U.S. Department
of Transportation
**Federal Railroad
Administration**

Office of Research and
Development
Washington, D.C. 20590

Railroad Yard Simulation Model

Description and Computer Program User's Manual

FRA/ORD-81/25-I
October 1981
Final Report

L. E. Wittig, C. E. Waldman, and
E. K. Bender
Bolt Beranek and Newman Inc.
50 Moulton Street
Cambridge, MA 02238

NOTICE

This document is disseminated under the sponsorship of the U.S. Department of Transportation in the interest of information exchange. The United States Government assumes no liability for the contents or use thereof.

NOTICE

The United States Government does not endorse products or manufacturers. Trade or manufacturer's names appear herein solely because they are considered essential to the object of this report.

Technical Report Documentation Page

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
FRA/ORD-81/25.I		

4. Title and Subtitle	5. Report Date
Railroad Yard Simulation Model: Description and Computer Program Users' Manual	October 1981
	6. Performing Organization Code
	8. Performing Organization Report No.
7. Author(s)	BBN Report No. 4606
L.E. Wittig, C.E. Waldman, E.K. Bender	
9. Performing Organization Name and Address	10. Work Unit No. (TRAIS)
Bolt Beranek and Newman, Inc. 10 Moulton Street Cambridge, MA 02238	11. Contract or Grant No. DOT-FR-8091
	13. Type of Report and Period Covered
12. Sponsoring Agency Name and Address	Final Report Sept. 1978 - Feb. 1981
U.S. Department of Transportation Federal Railroad Administration Office of Research and Development Washington, DC 20590	14. Sponsoring Agency Code FRA/RRD-11

15. Supplementary Notes

16. Abstract

This report is part of a larger study to identify potentially cost-effective advanced braking and coupling systems. The report describes a model for determining the cost savings in railroad yards that would result from the implementation of advanced braking and coupling systems. First, the operations of a hypothetical composite yard are explained in terms of logic diagrams. These diagrams are reduced to a set of equations, which, in turn, are incorporated into an interactive computer program. Flow diagrams for the program are included. Several example cases are presented that explain how the user determines which variables should be changed, show how he enters this information into the program, and finally, show how he executes the program.

This study has also resulted in the following reports; "Railroad Financial Evaluation Model: Description and Computer Program Users' Manual" (FRA/ORD-81/25.II); "Methodology for Evaluating the Cost and Benefit of Advanced Braking and Coupling Systems (FRA/ORD-79/57); and "Evaluation of the Costs and Benefits of Advanced Braking and Coupling Systems" (FRA/ORD-80/49).

17. Key Words	18. Distribution Statement
Braking and Coupling Systems Cost and Benefit Analysis Freight Cars Computer Program Users' Manual Railroad Yard Model	Document is available through the National Technical Information Service, Springfield, VA 22161

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of Pages	22. Price
Unclassified	Unclassified	90	

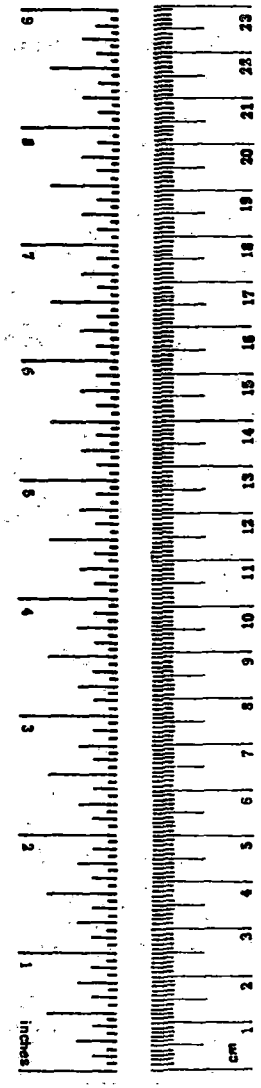
Form DOT F 1700.7 (8-72) Reproduction of completed page authorized

METRIC CONVERSION FACTORS

Approximate Conversions to Metric Measures

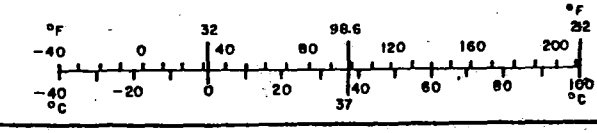
Symbol	When You Know	Multiply by	To Find	Symbol
	LENGTH			
in	inches	*2.5	centimeters	cm
ft	feet	30	centimeters	cm
yd	yards	0.9	meters	m
mi	miles	1.6	kilometers	km
	AREA			
in^2	square inches	6.5	square centimeters	cm^2
ft^2	square feet	0.09	square meters	m^2
yd^2	square yards	0.8	square meters	m^2
mi^2	square miles	2.6	square kilometers	km^2
	acres	0.4	hectares	ha
	MASS (weight)			
oz	ounces	28	grams	g
lb	pounds	0.45	kilograms	kg
	short tons (2000 lb)	0.9	tonnes	t
	VOLUME			
tsp	teaspoons	5	milliliters	ml
Tbsp	tablespoons	15	milliliters	ml
fl oz	fluid ounces	30	milliliters	ml
c	cups	0.24	liters	l
pt	pints	0.47	liters	l
qt	quarts	0.95	liters	l
gal	gallons	3.8	liters	l
ft^3	cubic feet	0.03	cubic meters	m^3
yd^3	cubic yards	0.76	cubic meters	m^3
	TEMPERATURE (exact)			
°F	Fahrenheit temperature	5/9 (after subtracting 32)	Celsius temperature	°C

*1 in = 2.54 (exactly). For other exact conversions and more detailed tables, see NBS Misc. Publ. 286, Units of Weights and Measures, Price $2.25, SD Catalog No. C13.10:286.



Approximate Conversions from Metric Measures

Symbol	When You Know	Multiply by	To Find	Symbol
	LENGTH			
mm	millimeters	0.04	inches	in
cm	centimeters	0.4	inches	in
m	meters	3.3	feet	ft
m	meters	1.1	yards	yd
km	kilometers	0.6	miles	mi
	AREA			
cm^2	square centimeters	0.16	square inches	in^2
m^2	square meters	1.2	square yards	yd^2
km^2	square kilometers	0.4	square miles	mi^2
ha	hectares (10,000 m^2)	2.5	acres	
	MASS (weight)			
g	grams	0.035	ounces	oz
kg	kilograms	2.2	pounds	lb
t	tonnes (1000 kg)	1.1	short tons	
	VOLUME			
ml	milliliters	0.03	fluid ounces	fl oz
l	liters	2.1	pints	pt
l	liters	1.06	quarts	qt
l	liters	0.26	gallons	gal
m^3	cubic meters	35	cubic feet	ft^3
m^3	cubic meters	1.3	cubic yards	yd^3
	TEMPERATURE (exact)			
°C	Celsius temperature	9/5 (then add 32)	Fahrenheit temperature	°F



ACKNOWLEDGMENT

The authors express their appreciation to the people and organizations that have helped considerably throughout this project. The FRA COTRs, Ms. Marilynne Jacobs and subsequently Dr. N. Thomas Tsai, have provided invaluable guidance and direction. In addition, an industry committee composed of Messrs. Geoffrey Cope of Dresser Industries, John Punwani of the Association of American Railroads, Bruce Shute of the New York Air Brake Co., Donald Whitney of the Burlington Northern Railroad, and Carl Wright of Westinghouse Air Brake Co. have performed important review and consultation. The American railroad industry, in particular the Southern Railway, Boston and Maine, Conrail, and several other railroads, has graciously provided information and an opportunity to observe railroad operations.

The authors would like to acknowledge the work of John Cohen, who wrote the original computer code; William Cote, who revised the code; and Deborah Melone, who edited the report.

TABLE OF CONTENTS

1. GENERAL INFORMATION

1.1 Summary

This report describes RAIL, a computer model for railroad classification yards that follows cars as they flow through the yards and keeps track of certain time and cost elements associated with processing these cars. This model was developed and used as part of a cost/benefit study of advanced braking and coupling systems sponsored by the Federal Railroad Administration (FRA). The model's major emphasis is on those tasks that involve braking and coupling systems, but other tasks, such as moving the cars, are included to give the model more structure. The RAIL model does not keep track of the time or cost for processing paperwork associated with moving cars through classification yards.

The purpose of this program is to assess the benefits of implementing advanced braking and coupling systems. Railroad cars are classified (separated and reassembled into new trains) in two types of yards -- hump yards and flat yards. An example of the use of the RAIL model in hump yard classification is as follows. At present, when cars are classified in a hump yard, a man must walk the length of the train and stop at each car to bleed the air from brake cylinders and reservoirs. With an advanced braking and coupling system, it may be possible to bleed the entire train by pushing a single button in the locomotive. The RAIL model is designed to determine the benefits that would result from such advanced systems. In some cases, the total time saved may be only the time to perform the individual tasks (for example, bleeding the cars); other cases may yield additional savings (for example, the time required to walk from car to car). The total cost to perform a task is determined by multiplying the amount of time it takes to perform the task by the pay

rate of the men involved and by the hourly value of the equipment used. The benefit of an advanced system is found by taking the difference between the present cost of processing cars through a yard and the cost of processing them with an advanced system.

In its present configuration, the RAIL model computes the benefits that would be accrued by the entire U.S. railroad industry. The model is constructed this way chiefly because the types of advanced systems under consideration require compatibility among all cars in interchange service. The model could be modified, however, to reflect more accurately the operations of a specific railroad or a specific yard.

This report describes the basic concept of the railroad yard simulation model, provides operating instructions for people who may wish to use the model as it now exists, and explains how to make minor modifications to it. Several example cases are included. A complementary financial model [1], developed as part of the same project but not discussed in this report, can be used to determine whether the benefits for a given advanced system, as computed by this model, will result in a favorable cost/benefit ratio.

The users' manual in this report follows the recommended format for computer program users' manuals shown in Federal Information Processing Standards Publication No. 38, "Guidelines for Documentation of Computer Programs and Automated Data Systems." In addition to the users' manual, the report includes an expanded section that describes the model.

The section on Operating Procedures presents several example cases demonstrating how to initiate the program, what types of commands can be made, and what the output looks like.

1.2 User Profile/Operational Environment

The railroad yard simulation model, RAIL, is an interactive computer program developed by Bolt Beranek and Newman Inc. (BBN) to run on the computer system maintained by the BBN Research Computer Center (RCC). The program is available through BBN on a time-shared system that uses Digital Equipment Corporation computers. Alternatively, tapes of the program can be obtained from BBN or NTIS and made operational on other computer systems.

After accessing the RAIL model through a keyboard computer terminal, users are prompted by instructions from the program. For example, typing the word HELP (followed by pressing the return key) will produce a brief explanation of the program, including a description of the commands that it will accept. The program is written in the FORTRAN computer language, but users do not need to know FORTRAN unless they want to make changes to the model. To run the program, users simply type brief commands on the terminal. Little typing skill is required to use the RAIL program, because all the commands and other inputs to the program are kept short.

To use the model properly, however, users must become familiar with the basic concepts of the yard model. For example, if a user wants to know the potential benefit of implementing a given advanced braking or coupling system, he must decide independently which tasks will be changed or eliminated. Sometimes this process may be relatively straightforward; sometimes, however, it may be more complicated, as when elimination of one task, such as coupling the air hoses, also eliminates a second task, such as walking from car to car.

2. DESCRIPTION OF THE RAIL MODEL

2.1 Purpose of Model

The railroad classification yard simulation model described in this report was developed as part of a larger study of the potential benefits of implementing advanced braking and coupling systems on railroad freight trains [2-4]. Many of the benefits associated with the implementation of these advanced systems involve elimination of a large number of simple tasks that are repeated daily in all railroad yards. This model keeps track of the time and money that would be saved if one or several of these tasks could be shortened or eliminated.

An example of such a benefit, cited earlier in the Summary, is the case of an advanced system that would eliminate the task of bleeding the brake cylinders and air reservoirs. This task requires a man to walk from car to car of a train on the arrival track, pulling and holding the brake release rod on each car for about 25 seconds. The task could be eliminated by an advanced system in which the train is wired so that all the cars can be bled by simply pushing a button in the locomotive.

To determine the benefit of eliminating this task by using the computer model, one sets the time to bleed a car (symbol BC in the computer program) equal to zero and runs the program. The program calculates the total time saved in one year in terms of (1) man-years for various types of crews, (2) hours of locomotives and switcher time, and (3) hours of car time. It also multiplies by the labor or equipment use rate for each of these items and gives the number of dollars saved in each category. The dollar values given are for 1979 labor and equipment use rates, and they apply to the entire U.S. railroad industry.

In the overall study of railroad braking and coupling sponsored by the FRA, this information on cost savings was next used in a financial model [1]. This second model kept track of implementation rates, inflation, taxes, and other variables to determine if it would be beneficial to invest in a given advanced system.

A more detailed description of the example task, bleeding the brakes, will illustrate why potential users must become familiar with the basic workings of the railroad classification yard simulation model (rather than just the computer code) if they are to use the model and corresponding program properly. In most hump yards, the brake bleeding task is performed in conjunction with an inbound inspection. Because these tasks are combined, even if the bleeding task is eliminated, the walk from car to car is still necessary. In addition, when a train is about to leave a yard, a man must also walk from car to car to couple the air hoses. If an automatic hose coupler were introduced, both the task of coupling the hoses together and the time to walk from car to car would be eliminated. Judgments as to which tasks will be eliminated or otherwise changed by a given technology are left to the program user. Therefore, users must understand the simulation model description presented in the next section before they can use the model intelligently.

2.2 An Overview of Railroad Yard Operations

The progress of a freight car as it passes through a classification yard can usually be divided into four major operations. In some yards, the order of these events or the tasks within them may vary, but the order presented here is the most common [2,5]. The major operations are:

1. Yarding the train
2. Classifying the train
3. Pull-down operation
4. Power brake test.

The remainder of this section gives a broad overview of these operations; Sec. 2.3 presents the detailed models and discusses limited portions of the models as examples of how they work.

Yarding the train. As a train approaches a yard, the crew may have standing instructions about how the train should be yarded, or they may receive instructions by radio. The instructions tell the crew which track to leave the cars on; in some cases, they specify whether the train should be broken into two or more parts and indicate the track on which each part should be left. When it reaches the outskirts of the yard, the train may have to stop to allow a crew member to get off and throw some switches, so that the train will go to the correct arrival track.

If the train is yarded on one track, it is pulled into place and a service brake reduction is made. Following this procedure, the hand brakes on several cars may be set, if the grade of the track requires such braking. This step is not necessary, for example, in a bowl-shaped yard, because the shape prevents the cars from rolling out. Next, a crew member pulls the pin on the coupler on the lead car, and the locomotive consist pulls away. Note that this pulling away will apply the emergency air brakes on the remaining cars. In some yards, the locomotive consist may join the caboose; this procedure may require several stops to throw switches, or the caboose may be left and classified with the rest of the freight cars.

Yarding the train on more than one track may require several additional maneuvers, such as stationing a man at the location where the train is to be broken, so that he can set hand brakes and pull the coupler pin. The rest of the steps described above are then repeated for each segment of the inbound train. During all these operations, a full road crew, a road locomotive consist, and an average train length of cars are all accumulating charges.

After the locomotives have left but before the cars are classified, the inbound inspection generally takes place. Blue flags are posted at the ends of the train to warn the switcher crews not to move the cars, because an inspector may be climbing about the cars, and moving them would be dangerous. More than one inspector may work the train; two inspectors may walk together along opposite sides of the cars, or they may start at opposite ends. Cars that fail inspection are bad-ordered. Such cars have bright-colored signs posted on them so that they will be switched to a special track to be repaired.

As the cars are inspected, the brakes are bled by pulling the release rod on the brake control valve of each car. If the classification yard is a hump yard, the rod is held long enough to drain the compressed air in both the brake cylinder and the reservoirs. The reservoirs are drained because air could leak out from them and reapply the brakes, causing long delays in the hump yard procedures. In flat yards, it is more common to bleed only the brake cylinders, because in these yards, a car on which the brakes reapply would not cause a long disruption in operations.

In some large yards, inspection is done on a rollby basis just before the hump; in these cases it is still necessary first to walk the length of the train for the sole purpose of bleeding

the cars. In such yards, an automatic air bleed system would save both the time of bleeding and of walking, but in most yards, where the bleeding is part of the inbound inspection, only the time to bleed the brakes would be saved. Note that not only the inspectors but the cars (although not the locomotives) are tied up while the train is being inspected and brakes are being bled. After inspection, the blue flags are removed, and the train is ready for classification.

Classifying the train. Classification is the process by which trains are broken down and then reconstituted into blocks of cars that will subsequently be assembled into trains. This operation varies according to whether the classification yard is a hump yard or a flat yard. In a hump yard, a locomotive is moved to the back of the block of cars to be classified, and the cars are slowly pushed over a hump. As they crest the hump, a man pulls the coupler pins, and the uncoupled cars roll freely down the hump to the proper classification tracks. A man (or computer) throws the proper switches to send the car to the right track, controlling the speed by retarders.

Cars that require special handling because they are carrying fragile or dangerous cargo are not allowed to roll freely. These cars are either set out on a special track just short of the hump, where they are picked up later by a switch locomotive, or they are lowered over the hump with the remaining train still coupled. In the second case, they are shoved to the proper classification track, and the remaining train is pulled back up over the hump. In many cases, the next car added to a special car must also be "shoved to rest," and so the same procedure is followed.

In hump yard classification, several people are involved. The locomotive has an engineer and perhaps a trainman to help him

maneuver through switches. A man at the hump pulls the coupler pins, and, depending on the size of the yard and the degree of automation, one or more men control the switches and retarders. In some yards, another switch engine and crew are out on the classification tracks, pushing the cars that stop short of coupling and compacting the cars on the tracks. This process is called trimming. Several other people keep track of the paper work, supplying the various crews with information, such as the track to which each car should be switched.

In a flat yard, the goal of classification is the same as in a hump yard, but the gravity hump is replaced by the switch engine, which pushes a cut (a group of cars) up to speed and then slows down as a man pulls the coupler pin. The cutoff car (or cars) then rolls freely to the right classification track. In a flat yard the switches are generally thrown by hand; the man who throws the electronically controlled switches in a hump yard is replaced in a flat yard by one or more men on the ground. A trim crew is not generally used in a flat yard, although the crew that classified the train may trim the classification tracks occasionally if necessary.

In flat yards, more often than in hump yards, a car may be switched more than once, particularly if the train will be dropped off in blocks or if it is a local delivery train. A cut of cars will be pulled from the class track and reswitched so that cars are arranged in an order that corresponds to the way they will be dropped off. The schemes for this rearrangement can seem somewhat involved to an outsider, but they are relatively straightforward railroad procedures [5]. Reswitching is also necessary when not enough classification tracks are available and the destination of the cars on classification tracks changes during the day.

