SPEED(NL)
DISTR(NL)=DISTR(NL)+DSTOP(NL)
TLOC3(NL)=TLOC3(NL)+TSTOP(NL)/SPEED(NL)
SPEED(NL)=1.0
TSTOP(NL)=1.0
DSTOP(NL)=1.0
TIME(NL)=1.0
IF (ILC .NE. 1) GO TO 150
ILC=0
SSCC=CTIM-CTLOC(NL)
IF (SSCC) 335,335,330
336 CTLOC(NL)=CTLOC(NL)+SSCC
TLOC4(NL)=TLOC4(NL)+SSCC
335 IF (IL .EQ. 0) GO TO 340
DO 337 I=1,IL
JJ=LW(I)
LW(I)=0
TLOC4(JJ)=TLOC4(JJ)+(CTLOC(NL)-CTLOC(JJ))
CTLOC(JJ)=CTLOC(JJ)+(CTLOC(NL)-CTLOC(JJ))
ILC=ILC+1
337 LC(ILC)=JJ
IL=0
GO TO 340
150 IL=IL+1
LW(II)=NL
IF(IL .EQ. 1) GO TO 130
C C
ARANGE LW(IL) IN ASCENDING ORDER
C C
DO 131 I=1,IL
131 IA(I)=LW(I)
NA=IL-1
DO 132 I=1,NA
M=J
MA=J+1
DO 133 I=MA,IL
133 IF (IA(I) .LT. IA(M)) M=I
ITEMP=IA(J)
IA(J)=IA(M)
132 IA(M)=ITEMP
C C
SEARCH FOR THE SHORTEST VALUE OF CTLOC(I) EXCEPT FOR CTLOC(IL)
C
ML=1
N1=2
DO 134 J=1,IL
N1=IA(I)
IF ( NJ .EQ. ML ) GO TO 560
N1=1
GO TO 561
ML=ML+1
N1=ML+1
IF ( I .NE. IL ) GO TO 134
IF ( ML .EQ. NLOC0 ) GO TO 139
GO TO 561
134 CONTINUE
561 DO 562 J=NK,IL
N1=IA(J)
IF ( NJ .EQ. N1 ) GO TO 563
N2=NJ-1
DO 137 K=N1,N2
137 IF ( CTLOC(ML) .GT. CTLOC(K) ) ML=K
563 N1=NJ+1
562 CONTINUE
IF ( NJ .EQ. NLOC0 ) GO TO 139
581 DO 140 J=N1,NLOC0
140 IF ( CTLOC(ML) .GT. CTLOC(J) ) ML=J
139 ML=ML
GO TO 200
130 NJ=LOW(IL)
IF ( NJ .EQ. 1 ) GO TO 141
ML=1
N1=2
IF ( NJ .EQ. 2 ) GO TO 142
N2=NJ-1
DO 143 J=N1,N2
143 IF ( CTLOC(ML) .GT. CTLOC(J) ) ML=J
142 N1=1J+1
N4=NLOC0
IF ( NJ .EQ. NLOC0 ) GO TO 144
DO 145 J=N3,N4
145 IF ( CTLOC(ML) .GT. CTLOC(J) ) ML=J
144 ML=ML
GO TO 200
141 ML=2
N1=3
N2=NLOC0
DO 246 J=N1,N2
246 IF ( CTLOC(ML) .GT. CTLOC(J) ) ML=J
NL=ML
GO TO 200
125 IF ( NLUE .EQ. 1 ) GO TO 146
IF ( IDE(NL) .EQ. 1 ) GO TO 146
IF ( IILC .EQ. 0 ) GO TO 150
N11=IDE(NL)-1
DO 341 I=1,N11
DO 342 J=1,NLOC0
342 IF ( IDE(J) .EQ. I) GO TO 343
PRINT 344, I
344 FORMAT ( ' IDE(NL)=I AT 344 IN TRANS COULD NOT BE FOUND', I5)
STOP
343 JNL=J
DO 345 K=1, ILC
345 IF ( JNL .EQ. LC(K)) GO TO 346