Pull-down operation. The pull-down operation is the method by which blocks of cars are moved from the classification tracks to a departure track where they are assembled as a new train. Generally, the cars on the classification tracks are not fully coupled together. Consequently, a switch engine enters the track at one end (usually the opposite end from the hump or switch lead), and a man walks the track supervising the coupling of cars. He may, for example, have to step between cars and align the couplers by hand and then signal the switch engine to move forward. The signaling may be done by hand or by radio, depending on union agreements or availability of equipment.

Usually the cars on three or four classification tracks are pulled to make up a single train on the departure track, and the switch crew may have to operate several switches along the way as well as couple and uncouple cars at the locomotive. If the classification tracks or departure tracks are on a grade, hand brakes will have to be released or set at various points.

Power brake test. Once the cars are assembled on a departure track, they are ready to be charged with air. The most common way to charge the train in a large yard is to use yard air - that is, compressed air that is available at several locations scattered about the yard. Alternatively, the train can be charged from a switch engine or from the road locomotives that will be used to move the train. When yard air is used, the most common procedure is called "walking the air." An air hose is connected to one or both ends of the train, and from that point a man walks the train coupling the air hoses and checking the angle cocks as he goes. If two men are working, one starts from each end, and the two meet in the middle, where they wait for the air pressure to rise; then they retreat, listening for air leaks. When they reach the ends again, they close off the appropriate angle cocks and disconnect the yard air. If a switch engine or

road locomotive is used, the air hoses between all cars are probably coupled before the compressed air from the locomotive is applied.

Assuming that the train was charged with yard air or by a switch engine, the next step is to add the road locomotives. This process requires several small tasks, such as connecting the air hoses and opening the angle cocks, being careful to perform each task in the right order.

After the road locomotives are added but before the train is allowed to leave the yard, it must pass the power brake test. This test can be initiated when the pressure is at least 60 psi at the rear of the train and within 15 psi of the supply pressure at the head of the train. If these criteria are not met, the crew must walk the train to find the problem. Whenever possible, the problem will be fixed immediately; however, if the problem is traced to a specific car that cannot be easily fixed, the train must be broken, the car set out, and the train reconstituted.

The above scenario is for the yard where the train originates; at the point of origin, the power brake law requires that all brakes work. If a car develops brake trouble later on, its brakes may be "cut out"; this means that the operating valve on the troubled car is bypassed and the car is allowed to remain in the train. In some cases, usually when the weather is very cold, excess leakage may be distributed along the entire train, and the criteria can only be met by shortening the train.

When the 60 psi rear end pressure is met and the gradient between the ends of the train is 15 psi or less, a service reduction of 15 psi is made. After a short wait, the leakage rate is measured in terms of reduction in pressure per minute at the locomotive. The criterion for this test is that the leakage

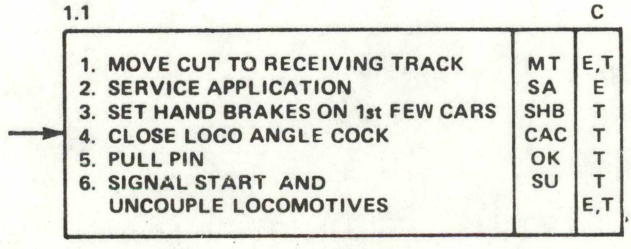
rate must be 5 psi per minute or less. Again, if the criterion is not met, the problem must be found and corrected.

If the leakage rate test is passed, a full service reduction is made and crew members walk the train and inspect it to make sure that the brakes on each car apply, the brake cylinder piston travel is within tolerance, and the brake equipment is in proper condition. Finally, the brakes are released, and the train is inspected again to make sure that all the brakes release. As with the other tests, either the problem must be fixed immediately, or the car that causes the problem must be set out. Once all parts of the power brake test are passed, the train is ready to depart.

2.3 Logic Diagrams

In this section, the movement of a car through a yard is reduced to a set of logic diagrams, and these diagrams are reduced to a set of equations that form the basis for the computer program.

The sequence of steps in car classification, described in general terms in the previous section, can be reduced to a listing of elementary tasks. For example, if a train is to be yarded on a single track, the elementary tasks are as shown in the following box, labeled 1.1 below:



Each elementary task is given a symbol, such as MT for move train and SA for a service application of the brakes. These symbols are assigned times, based on a consensus of the times measured in the AAR study [6] and those measured in the BBN study [2,3]. The column labeled C is used to keep track of the minimum number of crew members needed to perform the task. All of the tasks in the box are then reduced to a single equation for the total time they take to complete. The time to complete the items in Box 1.1, symbolized as T11, is given by the equation

$$T11 = MT + SA + HBT \times (SHB+2 \times WIC) + CAC + OK + SU. \qquad (1)$$

Generally, the total time is just the sum of the individual tasks, but in some cases the equation for total time is slightly more complicated. In the above equation, the total time to set the hand brakes is given by the expression HBT * (SHB+2*WIC), where HBT symbolizes the number of hand brakes set per train, SHB is the time to set the hand brakes on a single car, and WIC is the time it takes to walk one car length. The factor of two is used because after setting the hand brakes, the trainman must walk back to the head of the train.

The above example illustrates that in using the model, it is important to remember that judgment is required in deciding what variables to change. To determine the benefit of implementing a system that would automatically set the hand brakes by pushing a button in the locomotive, for example, one would not set SHB to zero; rather, one would set HBT to zero. In this case, this would also reduce to zero the time allowed for walking from car to car.

Often there are alternative ways to accomplish a given task. In one yard, for example, the crew may yard a train on a single arrival track, and in another yard they may use two

tracks. The model accounts for these alternatives by the method illustrated in Fig. 1.

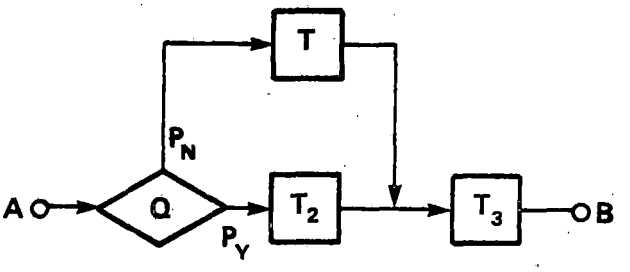


FIG. 1. EXAMPLE OF METHOD USED TO ACCOUNT FOR ALTERNATIVE WAY TO ACCOMPLISH A GIVEN TASK.

At the decision point Q a probability value, P_Y, is assigned the route corresponding to a **yes** answer to the question in the diamond-shaped box. This value is between 0 and 1. A value of $1 - P_Y$ is assigned to P_N (the route corresponding to a **no** answer). The average total time to do the work between A and B is therefore

$$T_{AB} = P_N T_1 + P_Y T_2 + T_3 \quad . \tag{2}$$

The probability values used in the computer model are based on an average of several yards visited by BBN [2,3]. In this respect the model does not represent any particular yard, but rather a hypothetical composite yard reflecting the average way of handling cars. However, the model could be made to reflect a given yard more accurately by changing the baseline values of the probabilities.

Figures 2 to 5 show the logic diagrams in all their details for the complete flow of a car through a yard. Each of the major operations (yarding the train, classification, pull-

down, and power brake test) is presented on a separate page. The equation for each box is written below it. The average time it takes to accomplish a given operation is then reduced to an equation similar to Eq. (2) above. This value is normalized by the number of cars processed per operation. For example, if one train is processed per operation, the time to perform this operation is divided by the average length of a train.

The cost assigned to a given crew, or a given type of equipment, is then determined by multiplying the pay rate by the time per car. Finally, each of these values is multiplied by the total number of cars classified per year.

To use the computer program to assess the benefit of a given system, one determines which tasks would change and makes these changes by typing them into the program. When the program is run, it then makes two calculations: first, the cost to classify all cars using present equipment and procedures, and second, the cost to classify cars with the changes. The program output is the difference between these two costs. Section 3 presents several examples of how to access the program and make changes; it also presents sample output.

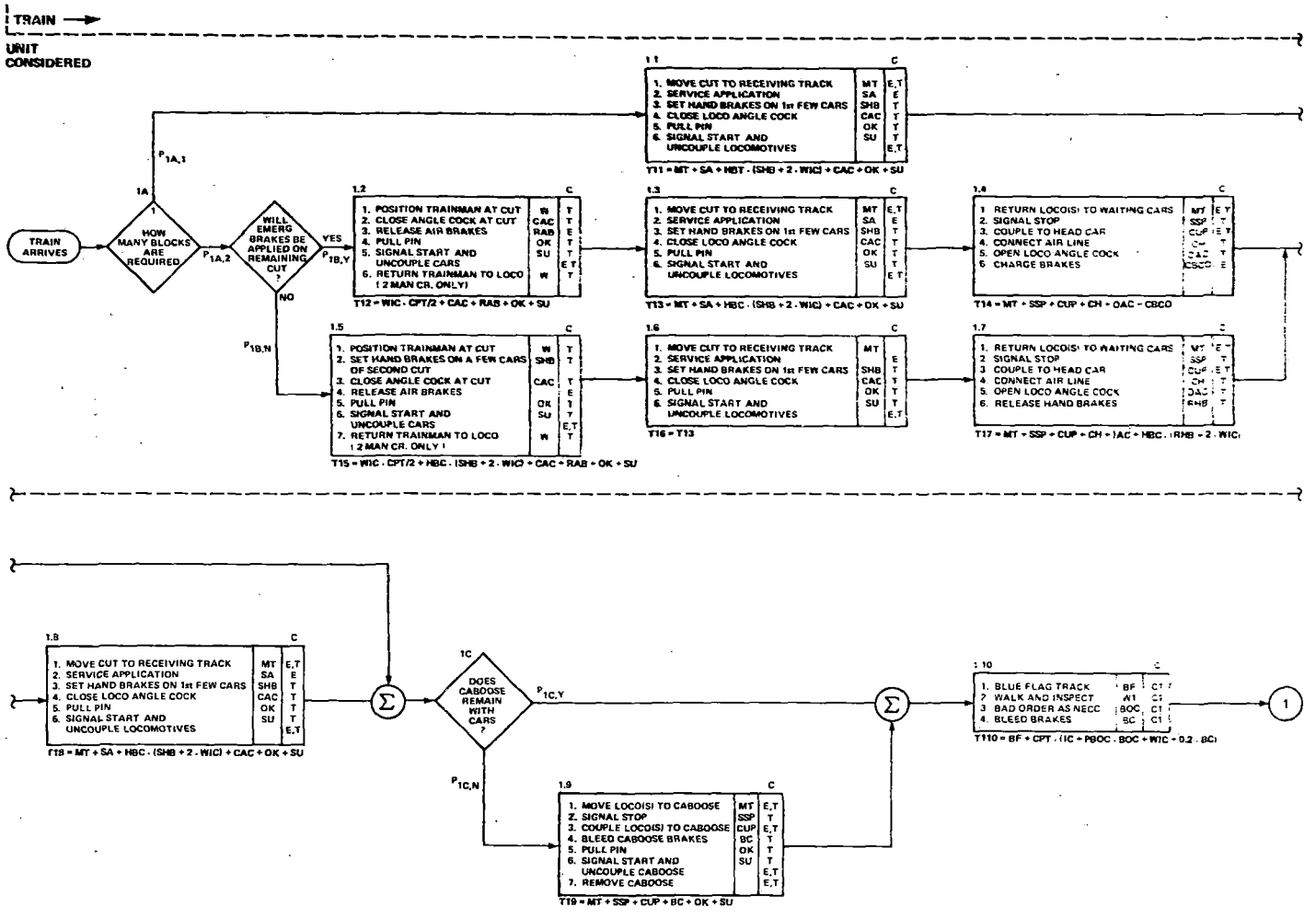


FIG. 2. YARDING THE TRAIN.

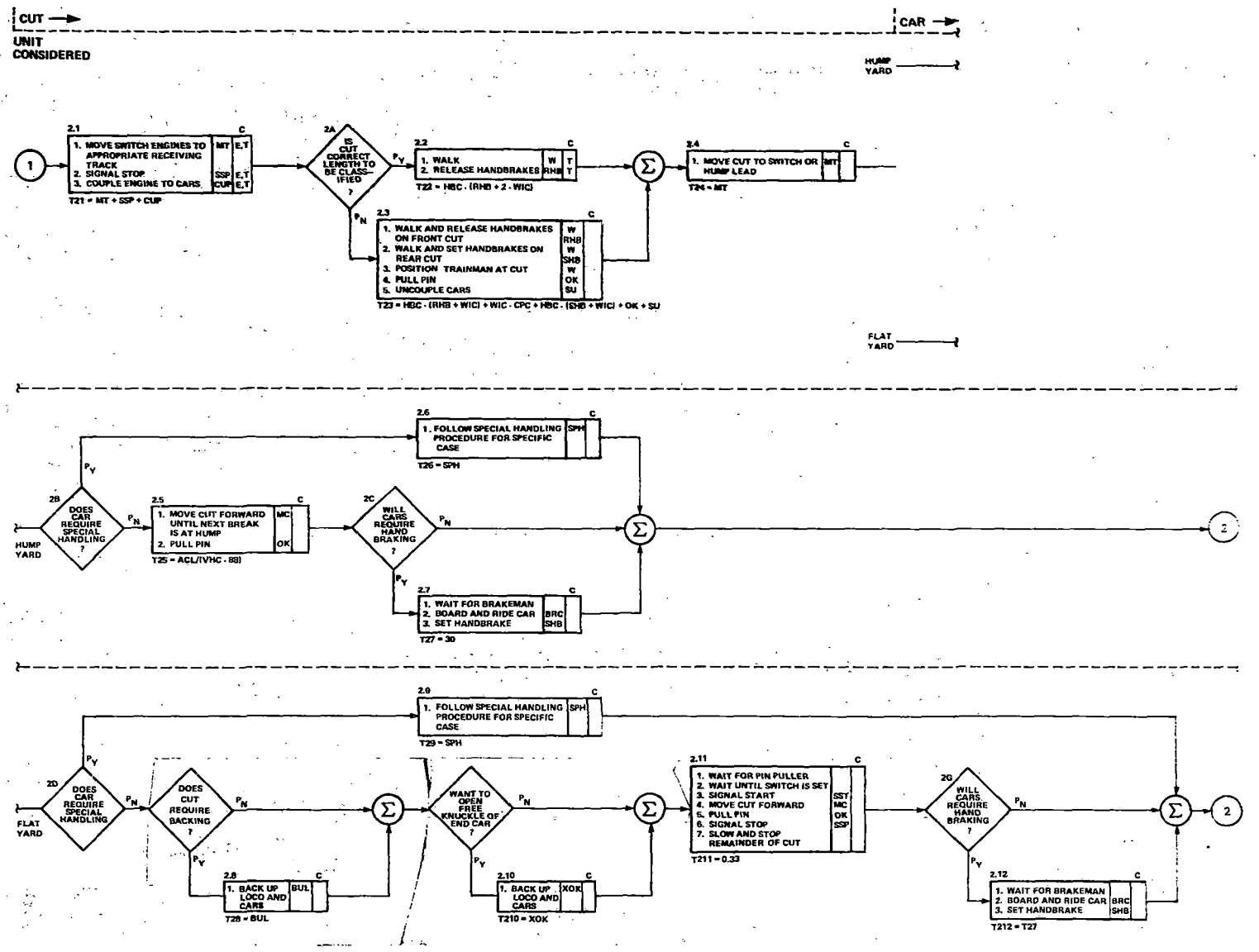


FIG. 3. CLASSIFYING THE TRAIN.

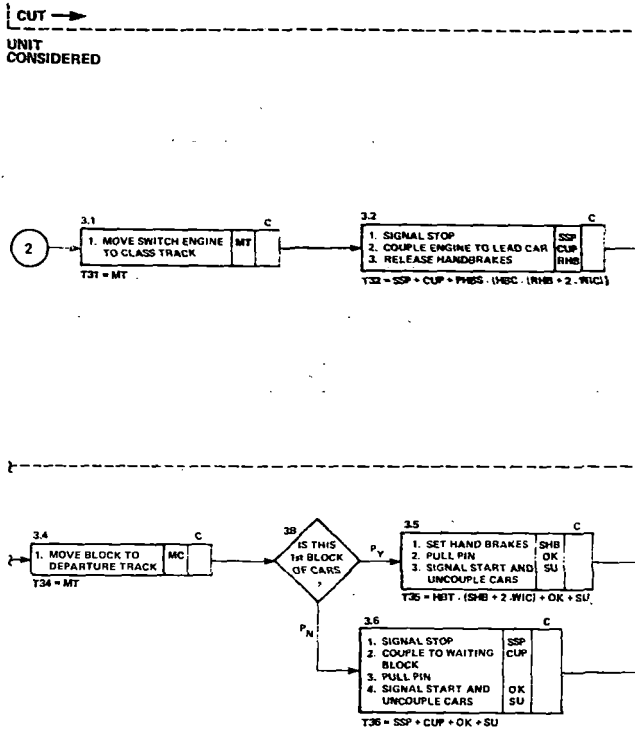
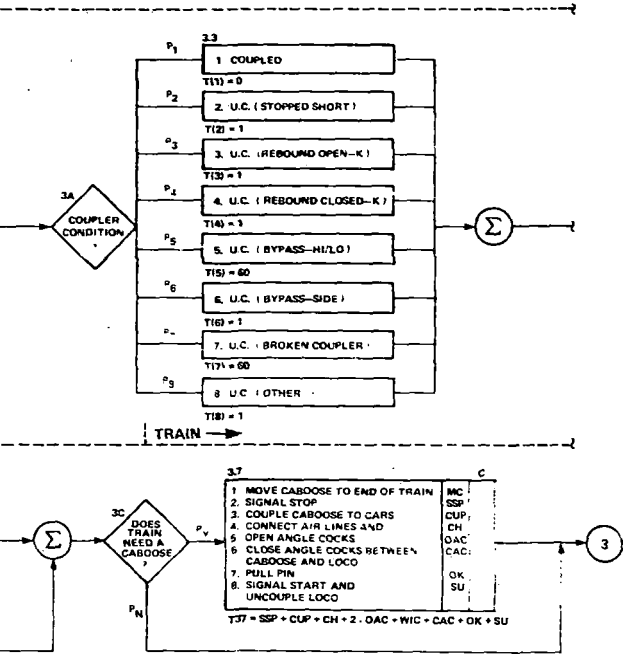


FIG. 4. PULL-DOWN OPERATION.


3.3
1 COUPLED
T(1) = 0
2. U.C. (STOPPED SHORT)
T(2) = 1
3. U.C. (REBOUND OPEN–K)
T(3) = 1
4. U.C. (REBOUND CLOSED–K)
T(4) = 1
5. U.C. (BYPASS–HI/LO)
T(5) = 60
6. U.C. (BYPASS–SIDE)
T(6) = 1
7. U.C. (BROKEN COUPLER)
T(7) = 60
8 U.C. (OTHER
T(8) = 1
3A
COUPLER CONDITION
TRAIN
3.7
C
1 MOVE CABOOSE TO END OF TRAIN
2. SIGNAL STOP
3. COUPLE CABOOSE TO CARS
4. CONNECT AIR LINES AND
5 OPEN ANGLE COCKS
6 CLOSE ANGLE COCKS BETWEEN CABOOSE AND LOCO
7. PULL PIN
8. SIGNAL START AND UNCOUPLE LOCO
MC
SSP
CUP
CH
OAC
CAC
OK
SU
T37 = SSP + CUP + CH + 2 . OAC + WIC + CAC + OK + SU
3C
DOES TRAIN NEED A CABOOSE
3

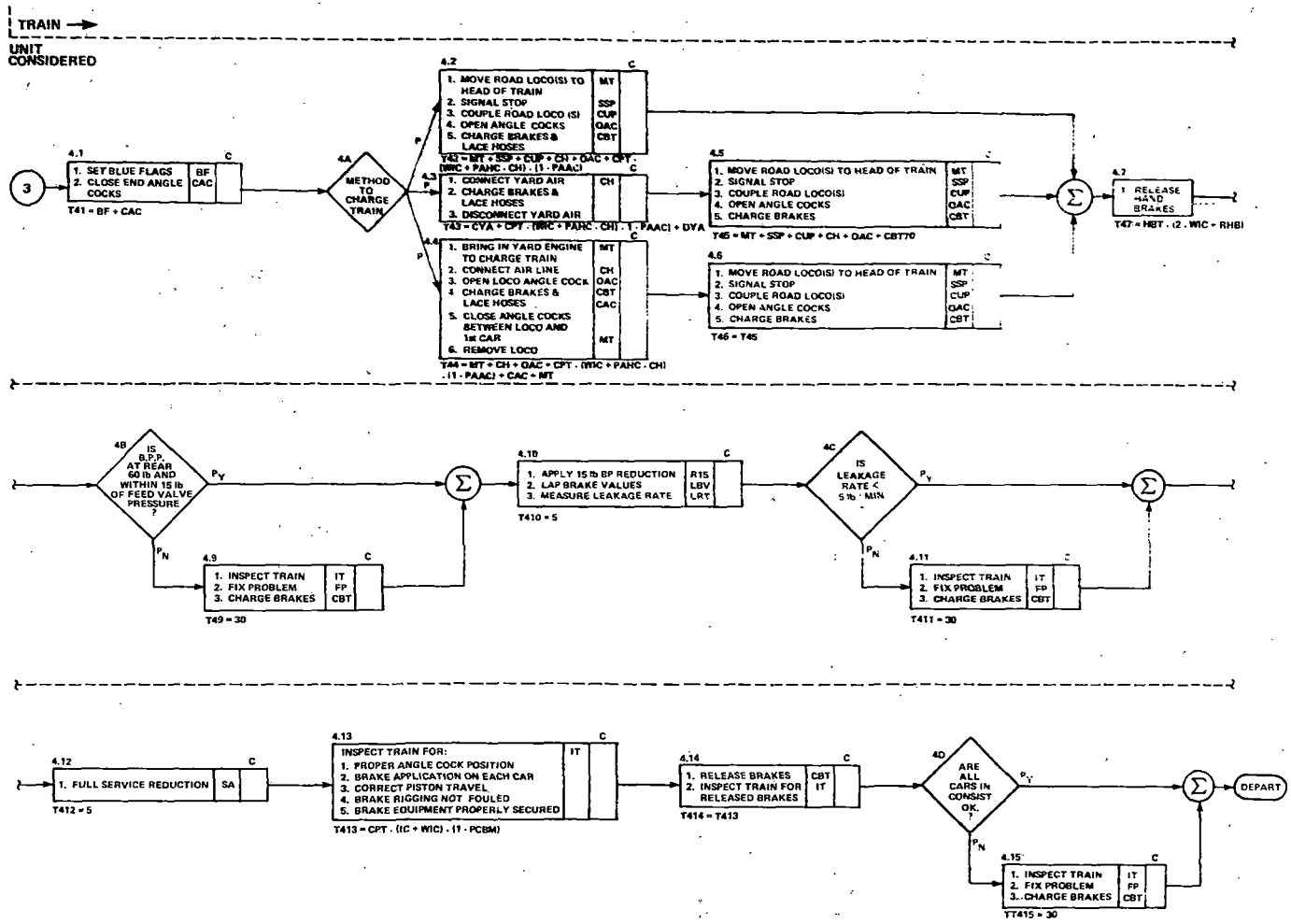


FIG. 5. POWER BRAKE TEST.

2.4 Program Structure

2.4.1 Program flow diagrams

This section presents flow diagrams for the main computer program and a few of the major subroutines. Because RAIL is an interactive program, a large portion of the program is devoted to moving files and producing the right response to a given command. The major computational work is performed in subroutines DIFF and PRGRM.

Also included in this section are

1. A table of all the subroutines with a brief explanation of their function
2. A glossary of the major variables
3. A description of the major files.

A listing of the complete program is presented in Appendix A.

MAIN PROGRAM: RAIL

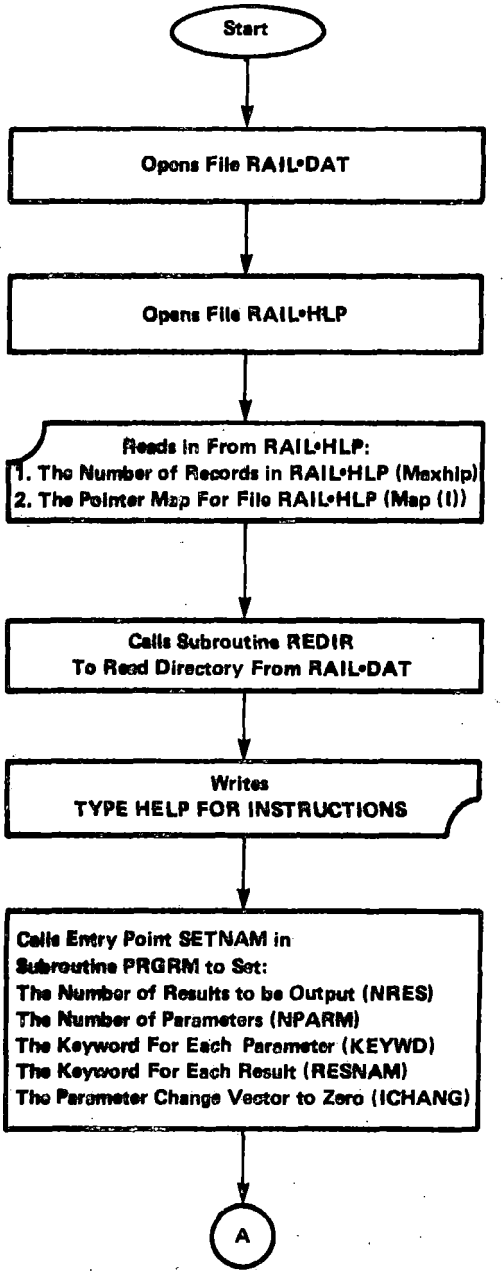


Figure 6a Flow Diagram

MAIN PROGRAM: RAIL

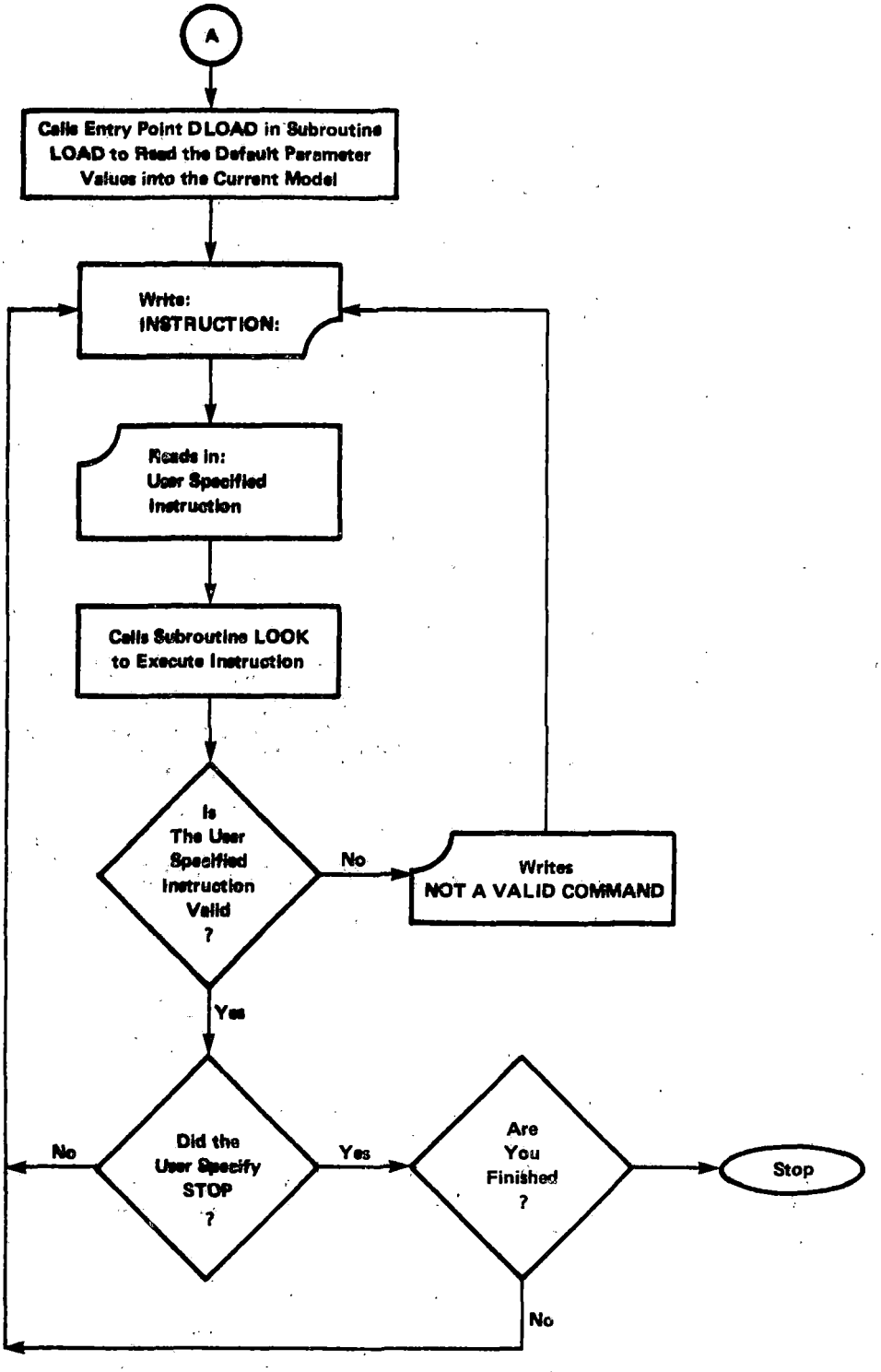


Figure 6b Flow Diagram (continued)

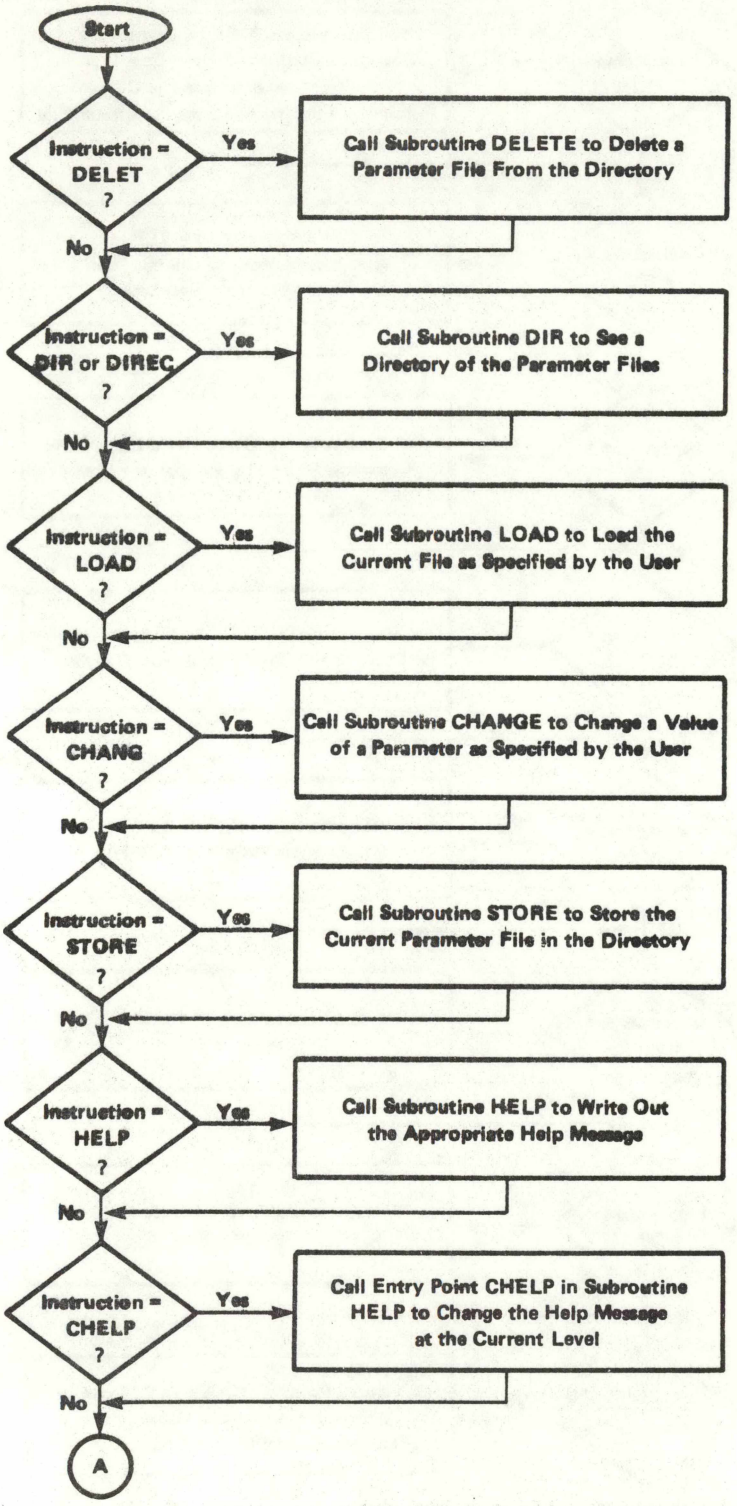


Figure 7a Flow Diagram

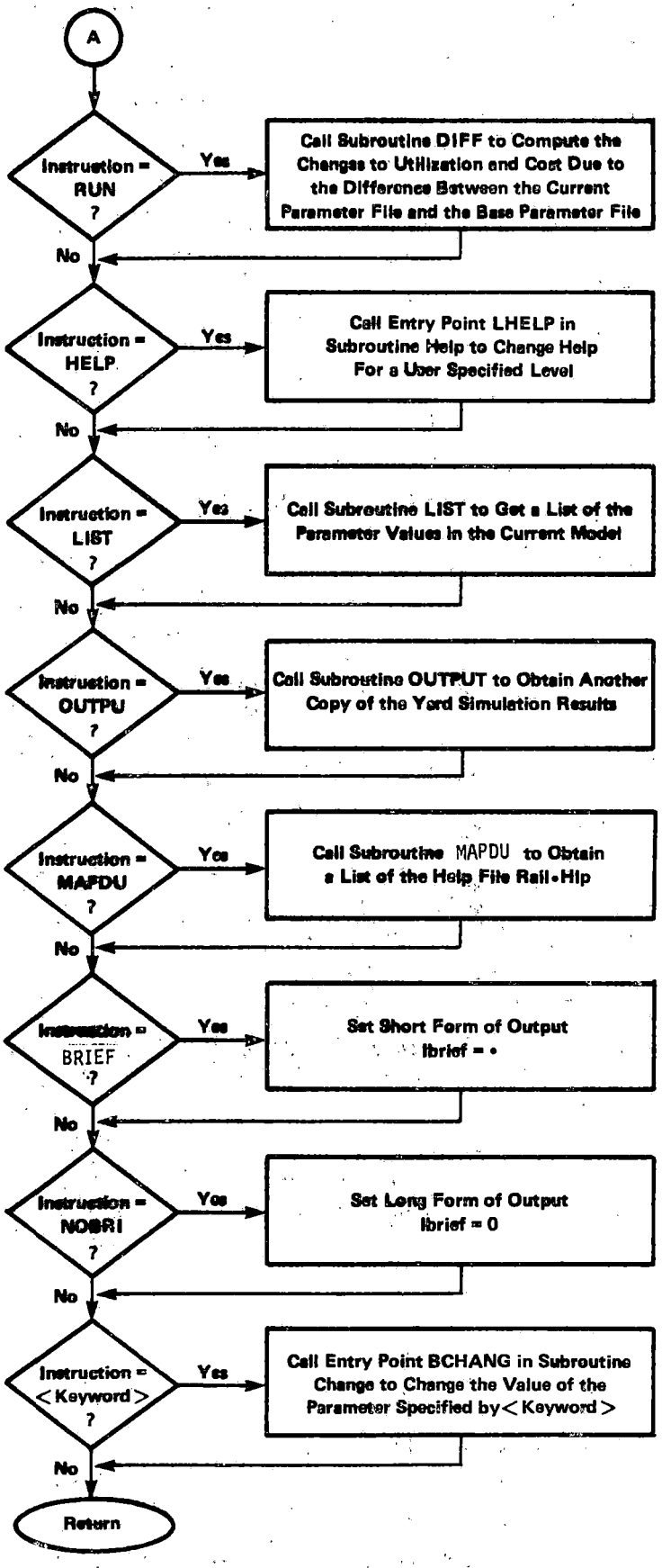


Figure 7b Flow Diagram(continued)