GO TO 341
346 ILC=ILC+1
LC(ILC)=NL
SSCC=CTTM-CTLOC(NL)
CTLOC(NL)=CTLOC(NL)+SSCC
TLOC4(NL)=TLOC4(NL)+SSCC
IF ( IL .EQ. 0) GO TO 371
DO 372 K=1, ILC
JJ=JW(K)
LW(K)=0
ILC=ILC+1
LC(ILC)=JJ
TLOC4(JJ)=TLOC4(JJ)+(CTTM-CTLOC(JJ))
372 CTLOC(JJ)=CTLOC(JJ)+(CTTM-CTLOC(JJ))
IL=0
GO TO 371
371 IF ( I .EQ. N11) GO TO 425
N10=I+1
GO TO 347
341 CONTINUE
GO TO 150
347 DO 348 L=N10, N11
DO 349 J=1, NLOC0
349 IF ( IDE(J) .EQ. L) GO TO 350
PRINT 351, L
351 FORMAT ( 1X, 'STOPPED AT 351 IN TRANS', I5)
STOP
353 JJ=J
TLOC4(JJ)=TLOC4(JJ)+(CTTM-CTLOC(JJ))
CTLOC(JJ)=CTTM
ILC=ILC+1
LC(ILC)=JJ
348 CONTINUE
425 IF ( IDE(NL) .EQ. NLDE) GO TO 343
N13=IDE(NL)+1
DO 420 I=N13, NLDE
DO 421 J=1, NLOC0
421 IF ( IDE(J) .EQ. I) GO TO 422
PRINT 423
423 FORMAT ( 1X, 'STOPPED AT 423 IN TRANS')
STOP
422 JJ=J
ILC=ILC+1
LC(I!C)=JJ
SSCC=CTTM-CTLOC(JJ)
CTLOC(JJ)=CTLOC(JJ)+SSCC
TLOC4(JJ)=TLOC4(JJ)+SSCC
CONTINUE
GO TO 340
146 IF ( LS(NS+1) .EQ. 0 .OR. LS(NS+1) .EQ. 1) GO TO 147
IF ( ILC .EQ. 0) GO TO 150
NNSS=NS+1
DO 352 I=1, NLOC0
352 IF ( LL(I) .EQ. NNSS) GO TO 353
PRINT 354, NNSS
354 FORMAT ( 1X, 'STOPPED AT 345 IN TRANS', I5)
STOP
II=I
IF ( LOAD(I) .EQ. 0) GO TO 356
J1=II+1
DO 355 K=J1, NLOC0
355 IF ( LL(K) .EQ. NNSS) GO TO 357
PRINT 354, NNSS
STOP
II=K
DO 358 I=1, ILC
358 IF ( LC(I) .EQ. II) GO TO 359
GO TO 150
SSCC=CTTM-CTLOC(NL)
TLOC4(NL)=TLOC4(NL)+SSCC
CTLOC(NL)=CTLOC(NL)+SSCC
ILC=ILC+1
LC(ILC)=NL
IF ( IL .EQ. 0) GO TO 340
DO 370 J=1, IL
JJ=LW(J)
LW(J)=0
TLOC4(JJ)=TLOC4(JJ)+(CTTM-CTLOC(JJ))
CTLOC(JJ)=CTLOC(JJ)+(CTTM-CTLOC(JJ))
ILC=ILC+1
370 LC(ILC)=JJ
IL=0
GO TO 340
NLDE=NLDE-1
DO 148 I=1, NLOC0
148 IF ( IDE(I) .EQ. 0) GO TO 148
IDE(I)=IDE(I)-1
148 CONTINUE
CALL MOTION ( NL, NS)
CTLOC(NL)=CTLOC(NL)+TIME(NL)/60.0
NS=NS+1
LL(NL)=NS
LS(NS)=LS(NS)+2
IF (LOGPRT .NE. 0) GO TO 165
NS1=NS-1
NS2=NS
PRINT 411
PRINT 412, NL, CTLOC(NL), NS1, NS2, LOAD(NL)