SUBROUTINE PROGRAM

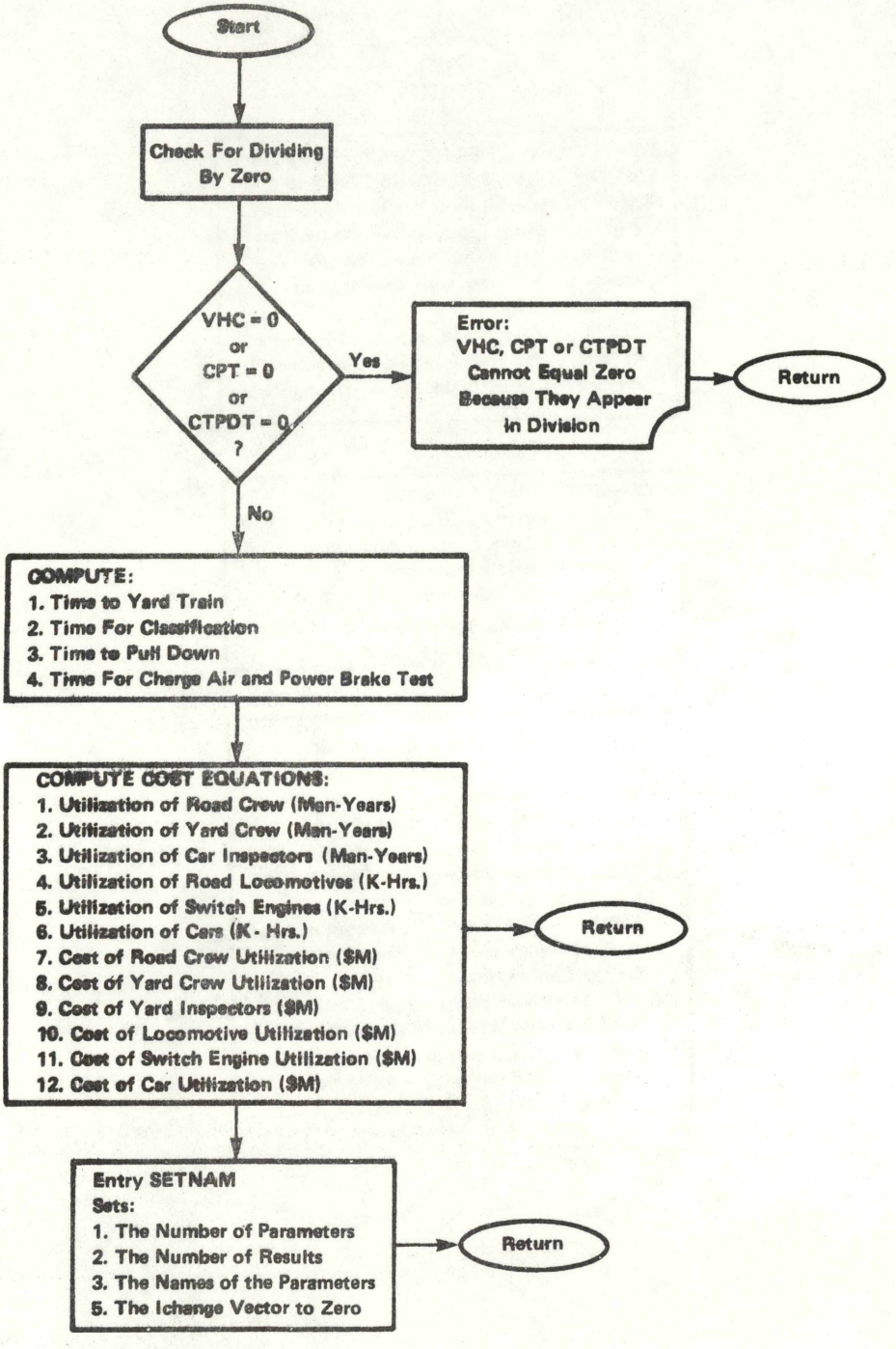


Figure 8 Flow Diagram

SUBROUTINE DIFF

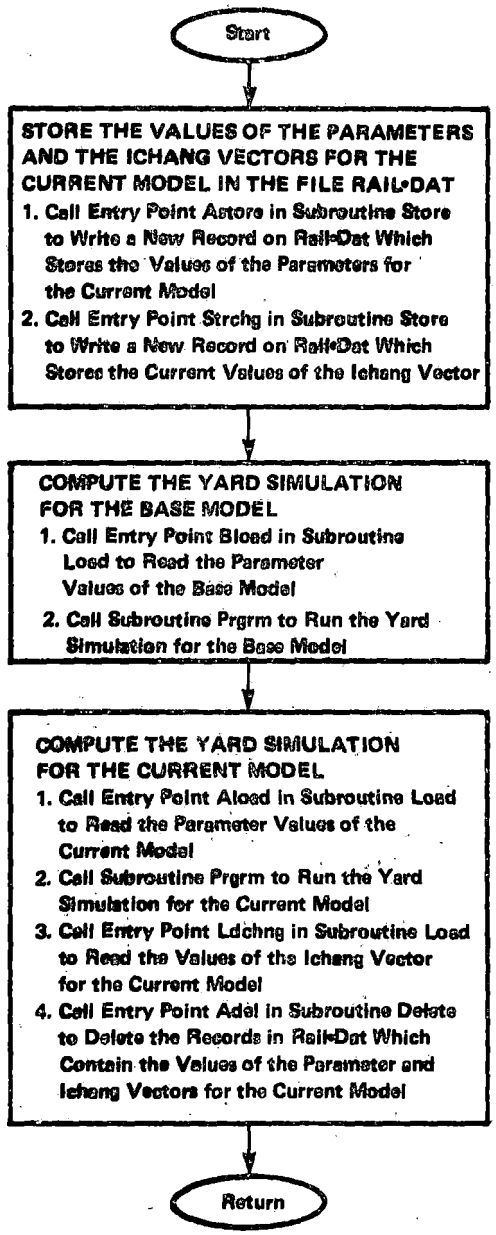


Figure 9 Flow Diagram

SUBROUTINE OUTPUT

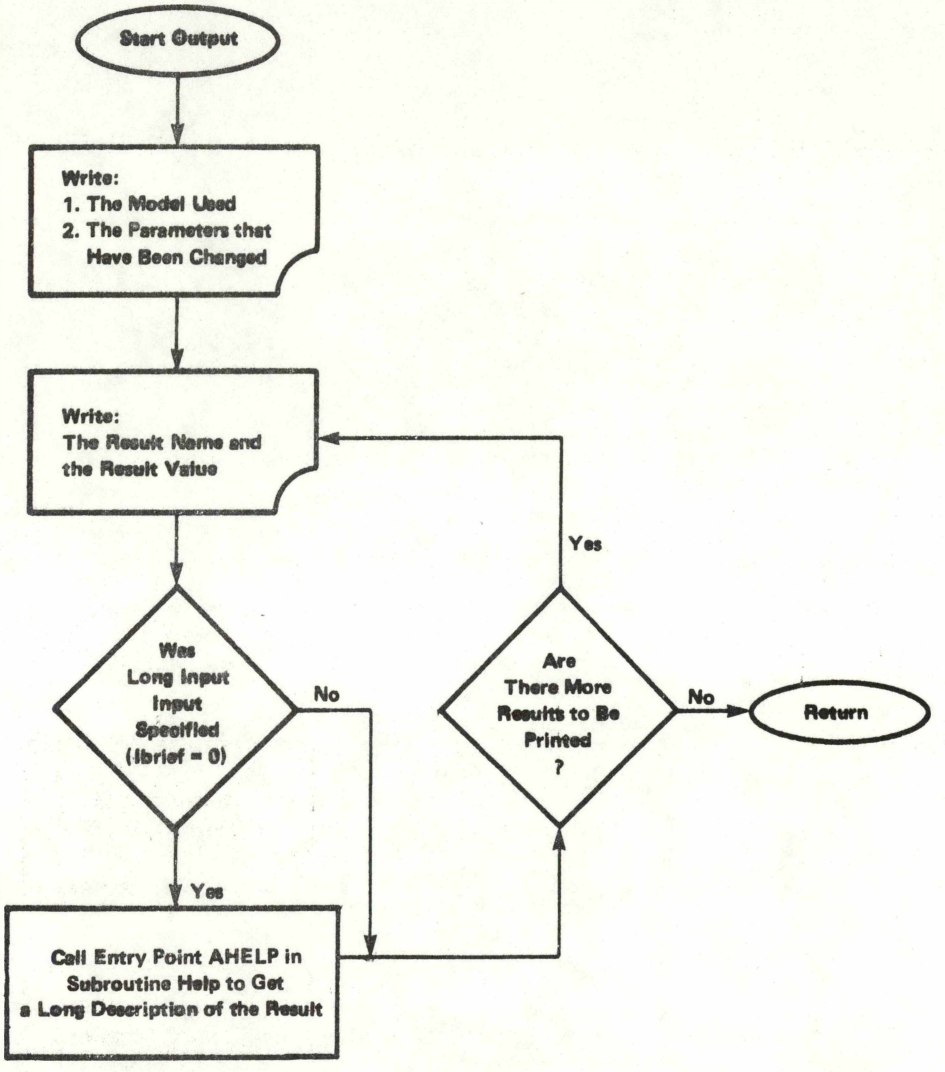


Figure 10 Flow Diagram

2.4.2 Summary of subroutines

This section presents a listing of the subroutines used by program RAIL, describes their functions, and cross-lists the subroutines that call them as well as the subroutines they call.

TABLE 1. DESCRIPTION OF SUBROUTINES

NAME OF SUBROUTINE		FUNCTION	CALLS	CALLED BY
Subroutine	LOOK	Calls subroutine specified by the user	User-specified subroutines	MAIN LOCATE CHANGE
Subroutine	HELP	Prints or changes messages		
Entries:	HELP	Prints HELP message for a specific level of the program		LOOK
	AHELP	Prints message from RAIL.HLP specified by calling program		CHANGE MAPDU OUTPUT LIST
	LHELP	Changes messages in RAIL.HLP for a user specified level		LOOK
	CHELP	Changes messages in RAIL.HLP for the current level		LOOK
Subroutine	LOAD	Retrieves values from RAIL.DAT		
Entries:	LOAD	Retrieves the values of a user-specified model	LOCATE	LOOK
	ALOAD	Retrieves the parameter values of the current model after it was saved for the yard simulation		DIFF
	BLOAD	Retrieves the parameter values of the base model		DIFF
	DLOAD	Retrieves the parameter values of the current model	ALOC	MAIN
	LDCHNG	Retrieves the stored values of the parameter change vector (ICHANG)		DIFF
Subroutine	CHANGE	Changes a parameter value of the current model		
Entries:	CHANGE	Prompts for parameter keyword then changes value of the parameter	AHELP LOOK	LOOK
	BCHANGE	Allows abbreviated form of change ('C KEYWORD')		LOOK
Subroutine	OUTPUT	Outputs the results of the yard simulation	AHELP	DIFF LOOK
Subroutine	LIST	Lists the current values of all the parameters	AHELP	LOOK

TABLE 1 *(Cont.)*

NAME OF SUBROUTINE		FUNCTION	CALLS	CALLED BY
Subroutine DIR		Prints the directory of all parameter files that are stored		
Subroutine LOCATE		Locates file in directory		
Entries:	LOCATE	Finds the location of a user specified file in the directory	LOOK	LOAD STORE
	ALOC	Finds the location of the current parameter file		DLOAD
Subroutine STORE		Stores the current parameters in RAIL.DAT		
Entries:	STORE	Stores the current parameters in RAIL.DAT under a user-specified name	LOCATE	LOOK
	ASTORE	Stores the current parameters at the end of the RAIL.DAT file		DIFR
	STRCHG	Stores the current values of ICHANG at the end of the RAIL.DAT file		DIFF
Subroutine REDIR		Reads the directory from RAIL.DAT		MAIN DELETE
Subroutine DELETE		Deletes a file from the directory		
Entries:	DELETE	Deletes a user-specified file	LOCATE REDIR	LOOK
	ADEL	Deletes the parameter values for the ICHANG vectors that were stored for the yard simulation from RAIL.DAT		DIFF
Subroutine PRGRM		Executes yard simulation		
Entries:	PRGRM	Calculates results of the yard simulation		DIFF
	SETNAM	Sets values for names of model-specific variables		MAIN
Subroutine MAPDU		Lists the contents of RAIL.HLP	AHELP	LOOK
Subroutine DIFF		Controls the yard simulation results Takes the difference between the results of the current model and the base model to calculate the final yard simulation results	ASTORE STRCHG BLOAD PRGRM ALOAD LDCHNG ADEL	LOOK

2.4.3 Data files

RAIL.DAT and RAIL.HLP are two files that contain data needed to run the program. Both of these files are random access files. Each of these data files may be either read or written into one record at a time.

The first record of RAIL.DAT contains the number of parameter sets stored on RAIL.DAT. All other records in RAIL.DAT contain the name of a parameter set, the values of each parameter, and the number of parameters in that set.

RAIL.HLP contains help messages and descriptions. Records one to eight of this file contain the number of records in the file, and the pointer map for RAIL.HLP. Each subsequent record contains a line of text, a pointer to the next line, and a flag indicating the last line of the message.

Each message is identified by a level. Messages in levels 1 to 72 contain descriptions of the parameters. Messages in levels 81 to 92 contain a description of the outputs. Messages in levels 101 to 107 contain help messages.

2.4.4. Glossary of terms

The following table contains a glossary of the major variables used in the RAIL program.

TABLE 2. Glossary

VARIABLE	DESCRIPTION	FORMAT	DEFAULT
IBRIEF	IBRIEF = 0 if long form of output is desired IBRIEF = 1 if short form of output is desired	I3	0
NRES	Number of results to be output	I2	12
LEVEL	Used by subroutine HELP. Tells subroutine HELP which message to type	I3	101
NFILE	Number of parameter files stored in RAIL.DAT	I3	1
NPARM	Number of parameters	I3	58
IMODEL(4)	Name of the model which was last loaded into the program	4A5	DEFAULT
PARAM(80)	Current values of the parameters	80F14.8	DEFAULT Parameters in file
RESULT(20)	The most recent results of subroutine Prgrm	20F13.2	
ICHANG(80)	Identifies when a parameter value has been changed ICHANG(I)=0 if corresponding entry in PARAM (PARAM(I)) has not been changed since the last load operation ICHANG(I)=1 if corresponding entry in PARAM (PARAM(I)) has been changed since the last load	80F1	0
KEYWD(80)	5 character identifier for each parameter	80A5	Specified in SETNAM
RESNAM(20)	Identifier for the results	20A5	Specified in SETNAM
MAP(200)	Pointer to file RAIL.HLP	200A3	Read from RAIL.HLP
MAXHLP	Number of records in RAIL.HLP	I3	Read from RAIL.HLP
IDIR(4,20)	Copy of directory of the parameter files	20(4A5)	Read from RAIL.HLP

3. PROGRAM OPERATION

This section introduces users to the actual operation of the model on the BBN computer system. It is meant to provide users with specific information required to operate the model. It also presents sample runs of the program.

3.1 Initiating the Program

We assume that if you plan to run the program on the BBN system you will have established an account with the Research Computer Center (RCC) and that you are therefore authorized to use one of the RCC's systems. To establish a new account, call Tony Calleva at (617) 497-3484. He will explain the procedure to you. After you open your account, you will be able to transfer the RAIL program from BENDER'S directory to your directory. To inquire about this procedure, call the contact person at BBN, Bill Cote, at (617) 497-3719.

This section presents several sample runs of the classification yard simulation model. These examples are taken in part from cases run for the FRA cost/benefit study of advanced braking and coupling systems [2,3]. The first example includes a more detailed listing than would normally be necessary, in order to illustrate some of the information that the program provides to assist users. The later examples show how the program can be run more efficiently, with fewer commands, when the user requires less prompting information.

The first step in running the program is to log in to the computer. A brief summary of how to log into the BBN computer center is presented here. Detailed instructions on how to connect to your local Terminal Interface Message Processor (TIP) are included in the Terminal Interface Message Processor User's Guide [7].

If you are using a hard copy terminal with an acoustic coupler, first turn the terminal on and then dial the appropriate telephone number for your system. Upon receiving the high beep tone, insert the receiver into the cradle. Next, press the **Carriage Return** key <CR> twice. The computer will respond by typing a herald similar to

```
NCC TIP 424 #: 7
```

Next, type the @ sign followed by the word OPEN and a space. (Note that in this case **you** must press the shift key and then type the @ sign to get the right symbol.) Then type the number 241 followed by a carriage return. The computer will show

```
@OPEN 241
```

These commands will connect you directly to BBN SYSTEM C. Then the computer will respond by typing

```
TRYING...
OPEN
```

followed, usually within a few seconds, by a response similar to

```
BBN-TENEX 1.34.56, BBN-SYSTEM-C EXEC 1.54.66
```

Having reached System C, you are now ready to log in. The computer will print the @ sign to prompt you to begin. In this example, we will log in under the name BENDER. After the @ sign appears, type

```
@LOG BENDER  <PASSWORD>  55555  <CR>
```

Be sure to press the spacebar between each command you enter. The computer will not echo the password, so the actual response will look like

```
@LOG BENDER  55555
```

(The password does not appear, but the spaces before and after it do.)

The number shown in the above line is the account number. If you open a new account with BBN, your account number will be different from the one shown here.

At this point, the computer will respond with some standard information that will look something like this:

```
JOB 28 ON TTY26 7-Feb-81 12:17
PREVIOUS LOGIN: 5-Feb-81 14:59
```

To use the yard simulation model, after the @ prompt, type RAIL, followed by a carriage return. The computer response will appear as:

```
@RAIL
```

The computer will then respond with

```
TYPE HELP FOR INSTRUCTIONS.
INSTRUCTION:
```

You have now reached the model. From this point on, the program will respond as shown in the example.

If you make a mistake while typing in a command, you can delete one letter at a time by pressing the DEL (or rub out) key.

3.2 Example Cases

Since this is the first example of the RAIL program in this manual, we will provide a fairly complete listing here; Example 2 will present a shorter version of the model for the same braking and coupling device. By comparing these examples, you can see how to run the program more efficiently once you are more familiar with it. Example 3 presents a more complex situation, where several parameters are changed, and the cumulative effect of changes is demonstrated.

3.2.1 Example 1: Knuckle-open device (long version)

A knuckle-open device is one of the components investigated in the FRA cost/benefit study as part of a coupler system that would increase gathering range. The knuckle-open device is an invention that keeps the coupler knuckle sprung open unless it is joined to another coupler. This device increases the gathering range of couplers, because both couplers are always open, whereas the most common situation during coupling operations at present is that one coupler is open and one is closed.

For the purpose of our analysis, we assumed that the increased gathering range provided by the knuckle-open device alone would not be sufficient to eliminate the task in which a man walks the classification tracks and supervises the coupling of cars during the pull-down operation. However, we did assume

that this device would eliminate the "crossover open knuckle" task performed in flat yards. Therefore, in running the model, we set the symbol for this task, XOK, equal to zero.

To use the program, on the same line as the prompt INSTRUCTION: type HELP followed by a carriage return. You will get the following response:

```
INSTRUCTION:HELP
THIS PROGRAM COMPUTES THE CHANGES TO UTILIZATION AND COST DUE TO
CHANGES IN INPUT PARAMETERS. A SET OF PARAMETERS IS STORED IN A FILE
CALLED 'DEFAULT' WHICH BECOME THE CURRENT PARAMETERS UPON STARTING THE
PROGRAM. TO CHANGE A PARAMETER USE THE COMMAND CHANGE. TO RUN THE MODEL
USE THE COMMAND RUN. TO RESET THE PROGRAM TO THE DEFAULT PARAMETERS
USE THE COMMAND LOAD. NOTE THAT THE DEFAULT PARAMETERS CAN BE CHANGED BY
STORING NEW PARAMETERS IN THE FILE NAMED DEFAULT. THE BASELINE SET OF
PARAMETERS FROM WHICH DIFFERENCES WILL BE COMPUTED IS THE FILE THAT WAS
LAST LOADED. IF YOU WANT ADDITIONAL INSTRUCTIONS TYPE HELP AGAIN.
```

Since you want a full listing for this example, you would respond here by typing HELP a second time; the computer would respond with

```
INSTRUCTION:HELP
THE INSTRUCTIONS ARE:
      CHANGE          CHANGE THE VALUE OF A PARAMETER
      LIST            LIST THE CURRENT VALUES OF THE PARAMETERS
      STORE           STORE CURRENT PARAMETERS IN A FILE
      DIRECTORY       NAMES OF FILES CURRENTLY STORED
      LOAD            TO MAKE STORED VALUES THE NEW BASELINE
      RUN             COMPUTE DIFFERENCE BETWEEN CURRENT AND BASELINE
                      RESULTS FOR UTILIZATION AND COST
      OUTPUT          OBTAIN ANOTHER COPY OF RESULTS FROM RUN
      HELP            FOR MORE INFORMATION ABOUT A QUESTION BEING
                      ASKED BY THE COMPUTER
      BRIEF           SUPPRESS PRINTING OF DESCRIPTIONS OF PARAMETERS
      NOBRIEF         RETURN TO FULL OUTPUT
      STOP            TO STOP PROGRAM

INSTRUCTION:
```

If this is the first time you are running the program, or if you want to remind yourself of the names or values of the baseline parameters, type LIST followed by a carriage return.

The computer response will be

```
INSTRUCTION:LIST


CURRENT MODEL: DEFAULT


KEYWORD   VALUE

P1A1          0.7500  PROBABILITY OF YARDING A COMPLETE TRAIN:
P1BY          0.5000  PROBABILITY THAT EMERGENCY BRAKES WILL BE APPLIED:
P1CY          0.5000  PROBABILITY THAT CABOOSE REMAINS WITH CUT:
HBT          10.0000  NUMBER OF HAND BRAKES SET ON COMPLETE TRAIN:
HBC           5.0000  NUMBER OF HAND BRAKES SET ON A CUT:
CPT          67.2000  NUMBER OF CARS PER TRAIN:
LPT           3.0000  NUMBER OF ROAD LOCOMOTIVES PER TRAIN:
SHB           0.4640  TIME TO SET HAND BRAKES ON ONE CAR (MIN.):
RHB           0.1420  PROBABILITY THAT CABOOSE REMAINS WITH CUT:
WIC           0.3420  TIME TO WALK ONE CAR (MIN.):
CAC           0.2000  TIME TO CLOSE ANGLE COCK (MIN.):
OAC           0.3570  TIME TO OPEN ANGLE COCK (MIN.):
OK            0.0970  TIME TO OPEN KNUCKLE (MIN.):
CH            0.1250  TIME TO COUPLE HOSES (MIN.):
SU            0.5000  TIME TO SIGNAL LOCOMOTIVE AND UNCOUPLE
                      LOCOMOTIVE OR CARS (MIN.):
SSP           0.5000  TIME FROM SIGNAL STOP TO TRAIN STOP (MIN.):
CUP           0.5000  TIME TO COUPLE LOCOMOTIVE OR CARS (MIN.):
RAB           0.5000  TIME TO RELEASE AIR BRAKES (MIN.):
CBTO         15.0000  TIME TO CHARGE BRAKES ON UNCHARGED TRAIN (MIN.):
CBCO          7.0000  TIME TO CHARGE BRAKES ON UNCHARGED CUT (MIN.):
CBT70         2.0000  TIME TO CHARGE BRAKES ON NOMINALLY CHARGED TRAIN
                      (MIN.):
CBC70         1.0000  TIME TO CHARGE BRAKES ON NOMINALLY CHARGED CUT
                      (MIN.):
BC            0.4200  TIME TO BLEED CAR (MIN.):
MT           14.0000  TIME TO MOVE TRAIN, CUT, OR LOCOMOTIVE (MIN.):
SA            3.0000  TIME TO STOP TRAIN WITH SERVICE APLLICATION (MIN):
BF           10.0000  TIME TO BLUE FLAG TRAIN:
P2BY          0.1000  PROBABILITY THAT CAR REQUIRES SPECIAL HANDLING:
P2CY          0.0100  PROBABILITY THAT HUMPED CAR REQUIRES HAND BRAKING:
HBCC          5.0000  NO. OF HAND BRAKES SET ON A CLASSIFICATION CUT:
IC            0.1000  TIME TO INSPECT ONE CAR:
ACL          50.0000  AVERAGE CAR LENGTH (FT.):
VHC           3.0000  SPEED OF CUT DURING HUMPING (MPH):
P3A-1         0.8500  PROBABILITY THAT CARS COUPLED:
P3A-2         0.1000  PROBABILITY THAT A CAR STOPPED SHORT:
P3A-3         0.0122  PROBABILITY THAT THE CAR REBOUNDED WITHOUT DAMAGE:
P3A-4         0.0122  PROBABILITY OF A BYPASS WITHOUT DAMAGE:
P3A-5         0.0002  PROBABILITY OF A BYPASS WITH DAMAGE:
P3A-6         0.0122  PROBABILITY OF A NO LOCK DROP:
P3A-7         0.0010  PROBABILITY OF BROKEN COUPLER:
P3A-8         0.0122  PROBABILITY OF OTHER MISCOUPLING:
```

PBOC	0.0200	PROBABILITY THAT AN INBOUND CAR IS BAD ORDERED:
P3BY	0.2500	PROBABILTY THAT BLOCK IS FIRST TO BE PULLED DOWN:
P3CY	1.0000	PROBABILITY THAT TRAIN NEEDS A CABOOSE:
P4ARL	0.1000	PROBABILITY ROAD LOCOMOTIVE WILL CHARGE BRAKES:
P4AY	0.8000	PROBABILITY THAT YARD AIR WILL CHARGE TRAIN:
P4AYE	0.1000	PROBABILITY THAT A YARD ENGINE WILL CHARGE TRAIN:
CTPDT	5.0000	NUMBER OF CUTS PER DEPARTING TRAIN:
P4BY	0.9000	PROBABILITY THAT BPP AT REAR IS GREATER THAN 60LBS AND WITHIN 15 LBS OF FEED VALVE PRESSURE:
P4CY	0.8000	PROBABILITY THAT LEAKAGE RATE IS LESS THAN 5 PSI:
P4DY	0.9000	PROBABILITY THAT ALL CARS IN CONSIST ARE O.K.:
RCSZ	4.0000	ROAD CREW SIZE (MEN PER TRAIN):
YCSZ	4.0000	YARD CREW SIZE (MEN PER ENGINE):
CI	2.0000	CAR INSPECTORS PER TRAIN:
RLPT	3.0000	NUMBER OF ROAD LOCOMOTIVES PER TRAIN:
RLAB	29452.0000	LABOR RATE (DOLLARS PER YEAR):
RRLOC	24.0200	ROAD LOCOMOTIVE RATE (INCLUDING FUEL AND MAINTENANCE COST):
RSE	21.0200	SWITCH ENGINE RATE(INCLUDING FUEL AND MAINTENANCE COST:
RC	0.9000	CAR RATE (DOLLARS PER HOUR):
BOC	2.0000	TIME TO BAD ORDER ONE CAR:
SPH	10.0000	TIME TO SPECIAL HANDLE ONE CAR:
P2AYH	1.0000	PROBABILITY THAT A CUT IS THE CORRECT LENGTH TO BE CLASSIFIED IN A HUMP YARD:
P2AYF	0.2500	PROBABILITY THAT A CUT IS THE CORRECT LENGTH TO BE CLASSIFIED IN A FLAT YARD:
CTPTH	1.0000	CUTS PER TRAIN IN A HUMP YARD:
CTPTF	4.0000	CUTS PER TRAIN IN A FLAT YARD:
BUL	0.2500	TIME TO BACK UP LOCOMOTIVE DURING FLAT SWITCHING:
XOK	0.2000	TIME TO CROSS-OVER AND OPEN KNUCKLE:
PHBS	0.2500	PROBABILITY THAT HANDBRAKES ARE SET ON SOME CARS ON THE CLASSIFICATION TRACKS:
P2DY	0.1000	PROBABILITY THAT A CAR REQUIRES SPECIAL HANDLING TO BE CLASSIFIED IN A FLAT YARD:
P2EY	0.5000	CUTS PER TRAIN IN A HUMP YARD:
P2FY	0.5000	TIME TO BACK UP LOCOMOTIVE DURING FLAT SWITCHING:
PAHC	0.5000	PROBABILITY THAT HANDBRAKES ARE SET ON SOME CARS ON THE CLASSIFICATION TRACKS:
CYA	10.0000	TIME TO CONNECT YARD AIR:
DYA	10.0000	TIME TO DISCONNECT YARD AIR:
PAAC	0.0000	PROBABILITY OF AN AUTOMATIC AIR LINE CONNECTOR:
PCBM	0.0000	PROBABILITY OF A CAR BRAKE MONITOR SYSTEM:
SEPT	2.0000	NUMBER OF SWITCH ENGINES PER TRAIN:
PWEPD	0.0000	PROBABILITY THAT WALKING THE CUT CAN BE ELIMINATED DURING PULL DOWN OPERATION:

INSTRUCTION:

The above output in response to the instruction LIST takes several minutes to type out. Therefore, do not ask for a list every time you run the program. (Example 2 is a case in which the list of parameters is not printed.) If this is the first time you have used the program, or if you change the baseline values, it is a good idea to run a list, date it, and keep it for reference.

We now proceed with our specific example of the knuckle-open device. We stated earlier that the only benefit expected is that the time to "cross over and open the knuckle" would be reduced to zero. In the above list, we see that the symbol for this parameter is XOK, and its present value is 0.20 minutes. To change its value to zero, simply type CHANGE followed by a carriage return. The computer response is:

```
INSTRUCTION:CHANGE
KEYWORD OF PARAMETER:
```

After the colon, type XOK followed by a carriage return; the computer responds by asking what the value of XOK should be changed to, that is,

```
KEYWORD OF PARAMETER:XOK
TIME TO CROSS-OVER AND OPEN KNUCKLE:
```

In this example, you would respond by typing in a zero character (not a letter O), followed again by a carriage return. The computer responds by asking again for instructions:

```
TIME TO CROSS-OVER AND OPEN KNUCKLE:0

INSTRUCTION:
```

If the braking and coupling system being evaluated had the potential for changing other than time to perform other tasks,

you would type in these symbols at this point. However, for the example shown here, the XOK change is the only change. You are therefore ready to run the program. At this point, simply type RUN and press the carriage return. The computer will run the program (i.e., calculate the effect of the changes) and will type out the following information on the number of hours and the actual dollars saved.

```
INSTRUCTION:RUN


THE FOLLOWING PARAMETERS HAVE BEEN CHANGED:
XOK           0.00  TIME TO CROSS-OVER AND OPEN KNUCKLE:


BASE CASE: DEFAULT
USING THESE PARAMETERS THE FOLLOWING RESULTS ARE OBTAINED.
URC           0.00     UTILIZATION OF ROAD CREW (MAN-YEARS)
UYC        -725.76     UTILIZATION OF YARD CREW (MAN-YEARS)
UCI           0.00     UTILIZATION OF CAR INSPECTORS (MAN-YEARS)
URL           0.00     UTILIZATION OF ROAD LOCOMOTIVE (K-HRS.)
USE        -725.52     UTILIZATION OF SWITCH ENGINES (K-HRS.)
UC       -24377.34     UTILIZATION OF CARS (K-HRS.)
CURC          0.00     COST OF ROAD CREW UTILIZATION ($M)
CUYC        -21.38     COST OF YARD CREW UTILIZATION ($M)
CUCI          0.00     COST OF YARD INSPECTORS ($M)
CUL           0.00     COST OF LOCOMOTIVE UTILIZATION ($M)
CUSE        -15.25     COST OF SWITCH ENGINE UTILIZATION ($M)
CUC         -21.94     COST OF CAR UTILIZATION ($M)



INSTRUCTION:
```

Note that the values for the time saved for equipment are given in thousands of hours (K-hrs), and the cost savings are in millions of dollars ($M). The negative sign indicates that a reduction in hours or costs has resulted from the changes that were made to the values of the parameters. The cost values are for 1979 dollars.

The response time needed to compute this output is just a few seconds. If you are using a time-shared system it may take longer when there are many users on the system at the same time.

If you do not want to run any additional cases, type STOP<CR>, and the computer will respond

```
INSTRUCTION:STOP
ARE YOU FINISHED WITH THE PROGRAM?
```

Type YES<CR> to answer this question. The computer then responds with STOP, and gives a closing message:

```
ARE YOU FINISHED WITH THE PROGRAM?YES
STOP

END OF EXECUTION
CPU TIME: 18.17 ELAPSED TIME: 8:54.85
EXIT.
^C
```

You then log out by typing LOGO<CR>.

If you are using the BBN computer system, a response like the following will appear:

```
@LOGO

KILLED JOB 28, USER BENDER, ACCT 55555,TTY 26, AT 2/07/81 1227
  USED 0:0:22 IN 0:9:49
```

To disconnect from the local TIP, **type in** the @ sign by pressing the shift key and the @ key simultaneously. Then type CLOSE<CR>. What you typed will appear as:

```
@CLOSE
```

The system will then type

```
CLOSED
```

showing that you are disconnected from the TIP.
Example 1 ends here.

3.2.2 Example 2: Knuckle-open device (short version)

The preceding example showed a long version of the program, run with all the helpful information a new user needs. After users become more familiar with the program, they will probably not use the help or list commands. The number of steps needed to change a parameter can be shortened from three to two. Instead of typing out CHANGE, simply type C, press the space bar, and type the keyword of the variable you wish to change, followed by a carriage return. In addition, users can shorten output by using the BRIEF command. Example 2 repeats Example 1 but runs the program as briefly as possible. In this example, we will not show the logging-in procedure. We will pick up the program where you type in RAIL. (NOTE: Remember to press the CARRIAGE RETURN key after each command you

enter. From now on, these directions will not mention that <CR> follows each command.) The computer responds with

```
@RAIL
TYPE HELP FOR INSTRUCTIONS.
INSTRUCTION:
```

From Example 1, you know the response to HELP and LIST, so you can go right to the short version of the CHANGE command. As in Example 1, set XOK equal to zero. Follow this step with the BRIEF command, which will result in a shortened form of the output when you run the program. This interaction will look like:

```
INSTRUCTION:C XOK
TIME TO CROSS-OVER AND OPEN KNUCKLE:0

INSTRUCTION:BRIEF

INSTRUCTION:
```

At this point you can type RUN, and the response will be

```
INSTRUCTION:RUN


THE FOLLOWING PARAMETERS HAVE BEEN CHANGED:
XOK             0.00


BASE CASE: DEFAULT
USING THESE PARAMETERS THE FOLLOWING RESULTS ARE OBTAINED.
URC             0.00
UYC          -725.76
UCI             0.00
URL             0.00
USE          -725.52
UC         -24377.34
CURC            0.00
CUYC          -21.38
CUCI            0.00
```

```
CUL             0.00
CUSE          -15.25
CUC           -21.94

INSTRUCTION:
```

Note that typing the BRIEF command causes the description of the output parameters to be suppressed. This description could be restored for future runs by typing NOBRIEF in response to the INSTRUCTION: prompt. When the program is in the BRIEF mode, it will also suppress the description of parameters given in response to the LIST command and print only the keywords and their values. If you were finished with the above case, you would type STOP, leave the program, and log out as in Example 1. Example 2 ends here.

3.2.3 Example 3: Incompatible couplers

In the FRA cost/benefit study, the advanced system referred to as an incompatible coupler is a coupler developed in Europe by the International Union of Railways (UIC) [3]. This coupler has a wide gathering range, is always open and ready to be coupled (when not already coupled), and automatically couples the brake line air. It also allows for an electrical connection between cars, but this feature has no value of its own unless the cars are equipped with electronic devices. The benefits that would be expected from such an incompatible coupler are as follows:

1. As in Examples 1 and 2, it would eliminate the "cross over open knuckle" task (i.e., XOK would be set equal to zero).

2. It would eliminate not only the "couple air hoses" task but also the walking associated with this task. An automatic air hose coupler was anticipated when the program was written. If you review the equations for boxes 4.2, 4.3, and 4.4 in Fig. 5, you will see that setting PAAC (probability of an automatic air hose connection) equal to one will eliminate both the couple hose task and the walking. In addition, CH (time to couple hoses) should also be set equal to zero for other cases when the hoses are coupled -- for example, when the locomotives are attached to the consist.*

3. Finally, the wide gathering range of the incompatible coupler is assumed to be large enough to eliminate the need to walk the classification tracks and supervise the coupling of cars during the pull-down operation. This device was also anticipated when the program was written. Setting PWEPD (probability that walking the train can be eliminated during pull-down) equal to one will eliminate this task.

You are now ready to run the program. This example will not show the log-in portion; it will pick up the program after the user has typed in RAIL (followed by a carriage return) and the computer has responded with the INSTRUCTION: prompt.

---

*Note that the PAHC (probability that air hoses must be connected on departure tracks) term in the above equation is not used to allow for an automatic coupler. This term allows for the fact that cars are frequently switched in pairs; consequently their hoses were not uncoupled and therefore would not have to be recoupled.

```
@RAIL
TYPE HELP FOR INSTRUCTIONS.
INSTRUCTION:
```

At this point the most efficient procedure would be to change XOK to zero, CH to zero, PAAC to one, PWEPD to one, and then to run the program. However, to illustrate another aspect of the program, we will perform these changes in two steps and run the program at the intermediate stage.