165 IF (IL .EQ. 0) GO TO 300
DO 151 1=1,IL
JJ=LW(I)
LW(I)=0
CTLOC(JJ)=CTLOC(JJ)+TIME(NL)/60.0
IL=0
GO TO 300

C TRAIN LOADED

123 NS=LL(NL)
IF (NS .EQ. 1) GO TO 155
IF (NS .EQ. NSW) GO TO 156
IF (NS .EQ. 2) GO TO 157
IF (LS(NS-1) .EQ. 1 .OR. LS(NS-1) .EQ. 3) GO TO 460
GO TO 157

460 IF (ILC .EQ. 0) GO TO 129
NS=NS-1
DO 462 I=1,NLOC0

462 IF (LL(I) .EQ. NSS) GO TO 465
PRINT 464, NSS
FORMAT (1X,'STOPPED AT 464 IN TRANS', I5)
STOP

463 II=I
IF (LOAD(II) .EQ. 1) GO TO 466
J=II+1
DO 465 K=J1,NLOC0

465 IF (LL(K) .EQ. NSS) GO TO 467
PRINT 464, NSS
STOP

467 II=K
DO 468 I=1,ILC

468 IF (LC(I) .EQ. II) GO TO 469
GO TO 129

469 ILC=ILC+1
LC(ILC)=NL
IELD0=1
GO TO 129

157 CTLOC(NL)=CTLOC(NL)+TPASS(NL)/60.0
DISTR(NL)=DISTR(NL)+DSTOP(NL)
TLOC3(NL)=TLOC3(NL)+TPASS(NL)/60.0
LS(NS)=LS(NS)-1

C CALL MOTION(NL,NS)
CTLOC(NL) = CTLOC(NL) + TIME(NL) / 60.0
TLOC3(NL) = TLOC3(NL) + TIME(NL) / 60.0
NS = NS - 1
LL(NL) = NS
LS(NS) = LS(NS) + 1
IF ( LOGPRT .NE. 0) GO TO 414
NS1 = NS + 1
NS2 = NS
PRINT 411
PRINT 412, NL, CTLOC(NL), NS1, NS2, LOAD(NL)
414 IF (NS .EQ. 1) GO TO 160
IF (IL .EQ. 0) GO TO 300
IF (LIL .EQ. 0) GO TO 525
DO 162 I = 1, IL
DO 526 J = 1, LIL
526 IF (LW(I) .EQ. LLW(J)) GO TO 529
ATPS = 0.0
GO TO 530
529 ATPS = FTP5
530 JJ = LW(I)
CTLOC(JJ) = CTLOC(JJ) + ATPS + TIME(NL) / 60.0
TLOC3(JJ) = TLOC3(JJ) + ATPS + TIME(NL) / 60.0
DO 531 I = 1, IL
DO 532 J = 1, LIL
532 IF (LW(I) .EQ. LLW(J)) GO TO 531
LIL = LIL + 1
LLW(LIL) = LW(I)
531 CONTINUE
FTP5 = TSTOP(NL) / 60.0
DO 37 I = 1, IL
37 LW(I) = 0
IL = 0
GO TO 300
525 DU 526 I = 1, IL
JJ = LW(I)
CTLOC(JJ) = CTLOC(JJ) + TIME(NL) / 60.0
TLOC3(JJ) = TLOC3(JJ) + TIME(NL) / 60.0
LIL = IL
FTP5 = TSTOP(NL) / 60.0
DO 527 I = 1, LIL
527 LW(I) = LW(I)
DO 38 I = 1, IL
38 LW(I) = 0
IL = 0
GO TO 300
163 CTLOC(NL) = CTLOC(NL) + TSTOP(NL) / 60.0
DISTR(NL) = DISTR(NL) + TSTOP(NL)
TLOC3(NL) = TLOC3(NL) + TSTOP(NL) / 60.0
SPEED(NL) = 0.0
NLDL = NLDL + 1
IDL(NL) = NLDL
IF ( IL .EQ. 0) GO TO 159
DO 163 I=1,IL
JJ=LW(I)
LW(I)=0
CTLOC(JJ)=CTLOC(JJ)+TSTOP(NL)/60.0+TIME(NL)/60.0
163 TLOC4(JJ)=TLOC4(JJ)+TSTOP(NL)/60.0+TIME(NL)/60.0
IL=0
159 IF ( NLDL .EQ. 1) GO TO 161
GO TO 300
C
161 CALL DUMP(NL)
C
CTLOC(NL)=CTLOC(NL)+TDUMP(NL)
TLOC2(NL)=TLOC2(NL)+TDUMP(NL)
NLDL=NLDL-1
IDL(NL)=IDL(NL)-1
NLDE=NLDE+1
LDE(NL)=NLDE
LS(NS)=LS(NS)-1
WTRRN(NL)=WTRRN(NL)-WLOAD(NL)
WTD=WTD+WLOAD(NL)
WLOAD(NL)=0.0
GO TO 300
196 IF ( LS(NS-1) .EQ. 1 .OR. LS(NS-1) .EQ. 3) GO TO 500
GO TO 511
506 IF ( ILC .EQ. 0) GO TO 150
NNSS=NS-1
DO 501 I=1,NLCO
501 PRINT 502
502 FORMAT ( 1X,'STOPPED AT 503 IN TRANS'),STOP
503 II=I
IF ( LOAD(II) .EQ. 1) GO TO 504
J1=II+1
DO 505 K=J1,NLCO
505 IF ( LL(K) .EQ. NNSS) GO TO 506
PRINT 507
507 FORMAT ( 1X,'STOPPED AT 507 IN TRANS'),STOP
508 II=K
504 DO 508 I=1,ILC
508 IF ( LC(I) .EQ. II) GO TO 509
GO TO 150
509 SSCC=CTM-CTLOC(NL)
TLOC4(NL)=TLOC4(NL)+SSCC
CTLOC(NL)=CTLOC(NL)+SSCC
ILC=ILC+1
LC(ILC)=NL
IF ( IL .EQ. 0) GO TO 340
DO 510 J=1,IL
JJ=LW(J)
Lw(J)=0
TLOC4(JJ)=TLOC4(JJ)+(CTTM-CTLOC(JJ))
CTLOC(JJ)=CTLOC(JJ)+(CTTM-CTLOC(JJ))
ILC=ILC+1