We proceed by first setting XOK and CH equal to zero, followed by the instruction to run. This procedure produces the following record:

```
INSTRUCTION:CHANGE
KEYWORD OF PARAMETER:XOK
TIME TO CROSS-OVER AND OPEN KNUCKLE:0

INSTRUCTION:CHANGE
KEYWORD OF PARAMETER:CH
TIME TO COUPLE HOSES (MIN.):0

INSTRUCTION:RUN


THE FOLLOWING PARAMETERS HAVE BEEN CHANGED:
CH            0.00  TIME TO COUPLE HOSES (MIN.):
XOK           0.00  TIME TO CROSS-OVER AND OPEN KNUCKLE:


BASE CASE: DEFAULT
USING THESE PARAMETERS THE FOLLOWING RESULTS ARE OBTAINED.
URC         -54.04     UTILIZATION OF ROAD CREW (MAN-YEARS)
UYC        -776.99     UTILIZATION OF YARD CREW (MAN-YEARS)
UCI        -162.79     UTILIZATION OF CAR INSPECTORS (MAN-YEARS)
URL         -81.05     UTILIZATION OF ROAD LOCOMOTIVE (K-HRS.)
USE        -776.74     UTILIZATION OF SWITCH ENGINES (K-HRS.)
UC       -36103.13     UTILIZATION OF CARS (K-HRS.)
CURC         -1.59     COST OF ROAD CREW UTILIZATION ($M)
CUYC        -22.88     COST OF YARD CREW UTILIZATION ($M)
CUCI         -4.79     COST OF YARD INSPECTORS ($M)
CUL          -1.95     COST OF LOCOMOTIVE UTILIZATION ($M)
CUSE        -16.33     COST OF SWITCH ENGINE UTILIZATION ($M)
CUC         -32.49     COST OF CAR UTILIZATION ($M)



INSTRUCTION:
```

If we now go on to change PAAC and PWEPD to one, followed by the instruction to run, we note that XOK and CH are still set to zero. That is, the **changes are cumulative** from one step to the next. The output would be as follows:

```
INSTRUCTION:CHANGE
KEYWORD OF PARAMETER:PAAC
PROBABILITY OF AN AUTOMATIC AIR LINE CONNECTOR:1

INSTRUCTION:CHANGE
KEYWORD OF PARAMETER:PWEPD
PROBABILITY THAT WALKING THE CUT CAN BE ELIMINATED
                      DURING PULL DOWN OPERATION:1

INSTRUCTION:RUN


THE FOLLOWING PARAMETERS HAVE BEEN CHANGED:
CH            0.00  TIME TO COUPLE HOSES (MIN.):
XOK           0.00  TIME TO CROSS-OVER AND OPEN KNUCKLE:
PAAC          1.00  PROBABILITY OF AN AUTOMATIC AIR LINE CONNECTOR:
PWEPD         1.00  PROBABILITY THAT WALKING THE CUT CAN BE ELIMINATED
                      DURING PULL DOWN OPERATION:


BASE CASE: DEFAULT
USING THESE PARAMETERS THE FOLLOWING RESULTS ARE OBTAINED.
URC        -226.55    UTILIZATION OF ROAD CREW (MAN-YEARS)
UYC       -6627.02    UTILIZATION OF YARD CREW (MAN-YEARS)
UCI       -1025.31    UTILIZATION OF CAR INSPECTORS (MAN-YEARS)
URL        -339.78    UTILIZATION OF ROAD LOCOMOTIVE (K-HRS.)
USE       -6626.01    UTILIZATION OF SWITCH ENGINES (K-HRS.)
UC      -284797.19    UTILIZATION OF CARS (K-HRS.)
CURC         -6.67    COST OF ROAD CREW UTILIZATION ($M)
CUYC       -195.18    COST OF YARD CREW UTILIZATION ($M)
CUCI        -30.20    COST OF YARD INSPECTORS ($M)
CUL          -8.16    COST OF LOCOMOTIVE UTILIZATION ($M)
CUSE       -139.28    COST OF SWITCH ENGINE UTILIZATION ($M)
CUC        -256.32    COST OF CAR UTILIZATION ($M)



INSTRUCTION:
```

If you do not want the changes to accumulate, you can do one of the following:

1. Change the individual parameters back to their default values.
2. Type in LOAD; when the program asks which file, type in DEFAULT. This command will set all parameters back to their default values.
3. Log out; when you log back in, all parameters will be set to their default values.

After the last instruction prompt, you can type STOP and proceed to log out as in Example 1, or you can go on to run additional cases.

Example 3 ends here.

3.3 The STORE, DIRECTORY, LOAD, and DELETE Commands

You may recall from Example 1 that the response to the second HELP command was a list of acceptable commands. Some of these commands (HELP, LIST, CHANGE, RUN, and STOP) were explained in that example, but others were not. This section presents a single example that explains the STORE, DIRECTORY, LOAD, and DELETE commands.

First, as the first three examples show, these commands do not have to be used in the normal course of running the program. They are used instead to create, modify, or eliminate additional files. There are a few cases where you might want to use these commands. For example:

1. The default value of a parameter may not be correct, and you may want to change it.

2. You may want to create a new or modified file of parameters that you can recall quickly.

3. You may want to delete a file that you created but no longer need.

The following example demonstrates how to use the STORE, DIRECTORY, LOAD, and DELETE commands. When you initiate program RAIL, it loads the values of the parameters stored in a file named DEFAULT into the current operating file, and it also tells the program to use the DEFAULT values as the baseline for computations performed in response to the RUN command. As Example 1 showed, when the computer responded to the command LIST, it produced a list of the parameters, their values, and a description of each parameter. If you now change one or more of the parameters, the values in the current operating file will differ from the values in the default file, which are also used as the baseline, in accordance with the changes that you made. For example, you may initiate RAIL and change the value of XOK as follows:

```
INSTRUCTION:C XOK
TIME TO CROSS-OVER AND OPEN KNUCKLE:0
```

If you wish to create a new file, use the command STORE. The computer will come back and ask for a name to give this new file. In this example we called the new file TEST.*

*Calling the new file DEFAULT (a file that already exists) would replace the values of the parameters in the old DEFAULT file by the present current values of the parameters. In this example, the only difference is the value of XOK. This method is used to modify the values in the DEFAULT file.

```
INSTRUCTION:STORE
STORE IN FILE CALLED:TEST
```

Asking for a list of the existing files by typing DIRECTORY (DIR also works) would result in the following output.

```
INSTRUCTION:DIR
 1     DEFAULT
 2     TEST
```

In this case, the output shows that the file TEST has been added to the directory.

The file TEST can be used as the new baseline by using the LOAD command. For example:

```
INSTRUCTION:LOAD
LOAD VALUES STORED IN FILE:TEST
```

The values of the parameters in the file TEST now have been loaded into the current operating file and the program is instructed to use the values in the file TEST as the baseline values. If you now change the value of a parameter, the current operating file will differ from TEST by only the value of the parameter that you changed. For example, changing the value of PWEPD to 1.0 and running the program gives

```
INSTRUCTION:C PWEPD
PROBABILITY THAT WALKING THE CUT CAN BE ELIMINATED
                        DURING PULL DOWN OPERATION:1

INSTRUCTION:RUN


THE FOLLOWING PARAMETERS HAVE BEEN CHANGED:
PWEPD          1.00  PROBABILITY THAT WALKING THE CUT CAN BE ELIMINATED
                        DURING PULL DOWN OPERATION:
BASE CASE: TEST
USING THESE PARAMETERS THE FOLLOWING RESULTS ARE OBTAINED.
URC              0.00     UTILIZATION OF ROAD CREW (MAN-YEARS)
UYC          -5677.53     UTILIZATION OF YARD CREW (MAN-YEARS)
UCI              0.00     UTILIZATION OF CAR INSPECTORS (MAN-YEARS)
URL              0.00     UTILIZATION OF ROAD LOCOMOTIVE (K-HRS.)
USE          -5676.79     UTILIZATION OF SWITCH ENGINES (K-HRS.)
UC         -190740.00     UTILIZATION OF CARS (K-HRS.)
```

```
CURC          0.00     COST OF ROAD CREW UTILIZATION ($M)
CUYC       -167.21     COST OF YARD CREW UTILIZATION ($M)
CUCI          0.00     COST OF YARD INSPECTORS ($M)
CUL           0.00     COST OF LOCOMOTIVE UTILIZATION ($M)
CUSE       -119.33     COST OF SWITCH ENGINE UTILIZATION ($M)
CUC        -171.67     COST OF CAR UTILIZATION ($M)
```

This output is the difference in utilization and costs between the values in TEST as the baseline, and the values in the current operating file (which in this example only differ from the values in TEST by the change in PWEPD from 0 to 1).

If you no longer wish to maintain a file that you created, you can delete it by typing the command DELETE. For example:

```
INSTRUCTION:DELETE
DELETE FILE NAMED:TEST
```

The result of the above instruction is that the file TEST has been eliminated. To confirm that the file was eliminated, type DIR to see the contents of the directory:

```
INSTRUCTION:DIR
 1     DEFAULT
```

As the output shows, TEST has been eliminated.

3.4 Utility Subroutines

The subroutines described in this section are for the programmer who wants to change messages stored in random access files. These messages include the descriptions of parameters that are typed out in response to the LIST command, the description of the output parameters that you get in response to the RUN or OUTPUT commands, and several messages that you get in response to the HELP command.

In general you should not have to use CHELP or LHELP unless you want to change the description of either a parameter or a help message. Once you call CHELP or LHELP in response to the INSTRUCTION prompt, the old description or help message is destroyed and replaced by the new message. You cannot edit the old message, and you must completely type in the new message exactly as you want it stored.

If you do decide to change one of the messages, you should first type MAPDU (followed by a carriage return). This will give you a listing of all the stored messages, starting with Level 1. A portion of this listing is shown below.

```
LEVEL= 86
   UTILIZATION OF CARS (K-HRS.)

LEVEL= 87
   COST OF ROAD CREW UTILIZATION (DOLLARS)

LEVEL= 88
   COST OF YARD CREW UTILIZATION ($M)
```

Now if you wanted to change one of these messages, you would type LHELP (followed by a carriage return). The program would respond with the prompt LEVEL: and you would enter the number of the message that you wanted to change. For example:

```
INSTRUCTION:LHELP
LEVEL:87
ENTER A LINE.
   COST OF ROAD CREW UTILIZATION ($M);
DO YOU WANT TO ENTER ANOTHER LINE (A CONTINUATION)?NO
```

In this example, the portion of the message that read **dollars** was changed to read **$M.*** Note that the entire message had to be retyped, even though only part of it was changed.

When you are entering a new description or message, you should follow these procedures:

1. At the end of a keyword description you should include a colon as part of the message, followed by a semicolon and a dollar sign.
2. A semicolon should be typed at the end of each line of a multi-line description or message, including the last line.

A second technique that can be used to change a help message, but not the description of a parameter, is to type the command CHELP. This command will allow you to change the help message corresponding to the assigned level number at the time when you called CHELP. For example, if you entered the DELETE command, you would be at level 105. If, instead of entering the name of a file you wanted to delete, you entered HELP, the computer would respond with the following help message:

> **UP TO TWENTY FILES MAY BE STORED AT ONE TIME. IF A FILE IS NOT NEEDED IT SHOULD BE REMOVED BY TYPING FILENAME AS DISPLAYED BY THE COMMAND DIRECTORY. IF YOU DO NOT WANT TO DELETE A FILE TYPE STOP.**

If you wanted to change this message, you would type CHELP and then begin to enter the new message. Again, as in the above example, the old message would be destroyed after you typed CHELP<CR>.

---

*The above operation only changed the message; a corresponding scaling of the computed values was made at the appropriate equation in the Fortran program.

In the above example on the use of CHELP, it is not necessary to give the HELP command in the step preceding the CHELP command, but using the HELP command is a good way to view the contents of the message that you are planning to change. The advantage of using CHELP instead of LHELP is that you avoid producing the listing from MAPDU, which is time-consuming to print out. Of course, if you already had a copy of the MAPDU listing, you would not have to get a new one each time you wanted to use LHELP.

REFERENCES

1. J.A. Kane and C.E. Waldman, "Railroad Financial Evaluation Model: Description and Computer Program Users' Manual," U.S. Department of Transportation, Federal Railroad Administration, Report No. FRA/ORD-81/25.II, March 1981.

2. E.K. Bender, A.J. Berger, J.W. Ernest, and L.E. Wittig, "Methodology for Evaluating the Cost and Benefit of Advanced Braking and Coupling Systems," U.S. Department of Transportation, Federal Railroad Administration, Report No. FRA/ORD-79/57, November 1979.

3. E.K. Bender, L.E. Wittig, and H.A. Wright, "Evaluation of the Costs and Benefits of Advanced Braking and Coupling Systems," U.S. Department of Transportation, Federal Railroad Administration, Report No. FRA/ORD-80/49, October 1980.

4. E.K. Bender, L.E. Wittig, and H.A. Wright, "Recommendations for Research and Development on Advanced Braking and Coupling Systems," U.S. Department of Transportation, Federal Railroad Administration, January 1981.

5. S.J. Petracek, A.E. Moon, R.L. Kiang, and M.W. Siddiqee, "Railroad Classification Yard Technology - A Summary and Assessment," U.S. Department of Transportation, Federal Railroad Administration, Report No. FRA/ORD-76/304, January 1977.

6. S.K. Punwani and L. Eshelman, "Advanced Coupling Concepts Project - Phase 1-1/2 Report Including General Economic Model," Association of American Railroads, Technical Center, Report No. R-285, July 1977.

7. W.R. Crowther, D.C. Walden, and J. Malman, "Terminal Interface Message Processor User's Guide," BBN Report No. 2183, December 1971, revised July 1977.