510 LC(ILC)=JJ
IL=0
GO TO 340

511 LS(NS)=LS(NS)-1

CALL MOTION (NL,NS)

NS=NS-1
LS(NS)=LS(NS)+1
LL(NL)=NS
CTLOC(NL)=CTLOC(NL)+TIME(NL)/60.*60
TLOC3(NL)=TLOC3(NL)+TIME(NL)/60.*60
IF (LOGPRT .NE. 0) GO TO 165
NS1=NS+1
NS2=NS
PRINT 411
PRINT 412, NL, CTLOC(NL), NS1, NS2, LOAD(NL)
GO TO 165

155 NDL=NDL+1

IUL(NL)=NDL
IF (NDL .EQ. 1) GO TO 167
IF (IUL(NL) .EQ. 1) GO TO 168
IF (ILC .EQ. 0) GO TO 150
N1=IUL(NL)-1
DO 381 I=1,N1
DO 382 J=1,NLOC0

382 IF (JUL(J) .EQ. I) GO TO 383
PRINT 384, I

FORMAT (1X,'STOPPED AT 384 IN TRANS', I5)
STOP

383 JNL=J
DO 385 K=1,ILC

385 IF (JNL .EQ. LC(K)) GO TO 386
GO TO 381

386 ILC=ILC+1
LC(ILC)=NL
SSCC=CTTM-CTLOC(NL)
CTLOC(NL)=CTLOC(NL)+SSCC
TLOC4(NL)=TLOC4(NL)+SSCC
IF (IL .EQ. 0) GO TO 391
DO 392 K=1,IL

392 JJ=LW(K)
Lw(K)=0
ILC=ILC+1
LC(ILC)=JJ
TLOC4(JJ)=TLOC4(JJ)+(CTTM-CTLOC(JJ))

392 CTLOC(JJ)=CTLOC(JJ)+(CTTM-CTLOC(JJ))
IL=0

391 IF ( I .EQ. N11) GO TO 426
  N10=I+1
  GO TO 387

381 CONTINUE
  GO TO 150

387 DO 388 L=N10,N11
  DO 389 J=1,NLOC0
  388 IF ( IDL(J) .EQ. L) GO TO 400
  PRINT 401
  401 FORMAT ( 1X, 'STOPPED AT 401 IN TRANS')
  STOP

400 JJ=J
  TLOC4(JJ)=TLOC4(JJ)+(CTTM-CTLUC(JJ))
  CTLOC(JJ)=CTLOC(JJ)+(CTTM-CTLOC(JJ))
  ILC=ILC+1
  LC(ILC)=JJ
  CONTINUE

426 IF ( INL(NL) .EQ. NLDL) GO TO 340
  N13=IDL(NL)+1
  DO 430 I=N13,NLDL
  DO 431 J=1,NLOC0
  431 IF ( IDL(J) .EQ. I) GO TO 432
  PRINT 433
  433 FORMAT ( 1X, 'STOPPED AT 433 IN TRANS')
  STOP

432 JJ=J
  ILC=ILC+1
  LC(ILC)=JJ
  SSCC=CTTM-CTLOC(JJ)
  CTLOC(JJ)=CTLOC(JJ)+SSCC
  TLOC4(JJ)=TLOC4(JJ)+SSCC
  CONTINUE
  GO TO 340

167 IF ( LS(NS) .EQ. 1) GO TO 169
  LS(NS)=LS(NS)-1
  GO TO 166

169 LS(NS)=LS(NS)+1
  GO TO 166

168 IF ( LS(NS) .EQ. 3 ) GO TO 160
  LS(NS)=LS(NS)+2

C 166 CALL DUMP(NL)

C

CTLOC(NL)=CTLOC(NL)+TUUMP(NL)
TLOC2(NL)=TLOC2(NL)+TUUMP(NL)
NLDL=NLDL-1
NLDU=NLDU+1
IDE(NL)=NLDE
WT1D = WT1D + WTLOAD(NL)  
WT1RN(NL) = WT1RN(NL) - WTLOAD(NL)  
WTLOAD(NL) = 0.0  
IF (NL DL *EQ. 1) GO TO 170  
DO 171 I=1,NLOCO  
IF ( IDL(I) *EQ. 0) GO TO 171  
IDL(I) = IDL(I) - 1  
171 CONTINUE  
GO TO 172  
172 IDL(NL) = IDL(NL) - 1  
DO 173 I = 1,IL 
JJ = LW(I)  
LW(I) = 0  
CTLOC(JJ) = CTLOC(JJ) + TUUMP(NL)  
173 TLOC4(JJ) = TLOC4(JJ) + TUUMP(NL)  
IL = 0  
GO TO 300  
175 IF ( CTLOC(I) *GE. TIMAX ) GO TO 176  
IF ( IDLOAD *EQ. 0) GO TO 301  
IF ( INL *ID TID *EQ. 1 *AND. NL *LT. NLOCO ) GO TO 301  
CTTM = CTIME + SCCTM  
IF ( ILC *EQ. 0) GO TO 734  
DO 733 JJ = 1,NLOCO  
DO 733 J = 1,ILC  
735 IF ( J *EQ. LC(J)) GO TO 736  
IF ( CTLOC(I) *LT. CTIM) GO TO 734  
ILC = ILC + 1  
LC(ILC) = I  
734 CONTINUE  
GO TO 736  
733 DO 720 I = 1,NLOCO  
IF ( CTLOC(I) *LT. CTTM) GO TO 720  
ILC = ILC + 1  
LC(ILC) = I  
720 CONTINUE  
736 IF ( ILC *EQ. 0) GO TO 301  
743 IF ( ILC *GE. NLOCO) GO TO 701  
IF ( ILC *EQ. 0) GO TO 301  
CONTINUE  
GO TO 736  
725 IF ( M *EQ. LC(I)) GO TO 726  
GO TO 728  
726 M = M + 1  
GO TO 727  
728 M = M + 1  
IF ( M1 *GT. NLOCO) GO TO 841  
DO 729 I = M1,NLOCO  
MJ = I
DO 730 J=1,ILC
  730 IF ( M J .LE. LC(J) ) GO TO 729
    IF ( CTLOC(M) .LE. CTLOC(MJ) ) GO TO 729
    M=MJ
  729 CONTINUE
  841 NL=NL+1
  176 GO TO 200
  177 FORMAT ( 1HI, 'SIMULATION TERMINATED BY MAX CLOCK TIME ALLOWED' )
    IDEOS=1
    GO TO 701
  C
  C PRINT THE SUMMARY OF SIMULATION AT THE END OF RUN
  C
  299 GO TO 193,HAUL,WTG,WTU
    PRINT 194
  194 FORMAT ( 1HI, 'LOCO NO, CLOCK TIME, LOADING TIME, DUMPING TIME, WAITING TIME, TRAVEL DISTANCE' )
    DO 195 I=1,NLOCO
      CTLOC(I)=CTLOC(I)/60.0
      TLOC1(I)=TLOC1(I)/60.0
      TLOC2(I)=TLOC2(I)/60.0
      TLOC3(I)=TLOC3(I)/60.0
      TLOC4(I)=TLOC4(I)/60.0
    195 PRINT 196, I, CTLOC(I), TLOC1(I), TLOC2(I), TLOC3(I), TLOC4(I), DISTRI(I)
    196 FORMAT ( 2X, I3, 3X, F12.3, 4(3X, F10.3), F15.3 )
      IDEOS=1
      GO TO 701
  END

COMPILATION: NO DIAGNOSTICS.