APPENDIX A
LISTING OF FORTRAN PROGRAM

```
      PROGRAM RAIL
C
C     IBRIEF: 0 NORMALLY, 1 IF SHORT FORM OF OUTPUT DESIRED
C     MODBAS: NAME OF FILE FOR BASELINE  (4A5)
C     NRES: NUMBER OF RESULTS TO BE OUTPUT
C     MATCH: USE VARIES THROUGHOUT PROGRAM, ROUGHLY 1 IF NO ERROR
C            0 IF ERROR ENCOUNTERED
C     LEVEL: FOR USE BY SUBROUTINE HELP. TELLS HELP WHICH MESSAGE TO
C            TYPE.
C     NFILE:  NUMBER OF FILES IN RAIL.DAT (UNIT 21)
C     NPARM:  NUMBER OF PARAMETERS
C     IMODEL: NAME OF CURRENT MODEL, 4A5
C     IDIR:   COPY OF DIRECTORY
C     PARAM:  CURRENT VALUES OF PARAMETERS
C     RESULT: MOST RECENT RESULTS OF SUBROUTINE PROGRM
C     ICHANG: 0 IF CORRESPONDING ENTRY IN PARAM HAS NOT BEEN CHANGED
C             SINCE LAST LOAD OPERATION (SEE SUBR. LOAD)
C             1 IF CHANGED (SEE SUBR. CHANGE)
C     KEYWD:  5 CHARACTER IDENTIFIER FOR EACH PARAMETER (SUBR. CHANGE)
C     RESNAM: IDENTIFIER FOR RESULT (SUBR. PROGRM)
C     MAP:    POINTER TO RAIL.HLP (SUBR. HELP)
C     MAXHLP: NUMBER OF RECORDS IN RAIL.HLP (SUBR. HELP)
      COMMON /FAC/IBRIEF,MODBAS(4),NRES
     1,MATCH,LEVEL,NFILE,NPARM,IMODEL(4),IDIR(4,20)
     1/DAT/PARAM(80),RESULT(20),ICHANG(80)
     2/LST/KEYWD(80),RESNAM(20)
     3/HLP/MAP(200),MAXHLP
C
      LEVEL=101
```

```
C       RAIL.DAT:       RECORD 1 CONTAINS NUMBER OF RECORDS TO FOLLOW
C               REMAINING RECORDS CONTAIN SETS OF PARAMETER VALUES
        OPEN(UNIT=21,DEVICE='DSK',FILE='RAIL.DAT',ACCESS='RANDOM'
        1,MODE='ASCII',RECORD SIZE=1143,ERR=210)
C       RAIL.HLP:       RECORDS 1 TO 8 CONTAIN MAXHLP, MAP
C               REMAINING RECORDS CONTAIN MESSAGE TO BE OUTPUT BY
C               SUBR. HELP
        OPEN(UNIT=22,DEVICE='DSK',FILE='RAIL.HLP',ACCESS='RANDOM'
        1,MODE='ASCII',RECORD SIZE=76,ERR=200)
        READ(22#1,10,ERR=230) MAXHLP,(MAP(I),I=1,23)
        READ(22#2,10,ERR=230) (MAP(I),I=24,47)
        READ(22#3,10,ERR=230) (MAP(I),I=48,71)
        READ(22#4,10,ERR=230) (MAP(I),I=72,95)
        READ(22#5,10,ERR=230) (MAP(I),I=96,119)
        READ(22#6,10,ERR=230) (MAP(I),I=120,143)
        READ(22#7,10,ERR=230) (MAP(I),I=144,167)
        READ(22#8,10,ERR=230) (MAP(I),I=168,191)
10      FORMAT(24I3,4X)
C       UNIT 19 SO OUTPUT CAN BE EASILY SENT TO PRINTERS WITH
C               MINOR PROGRAM CHANGES
15      OPEN(UNIT=19,DEVICE='TTY',ACCESS='SEQINOUT')
C       READ DIRECTORY FROM RAIL.DAT . FIRST 20 CHARACTERS OF EACH


C  <BENDER>RAIL.FOR;112   Thu 5-Feb-81 3:10PM        PAGE 1:1


C               FILE IN RAIL.DAT IS THE NAME OF THE FILE.
        CALL REDIR
20      FORMAT(I4)
40      FORMAT(4A5)
50      TYPE 60
60      FORMAT(' TYPE HELP FOR INSTRUCTIONS.')
C       SETNAM: SET NRES,NPARM,KEYWD TO RAILROAD VALUES(SUBR. PROGRM)
        CALL SETNAM
C       DLOAD;  LOAD MODEL SPECIFIED BY IMODEL INTO PARAM (SUBR. LOAD)
        CALL DLOAD
C       THIS IS THE CENTRAL POINT OF THE PROGRAM.
70      TYPE 80
80      FORMAT(' INSTRUCTION:',$)
C       LEVEL EQUALS 101 ONLY IF PROGRAM HAS JUST BEEN ENTERED.
C               A MORE DETAILED MESSAGE SHOULD BE GIVEN FOR THE NEW USER
        IF(LEVEL.NE.101) LEVEL=106
C       GET INSTRUCTION FROM USER
        ACCEPT 90,ANS
90      FORMAT(A5)
C       EXECUTE INSTRUCTION
        CALL LOOK(ANS)
C       IF USER'S INSTRUCTION VALID GET NEXT INSTRUCTION
```

```
      IF(MATCH.EQ.1) GO TO 70
      IF(ANS.EQ.'STOP') GO TO 110
      TYPE 100
100   FORMAT(' NOT A COMMAND. TYPE HELP FOR A LIST OF COMMANDS.')
      GO TO 70
C     CHECK THAT USER IS FINISHED WITH PROGRAM
110   TYPE 120
120   FORMAT(' ARE YOU FINISHED WITH THE PROGRAM?',$)
      ACCEPT 90,ANS
      IF(ANS.NE.'YES') GO TO 70
      CLOSE(UNIT=21)
      CLOSE(UNIT=22)
      STOP
C
C     ERROR PROCEDURES
200   TYPE 211
211   FORMAT(' ERROR WHILE OPENING RAIL.HLP')
      GO TO 15
210   TYPE 220
220   FORMAT(' ERROR WHILE OPENING RAIL.DAT')
      STOP
230   MAXHLP=8
      DO 240 I=1,200
240   MAP(I)=3
      GO TO 15
      END
C
C
C
```

```
C  <BENDER>RAIL.FOR;112   Thu 5-Feb-81 3:10PM          PAGE 1:2

C
      BLOCK DATA
      COMMON /FAC/IBRIEF,NOBBAS(4),NRES
      1,MATCH,LEVEL,NFILE,NPARM,IMODEL(4),IDIR(4,20)
      1/DAT/PARAM(80),RESULT(20),ICHANG(80)
      2/LST/KEYWD(80),RESNAM(20)
      3/HLP/MAP(200),MAXHLP
```

```
      DATA ICHANG/80*0/
      DATA IBRIEF/0/
      DATA MODBAS/'DEFAULT             '/
      DATA IMODEL/'DEFAULT             '/
      END
C
C
C
C     CALL SUBROUTINE SPECIFIED BY USER
      SUBROUTINE LOOK(JANS)
      COMMON /FAC/IBRIEF,MODBAS(4),NRES
     1,MATCH,LEVEL,NFILE,NPARM,IMODEL(4),IDIR(4,20)
5     MATCH=0
      IANS=JANS
      IF(IANS.EQ.'CURRE') CALL LOAD
      IF(IANS.EQ.'BASE') CALL CLOAD
      IF(IANS.EQ.'SETNA') CALL SETNAM
      IF(IANS.EQ.'DELET') CALL DELETE
      IF(IANS.EQ.'DIR') CALL DIR
      IF(IANS.EQ.'DIREC') CALL DIR
      IF(IANS.EQ.'LOAD') CALL LOAD
      IF(IANS.EQ.'CHANG') CALL CHANGE
      IF(IANS.EQ.'STORE') CALL STORE
      IF(IANS.EQ.'HELP') CALL HELP
      IF(IANS.EQ.'CHELP') CALL CHELP
      IF(IANS.EQ.'RUN') CALL DIFF
      IF(IANS.EQ.'LHELP') CALL LHELP
      IF(IANS.EQ.'LIST') CALL LIST
      IF(IANS.EQ.'OUTPU') CALL OUTPUT
      IF(IANS.EQ.'MAPDU') CALL MAPDU
      IF(IANS.EQ.'BRIEF') IBRIEF=1
      IF(IANS.EQ.'NOBRI') IBRIEF=0
      IF(IANS.EQ.'NOBRI'.OR.IANS.EQ.'BRIEF') MATCH=1
      IF(MATCH.EQ.1)  GO TO 15
      REREAD 30,IC,IWORD
30    FORMAT(A2,A5)
      IF(IC.EQ.'C ') CALL BCHNGE(IWORD)
      IF(MATCH.EQ.0) GO TO 20
15    MATCH=1
      TYPE 10
10    FORMAT(2X)
20    IANS=0
      RETURN
      END
```

```
C
C
C       READ FROM RAIL.HLP AND OUTPUT
        SUBROUTINE HELP
        INTEGER IRNT(3),LWITH(4),HLPFOR(16),IOUT(72),BLANK(15),TEXT(15)
        COMMON /FAC/IBRIEF,MODBAS(4),NRES
        1,MATCH,LEVEL,NFILE,NPARM,IMODEL(4),IDIR(4,20)
        1/HLP/MAP(200),MAXHLP
        DATA IRNT/'(', '''' , ' '/
        DATA LWITH/ '''' , ',' , '$' , ')' /    !',$)
C       IOUT:   AREA TO STORE A LINE OF CHARACTERS
C       BLANK:  DUMMY ARGUMENT
C       TEXT:   AREA TO STORE A LINE IN A5 FORM
C       MAP CONTAINS THE STARTING POINT IN A CHAIN OF RECORDS THAT
C               CONTAINS THE MESSAGE TO BE OUTPUT.
        I=MAP(LEVEL)
        GO TO 10
C       ENTRY POINT FOR THE REST OF THE PROGRAM TO ACCESS MESSAGES.
C               SUBR. OUTPUT,LIST
        ENTRY AHELP(I2)
C       FIND STARTING POINT AS BEFORE
        I=MAP(I2)
C       RECORD NUMBERS LESS THAN 8 ARE NOT MESSAGES
10      IF(I.LE.8) GO TO 50
C       RECORDS GREATER THAN MAXHLP ARE NOT PRESENT
        IF(I.GT.MAXHLP) GO TO 50
C       READ A LINE
        READ(22@I,19,ERR=30) J,IOUT,K
19      FORMAT(I3,72A1,I1)
20      FORMAT(I3,14A5,A2,I1)
        DO 23 L=72,2,-1
23      IF(IOUT(L).NE.' ') GO TO 310
        WRITE(19,300)
300     FORMAT(1X)
        GO TO 27
310     IF(IOUT(L).EQ.'@') GO TO 330
        IF(IOUT(L).EQ.';') L=L-1
        WRITE(19,320) (IOUT(LL),LL=1,L)
320     FORMAT(1X,72A1)
        GO TO 27
330     L=L-1
        IF(IOUT(L).EQ.';') L=L-1
        ENCODE (80,340,HLPFOR) IRNT,(IOUT(LL),LL=1,L),LWITH
```

```
340     FORMAT(80A1)
        WRITE(19,HLPFOR)
        GO TO 27
C       DOLLAR SIGN SUPPRESSES CARRAIGE RETURN SO USER MAY RESPOND
C       K EQUALS 1 ONLY AT END OF CHAIN OF LINES IN MESSAGE
27      IF(K.EQ.1) GO TO 25
C       J IS RECORD NUMBER FOR NEXT LINE, I IS R.N. OF CURRENT LINE
        I=J
        GO TO 10
```



```
C       SUCCESSFUL COMPLETION, RETURN TO CALLING PROGRAM
25      MATCH=1
        LEVEL=3
        RETURN
C       ERROR IN SUBR., DISPLAY DIAGNOSTIC INFORMATION
30      TYPE 40,LEVEL,I
40      FORMAT(' PROGRAM ERROR. LEVEL=',I3,'   I=',I3)
        GO TO 25
C       NO MESSAGE IS AVAILABLE, OUTPUT GENERAL MESSAGE
50      TYPE 60
60      FORMAT(' TYPE STOP TO RETURN TO MAIN LEVEL.')
        GO TO 25
C
C       CHANGE RESPONSE TO HELP
        ENTRY LHELP
C       PROMPT FOR LEVEL NUMBER FOR MESSAGE TO BE CHANGED
215     TYPE 220
220     FORMAT(' LEVEL:',$)
        READ(5,230,ERR=25) LEVEL
230     FORMAT(I3)
C
C       CHANGE MESSAGE FOR CURRENT LEVEL
        ENTRY CHELP
C       NEW:    0 IF LINE WILL REPLACE AN ALREADY EXISTING RECORD IN
C               RAIL.HLP
C               1 IF NEW RECORD IS CREATED
        NEW=0
C       FIND STARTING POINT IN CHAIN FOR EXISTING MESSAGE (SUBR. HELP)
```

```
        K=MAP(LEVEL)
C       K WILL BE LESS THAN 9 ONLY IF NO MESSAGE EXISTS
C       ICHMAP: 1 IF MAP HAS CHANGED, 0 IF NO CHANGE MADE YET
        ICHMAP=0
        IF(K.GT.8) GO TO 65
C       NO MESSAGE EXISTS, CREATE A NEW RECORD
        MAP(LEVEL)=MAXHLP+1
        NEW=1
C       PROMPT USER FOR MESSAGE
C       MAP HAS CHANGED SO REMEMBER TO CORRECT IT WHEN FINISHED
        ICHMAP=1
65      TYPE 70
70      FORMAT(' ENTER A LINE.')
        ACCEPT 80,TEXT
80      FORMAT(14A5,A2)
90      TYPE 100
100     FORMAT(' DO YOU WANT TO ENTER ANOTHER LINE (A CONTINUATION)?',$)
C       M:      'NO' IF THIS IS TO BE LAST LINE IN MESSAGE
C               ANYTHING ELSE MEANS YES
        READ(5,110,ERR=90) M
110     FORMAT(A5)
        LAST=0
        IF(M.EQ.'NO') LAST=1
```

C <BENDER>RAIL.FOR;112 Thu 5-Feb-81 3:10PM PAGE 1:5

```
C       IF THIS RECORD IS NEW INCREMENT MAXHLP
        IF(NEW.EQ.1) GO TO 180
C       READ LOCATION OF NEXT RECORD IN EXISTING CHAIN
        READ(22WK,20,ERR=200) NEXT,BLANK,J
C       IF THIS IS LAST LINE IN EXISTING CHAIN, NEXT LINE WILL BE NEW
        IF(NEXT.EQ.0) GO TO 190
C       IF NEW LINE IS BEING CREATED AND WILL BE LAST, END THE CHAIN
120     IF(NEW.EQ.1.AND.LAST.EQ.1) NEXT=0
C       WRITE LINE TO RAIL.HLP
130     WRITE(22WK,20,ERR=30) NEXT,TEXT,LAST
C       IF FINISHED WRAP-UP
        IF(LAST.EQ.1) GO TO 150
C       CHANGE NEXT TO CURRENT AND REPEAT
```

```
        K=NEXT
        GO TO 65
C       WRITE NEW VERSION OF MAP IF IT HAS CHANGED
150     MATCH=1
C       SKIP IF NO CHANGE
        IF(ICHMAP.EQ.0) GO TO 179
        WRITE(22#1,290) MAXHLP,(MAP(I),I=1,23)
        WRITE(22#2,290) (MAP(I),I=24,47)
        WRITE(22#3,290) (MAP(I),I=48,71)
        WRITE(22#4,290) (MAP(I),I=72,95)
        WRITE(22#5,290) (MAP(I),I=96,119)
        WRITE(22#6,290) (MAP(I),I=120,143)
        WRITE(22#7,290) (MAP(I),I=144,167)
        WRITE(22#8,290) (MAP(I),I=168,191)
290     FORMAT(24I3,4X)
179     RETURN
C
C       INCREMENT MAXHLP BECAUSE RECORD IS BEING CREATED
180     MAXHLP=MAXHLP+1
C     IF MAXHLP CHANGES  WE MUST WRITE OUT FIRST RECORD
       ICHMAP=1
C       SET CURRENT RECORD NUMBER TO NEWLY CREATED RECORD
        K=MAXHLP
C       NEXT WILL BE CREATED AT THE NEXT ROUND
        NEXT=MAXHLP+1
C       NEW LINE HAS BEEN CREATED
        NEW=1
        GO TO 120
C       NEW MESSAGE ENDS ON SAME LINE AS OLD MESSAGE SO NO CHANGE MADE
190     IF(LAST.EQ.1) GO TO 130
C       NEW MESSAGE IS LONGER THAN OLD MESSAGE
        NEXT=MAXHLP+1
        NEW=1
        GO TO 130
C       PRINT DIAGNOSTIC
200     TYPE 210,K
210     FORMAT(' PROGRAM ERROR. READ22 AT K=',I3)
        GO TO 25
        END
```

```
C
C
C
C       MOVE PARAMETERS FROM FILE TO CURRENT
        SUBROUTINE LOAD
        COMMON /FAC/IBRIEF,MODBAS(4),NRES
        1,MATCH,LEVEL,NFILE,NPARM,IMODEL(4),IDIR(4,20)
        1/DAT/PARAM(80),RESULT(20),ICHANG(80)
        2/LST/KEYWD(80),RESNAM(20)
        INTEGER ANS(4)
        GO TO 100
C
C       REST OF PROGRAM CAN TEMPORARILY STORE AND RETREIVE DATA
        ENTRY ALOAD(I2)
        I=I2
C       SKIP A RECORD BECAUSE FIRST ENTRY IN RAIL.DAT IS NFILE
5       I=I+1
C       READ PARAMETERS INTO CURRENT FILENAME, CURRENT PARAMETERS
        READ(21@I,10,ERR=50) IMODEL,PARAM,NPARM
10      FORMAT(4A5,80E14.8,I3)
        DO 20 J=1,80
C       RESET ICHANG BECAUSE OLD CHANGES ARE NO LONGER VALID
20      ICHANG(J)=0
40      MATCH=1
        RETURN
50      CALL ERRSNS(I,J)
        TYPE 60,I,J
60      FORMAT(' PROGRAM ERROR IN LOAD.  FIRST=',I3,'    SECOND=',I3)
        GO TO 40
C       PROMPT USER FOR FILENAME
100     TYPE 110
110     FORMAT(' LOAD VALUES STORED IN FILE:',$)
115     LEVEL=102
        ACCEPT 120,ANS
120     FORMAT(4A5)
C       FIND FILENAME IN DIRECTORY
        CALL LOCATE(ANS,I)
C       SEE SUBR. LOCATE FOR DESCRIPTION OF MATCH IN THIS CASE
        GO TO (5,40,100,130) MATCH
C       FILE NOT FOUND IN DIRECTORY, TRY AGAIN
130     TYPE 140
140     FORMAT(' FILE NOT FOUND.')
        GO TO 100
C
C       LOAD BASE CASE AS SPECIFIED BY MODBAS
        ENTRY BLOAD
C       FIND CONTENTS OF MODBAS IN DIRECTORY
        CALL ALOC(IMODEL,I)
C       IF NOT FOUND TYPE WARNING
```

```
      IF(MATCH.NE.1) GO TO 170
C     SKIP A RECORD BECAUSE OF RAIL.DAT
```



```
150   I=I+1
C     DONT CHANGE IMODEL
      READ(21WI,10,ERR=50) IMODEL,PARAM,NPARM
      GO TO 40
C
C     LOAD BASE CASE AS SPECIFIED BY USER
      ENTRY CLOAD
C     PROMPT FOR FILENAME
160   TYPE 110
      LEVEL=107
165   ACCEPT 120,ANS
      IF(ANS(1).EQ.'STOP') GO TO 40
C     FIND RESPONSE IN DIRECTORY
      CALL LOCATE(ANS,I)
      GO TO (150,40,160,170) MATCH
170   TYPE 190
      GO TO 165
190   FORMAT(' THE FILE SPECIFIED FOR BASE CASE IS NOT FOUND.',/
      1,' PLEASE REENTER OR TYPE STOP:',$)
C
C     LOAD AS SPECIFIED BY IMODEL
      ENTRY DLOAD
C     FIND NAME OF CURRENT MODEL IN DIRECTORY
      CALL ALOC(IMODEL,I)
C     IF NOT FOUND GIVE WARNING
      IF(MATCH.NE.1) GO TO 200
C     LOAD VALUES IN FILE
      GO TO 5
200   TYPE 210
210   FORMAT(' FILE SPECIFIED FOR CURRENT MODEL NOT FOUND.',/
      1,' PLEASE REENTER OR TYPE STOP:',$)
      GO TO 115
C
C     LOAD STORED VALUES OF ICHANG
```

```
      ENTRY LDCHNG(I3)
      I=I3+1
C     SEE SUBR. DIFF FOR REASON FOR STORING ICHANG
      READ(21#I,220,ERR=50) ANS,ICHANG
220   FORMAT(4A5,80I1,1043X)
      GO TO 40
      END
C
C
C
C
C     CHANGE VALUE OF A PARAMETER
C     THIS SUBROUTINE SEARCHES THE LIST KEYWD(80) FOR THE KEYWORD
C     THAT THE USER ENTERS. IF IT IS FOUND CALL AHELP TO TYPE THE
C     QUESTION CORRESPONDING TO THE KEYWORD. THE USER THEN ENTERS
C     THE NEW VALUE AND RETURNS TO THE CALLING SUBROUTINE. IF THE
C     USERS RESPONSE IS NOT A NUMBER ,REREAD THE RESPONSE AND STOP
```



```
C     CHECK IT AGAINST KEYWD  WHICH WILL RESULT IN A WARNING
C     MESSAGE. ENTRY BCHNGE IS TO ALLOW THE ABBREVIATED FORM
C     C <KEYWORD> INSTEAD OF CHANGE <RETURN> <KEYWORD>. THIS IS
C     DONE BY THE REREAD IN SUBR. LOOK
      SUBROUTINE CHANGE
      COMMON /FAC/IBRIEF,NODBAS(4),NRES
      1,MATCH,LEVEL,NFILE,NPARM,IMODEL(4),IDIR(4,20)
      1/DAT/PARAM(80),RESULT(20),ICHANG(80)
      2/LST/KEYWD(80),RESNAM(20)
      INTEGER TEXT(15)
      GO TO 10
      ENTRY BCHNGE(IWORD)
      IANS=IWORD
      GO TO 35
10    TYPE 20
20    FORMAT(' KEYWORD OF PARAMETER:',$)
      LEVEL=103
      ACCEPT 30,IANS
30    FORMAT(A5)
```

```
      IF(IANS.EQ.'STOP') GO TO 125
      CALL LOOK(IANS)
      IF(MATCH.EQ.1) GO TO 10
35    IF(IANS.EQ.'     ') GO TO 10
      DO 50 I=1,80
      IF(IANS.NE.KEYWD(I)) GO TO 50
C
C     SUCCESSFUL SEARCH
      GO TO 100
50    CONTINUE
C
C     KEYWD NOT FOUND
      TYPE 60
60    FORMAT(' THE KEYWORD YOU ENTERED IS NOT ON THE LIST. TYPE '
     1,'LIST FOR A LIST',/,' OF KEYWORDS. TYPE STOP IF YOU DO NOT'
     2,' WANT TO CHANGE A PARAMETER.')
      GO TO 10
C
C     TYPE QUESTION, READ ANSWER
100   CALL AHELP(I)
110   READ(5,120,ERR=130) X
120   FORMAT(E14.0)
      PARAM(I)=X
      ICHANG(I)=1
125   MATCH=1
      RETURN
C
C     READ ERROR
130   LEVEL=I
      ACCEPT 30,IANS
      IF(IANS.EQ.'STOP') GO TO 125
      CALL LOOK(IANS)
      IF(MATCH.EQ.1) GO TO 100
      GO TO 35
      END
```

```
C  <BENDER>RAIL.FOR;112   Thu 5-Feb-81 3:10PM        PAGE 1:9


C
C
C
C     OUTPUT RESULTS
```

```
      SUBROUTINE OUTPUT
      COMMON /FAC/IBRIEF,MODBAS(4),NRES
     1,MATCH,LEVEL,NFILE,NPARM,IMODEL(4),IDIR(4,20)
     1/DAT/PARAM(80),RESULT(20),ICHANG(80)
     2/LST/KEYWD(80),RESNAM(20)
      TYPE 10
10    FORMAT(1X,//
     1,' THE FOLLOWING PARAMETERS HAVE BEEN CHANGED:')
      DO 30 I=1,NPARM
      IF(ICHANG(I).EQ.0) GO TO 30
      TYPE 20,KEYWD(I),PARAM(I)
20    FORMAT(1X,A5,F13.2,2X,$)
      J=I
      IF(IBRIEF.EQ.1) GO TO 25
      CALL AHELP(J)
25    TYPE 35
35    FORMAT(2X)
30    CONTINUE
      TYPE 40,IMODEL
40    FORMAT(//,' BASE CASE: ',4A5
     1,/,' USING THESE PARAMETERS THE FOLLOWING RESULTS ARE '
     1,'OBTAINED.')
      DO 50 I=1,NRES
      TYPE 20,RESNAM(I),RESULT(I)
      J=I+80
      IF(IBRIEF.EQ.0) GO TO 45
      TYPE 35
      GO TO 50
45    CALL AHELP(J)
50    CONTINUE
      TYPE 70
70    FORMAT(////)
      MATCH=1
      RETURN
      END
C
C
C
C
C     LIST CURRENT VALUE OF ALL PARAMETERS
      SUBROUTINE LIST
      COMMON /FAC/IBRIEF,MODBAS(4),NRES
     1,MATCH,LEVEL,NFILE,NPARM,IMODEL(4),IDIR(4,20)
     1/DAT/PARAM(80),RESULT(20),ICHANG(80)
     2/LST/KEYWD(80),RESNAM(20)
      TYPE 4,IMODEL
4     FORMAT(///,' CURRENT MODEL: ',4A5,/)
      TYPE 5
```

```
5       FORMAT(/,' KEYWORD   VALUE',/)
        IF(NPARM.GE.1) GO TO 30
        TYPE 10
10      FORMAT(' PROGRAM ERROR. NPARM LESS THAN 1')
20      MATCH=1
        RETURN
30      DO 50 I=1,NPARM
        TYPE 35,KEYWD(I),PARAM(I)
35      FORMAT(1X,A5,1X,F14.4,2X,$)
        J=I
        IF(IBRIEF.EQ.1) GO TO 36
        CALL AHELP(J)
36      TYPE 40
40      FORMAT(1X)
50      CONTINUE
        GO TO 20
        END
C
C
C
C       PRINT DIRECTORY
        SUBROUTINE DIR
        COMMON /FAC/IBRIEF,MODBAS(4),NRES
        1,MATCH,LEVEL,NFILE,NPARM,IMODEL(4),IDIR(4,20)
        1/DAT/PARAM(80),RESULT(20),ICHANG(80)
        2/LST/KEYWD(80),RESNAM(20)
        IF(NFILE.LE.0) GO TO 40
        DO 10 I=1,NFILE
10      TYPE 20,I,(IDIR(J,I),J=1,4)
20      FORMAT(1X,I2,5X,4A5)
30      MATCH=1
        RETURN
40      TYPE 50
50      FORMAT(/,'  DIRECTORY IS EMPTY.',/)
        NFILE=0
        GO TO 30
        END
C
C
C
```

```
C       FIND LOCATION OF FILE IN DIRECTORY
        SUBROUTINE LOCATE(ANS,I)
        COMMON /FAC/IBRIEF,MODBAS(4),NRES
        1,MATCH,LEVEL,NFILE,NPARM,IMODEL(4),IDIR(4,20)
        1/DAT/PARAM(80),RESULT(20),ICHANG(80)
        2/LST/KEYWD(80),RESNAM(20)
        INTEGER ANS(4)
10      IF(ANS(1).EQ.'STOP') GO TO 60
        CALL LOOK(ANS(1))
        IF(MATCH.EQ.1) GO TO 70
        ENTRY ALOC(ANS,I)
        DO 30 I=1,NFILE
        DO 20 J=1,4
```

C <BENDER>RAIL.FOR;112 Thu 5-Feb-81 3:10PM PAGE 1:11

```
20      IF(ANS(J).NE.IDIR(J,I)) GO TO 30
        GO TO 50
30      CONTINUE
        GO TO 80
C
C       FOUND
50      MATCH=1
        RETURN
C
C       STOP
60      MATCH=2
        RETURN
C
C       REPEAT PROMPT
70      MATCH=3
        RETURN
C
C       NOT FOUND
80      MATCH=4
        RETURN
        END
C
C
C
C       STORE CURRENT PARAMETERS IN FILE
```

```
      SUBROUTINE STORE
      COMMON /FAC/IBRIEF,MODBAS(4),NRES
     1,MATCH,LEVEL,NFILE,NPARM,IMODEL(4),IDIR(4,20)
     1/DAT/PARAM(80),RESULT(20),ICHANG(80)
     2/LST/KEYWD(80),RESNAM(20)
      INTEGER ANS(4)
      GO TO 100
      ENTRY ASTORE(I2)
      I=I2
10    I=I+1
      WRITE(21#I,20,ERR=50) IMODEL,PARAM,NPARM
20    FORMAT(4A5,80E14.8,I3)
40    MATCH=1
      RETURN
50    TYPE 51
51    FORMAT(' PROGRAM ERROR IN STORE.')
      GO TO 40
100   TYPE 110
110   FORMAT(' STORE IN FILE CALLED:',$)
      LEVEL=104
      ACCEPT 120,ANS
120   FORMAT(4A5)
      CALL LOCATE(ANS,I)
      GO TO (10,40,100,130) MATCH
130   NFILE=NFILE+1
      I=NFILE
      DO 140 J=1,4
      IDIR(J,I)=ANS(J)
```

C <BENDER>RAIL.FOR;112 Thu 5-Feb-81 3:10PM PAGE 1:12

```
140   IMODEL(J)=ANS(J)
      WRITE(21#1,995) NFILE
995   FORMAT(I3,1140X)
      GO TO 10
C
C     STORE CURRENT VALUES OF ICHANG
      ENTRY STRCHG(I3)
150   I=I3+1
160   WRITE(21#I,180,ERR=50) IMODEL,ICHANG
180   FORMAT(4A5,80I1,1043X)
```

```
170     GO TO 40
        END
C
C
C
C       READ THE DIRECTORY FROM UNIT 21
        SUBROUTINE REDIR
        COMMON /FAC/IBRIEF,MODBAS(4),NRES
        1,MATCH,LEVEL,NFILE,NPARM,IMODEL(4),IDIR(4,20)
20      FORMAT(4A5,1123X)
5       FORMAT(I3,1140X)
        READ(21#1,5,ERR=200) NFILE
        IF(NFILE.LE.0) GO TO 30
        DO 120 I=2,NFILE+1
120     READ(21#I,20,ERR=100) (IDIR(J,I-1),J=1,4)
        GO TO 30
200     NFILE=0
30      MATCH=1
        RETURN
100     TYPE 110
110     FORMAT(' PROGRAM ERROR IN REDIR.')
        GO TO 200
        END
C
C
C
C
C       REMOVE A FILE FROM THE DIRECTORY
        SUBROUTINE DELETE
        COMMON /FAC/IBRIEF,MODBAS(4),NRES
        1,MATCH,LEVEL,NFILE,NPARM,IMODEL(4),IDIR(4,20)
        INTEGER ANS(4)
        DIMENSION XPARAM(80)
        GO TO 10
        ENTRY ADEL(I2)
        I=I2
        GO TO 50
10      TYPE 20
20      FORMAT(' DELETE FILE NAMED:',$)
        LEVEL=105
        ACCEPT 30,ANS
```

```
30      FORMAT(4A5)
        CALL LOCATE(ANS,I)
        GO TO (50,40,10,90) MATCH
40      MATCH=1
        RETURN
50      IF(I.LT.NFILE) GO TO 60
        NFILE=NFILE-1
        WRITE(21#1,995) NFILE
        GO TO 40
60      NFILE=NFILE-1
        WRITE(21#1,995) NFILE
995     FORMAT(I3,1140X)
        DO 80 J=I+1,NFILE+1
        K=J+1
        READ(21#K,70,ERR=110) ANS,XPARAM,MPARM
70      FORMAT(4A5,80E14.8,I3)
        WRITE(21#J,70) ANS,XPARAM,MPARM
80      CONTINUE
        CALL REDIR
        GO TO 40
90      TYPE 101
101     FORMAT(' FILE NOT FOUND.')
        GO TO 10
110     CALL ERRSNS(I,J)
        TYPE 120,K,I,J
120     FORMAT(' PROGRAM ERROR IN DELETE. K=',I3,' I=',I3,' J=',I3)
        GO TO 40
        END
C
C
C
C       DO SIMULATION
        SUBROUTINE PRGRM
        IMPLICIT REAL(A-Z)
        INTEGER I,NRES,MATCH,LEVEL,NFILE,IMODEL,IDIR,NAM,RESNAM
        1,NPARM,ICHANG
        REAL T(9)
        COMMON /FAC/IBRIEF,MODBAS(4),NRES
        1,MATCH,LEVEL,NFILE,NPARM,IMODEL(4),IDIR(4,20)
        2/LST/NAM(80),RESNAM(20)
        3/DAT/P1A1,P1BY,P1CY,HBT,HBC,CPT,LPT,SHB,RHB,WIC,CAC
        1,OAC,OK,CH,SU,SSP,CUP,RAB,CBTO,CBCO,CBT70,CBC70,BC,MT,SA,BF
        2,P2BY,P2CY,HBCC,IC,ACL,VHC,
     1  P3A(8),PBOC,P3BY,P3CY,P4ARL,P4AY,P4AYE
        3,CTPDT,P4BY,P4CY,P4DY,RCSZ,YCSZ,CI,RLPT,RLAB,RRLOC,RSE,RC
        4,BOC,SPH,P2AYH,P2AYF,CTPTH,CTPTF,BUL,XOK,PHBS,
```

```
     1 P2DY,P2EY,P2FY,PAHC,CYA,DYA,PAAC,PCBH,SEPT,FWEFD,NULL(3)
       8,URC,UYC,UCI,URL,USE,UC,CURC,CUYC,CUCI,CUL,CUSE,CUC,NULL1(8)
       8,ICHANG(80)
C
C
C  <BENDER>RAIL.FOR;112   Thu 5-Feb-81 3:10PM         PAGE 1:14

C       EQUATIONS
C       CHECK FOR DIVIDE BY ZERO
10      IF(VHC.EQ.0.0) GO TO 200
        IF(CPT.EQ.0.0) GO TO 210
        IF(CTPDT.EQ.0.0) GO TO 220
C       YARD TRAIN
        T11=MT+SA+HBT*(SHB+2*WIC)+CAC+OK+SU
        T12=WIC*CPT/2+CAC+RAB+OK+SU
        T13=MT+SA+HBC*(SHB+2*WIC)+CAC+OK+SU
        T14=MT+SSP+CUP+CH+OAC+CBC0
        T15=WIC*CPT/2+HBC*(SHB+2*WIC)+CAC+RAB+OK+SU
        T16=T13
        T17=MT+SSP+CUP+CH+OAC+HBC*(RHB+2*WIC)
        T18=MT+SA+HBC*(SHB+2*WIC)+CAC+OK+SU
        T19=MT+SSP+CUP+BC+OK+SU
        T110=BF+CPT*(IC+PBOC*BOC+WIC+0.2*BC)
        TY=P1A1*T11+(1-P1A1)*((P1BY*(T12+T14)+(1-P1BY)*(T15+T17))
        1+T13+T18)+(1-P1CY)*T19
C
C       CLASSIFICATION
        T21=MT+SSP+CUP
        T22=HBC*(RHB+2*WIC)
        CPC=CPT/CTPTH
        T23=HBC*(RHB+WIC)+WIC*CPC+HBC*(SHB+WIC)+OK+SU
        T24=MT
        T25=ACL/(VHC*88)
        T26=SPH
        T27=30
      TPCUTH=T21+P2AYH*T22+(1-P2AYH)*T23+T24
      TCSH=P2BY*T26+(1-P2BY)*T25
      T2H=CTPTH*TPCUTH+CPT*TCSH
```

```
      T28=BUL
      T29=SPH
      T210=XOK
      T211=0.33
      T212=T27
      CPC=CPT/CTPTF
      T23=HBC*(RHB+WIC)+WIC*CPC+HBC*(SHB+WIC)+OK+SU
      TPCUTF=T21+P2AYF*T22+(1-P2AYF)*T23+T24
      TCSF=P2DY*T29+(1-P2DY)*(P2EY*T28+P2FY*T210+T211)
      T2F=CTPTF*TPCUTF+CPT*TCSF
C
C       PULL DOWN
        T31=MT
        T32=SSP+CUP+PHBS*(HBC*(RHB+2*WIC))
        T(1)=0
        T(2)=1
        T(3)=1
        T(4)=1
        T(5)=60
        T(6)=1
        T(7)=60
        T(8)=1
        T33=0
        DO 40 I=1,8
```

C <BENDER>RAIL.FOR;112 Thu 5-Feb-81 3:10PM PAGE 1:15

```
40      T33=T33+(67.2/4.0)*(WIC/8.+P3A(I)*T(I))*(1-PWEPD)
        T34=MT
        T35=HBT*(SHB+2*WIC)+OK+SU
        T36=SSP+CUP+OK+SU
        T37=MT+SSP+CUP+CH+2*OAC+WIC+CAC+OK+SU
      TPD=4.*(T31+T32+T33+T34+P3BY*T35+(1-P3BY)*T36)+P3CY*T37
C
C       CHARGE AIR AND POWER BRAKE TEST
      T41=BF+CAC
      T42=MT+SSP+CUP+CH+OAC+CPT*(WIC+PAHC*CH)*(1-PAAC)
      T43=CYA+CPT*(WIC+PAHC*CH)*(1-PAAC)+BYA
      T44=MT+CH+OAC+CPT*(WIC+PAHC*CH)*(1-PAAC)+CAC+MT
      T45=MT+SSP+CUP+CH+OAC+CBT70
      T46=T45
      T47=HBT*(2*WIC+RHB)
      T48=0
```

```
      T49=30
      T410=5
      T411=30
      T412=5
      T413=CPT*(IC+UIC)*(1-PCBM)
      T414=T413
      TT415=30
      TCB=T41+P4ARL*T42+P4AY*(T43+T45)+P4AYE*(T44+T46)+T47
      TPBT=(1-P4BY)*T49+T410+(1-P4CY)*T411+T412+T413
     1 +T414+(1-P4DY)*T415
C
C       COST EQUATIONS
      URC=2522*(TY+(P4ARL*T42)/CI+(1-P4ARL)*T45+T47+TPBT/CI)*RCSZ/CPT
      UYC=(506*T2H+2016*T2F+2522*(TPD+P4AYE*T44/CI))*YCSZ/CPT
      UCI=2522*(T110+TCB+TPBT)/CPT
      URL=5043342*RLPT*(TY+P4ARL*T42/CI+(1-P4ARL)*T45+T47
     1 +TPBT/CI)/CPT/1.E03
      USE=SEPT*(1012692*T2H+4030650*T2F+5043342*(TPD+P4AYE*T44/CI))/CPT
     1  /1.E03
      UC=(5043342*(TY+TPD+(T110+TCB+TPBT)/CI)+1012692*T2H+4030650*T2F)
     1  /1.E03
        CURC=RLAB*URC/1.E06
        CUYC=RLAB*UYC/1.E06
        CUCI=RLAB*UCI/1.E06
        CUL=RRLOC*URL/1.E03
        CUSE=RSE*USE/1.E03
        CUC=RC*UC/1.E03
C
50      MATCH=1
        RETURN
        ENTRY SETNAM
```

```
C  <BENDER>RAIL.FOR;112   Thu 5-Feb-81 3:10PM          PAGE 1:16

C       SET VALUES FOR MODEL SPECIFIC VARIABLES
```

```
NPARM=77
NRES=12
NAM(1)='P1A1'
NAM(2)='P1BY'
NAM(3)='P1CY'
NAM(4)='HBT'
NAM(5)='HBC'
NAM(6)='CPT'
NAM(7)='LPT'
NAM(8)='SHB'
NAM(9)='RHB'
NAM(10)='VIC'
NAM(11)='CAC'
NAM(12)='OAC'
NAM(13)='OK'
NAM(14)='CH'
NAM(15)='SU'
NAM(16)='SSP'
NAM(17)='CUP'
NAM(18)='RAD'
NAM(19)='CBTO'
NAM(20)='CBCO'
NAM(21)='CBT70'
NAM(22)='CBC70'
NAM(23)='BC'
NAM(24)='HT'
NAM(25)='SA'
NAM(26)='BF'
NAM(27)='P2BY'
NAM(28)='P2CY'
NAM(29)='HBCC'
NAM(30)='IC'
NAM(31)='ACL'
NAM(32)='VHC'
NAM(33)='P3A-1'
NAM(34)='P3A-2'
NAM(35)='P3A-3'
NAM(36)='P3A-4'
NAM(37)='P3A-5'
NAM(38)='P3A-6'
NAM(39)='P3A-7'
NAM(40)='P3A-8'
NAM(41)='PBOC'
NAM(42)='P3BY'
NAM(43)='P3CY'
NAM(44)='P4ARL'
NAM(45)='P4AY'
NAM(46)='P4AYE'
NAM(47)='CTPDT'
NAM(48)='P4BY'
```

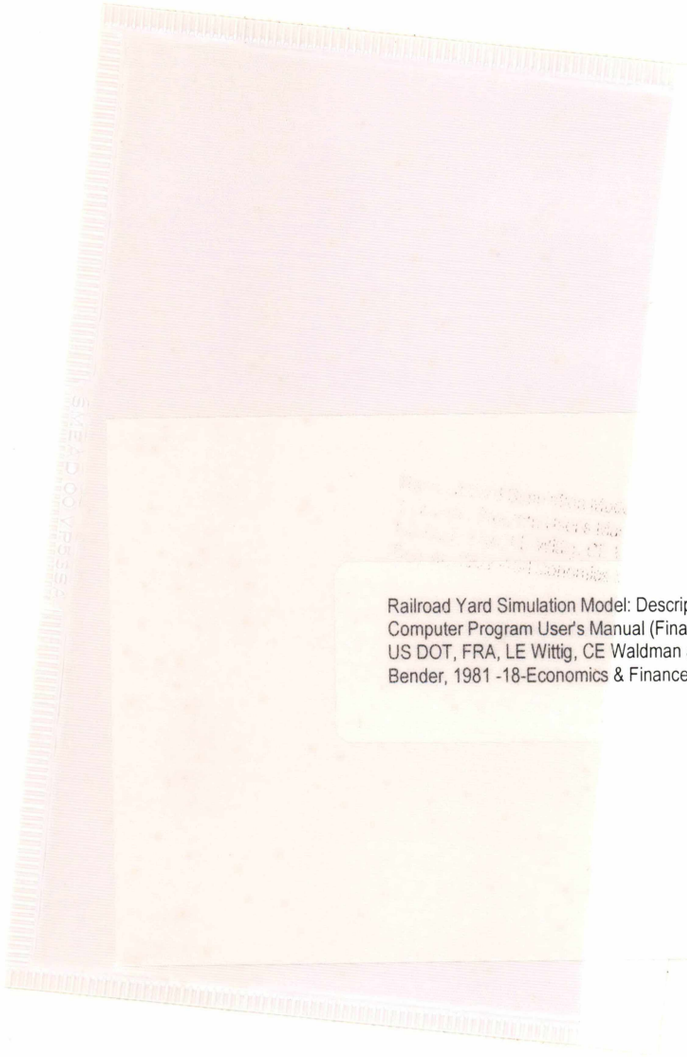
```
      NAM(49)='P4CY'
      NAM(50)='P4DY'
      NAM(51)='RCSZ'
      NAM(52)='YCSZ'
      NAM(53)='CI'
      NAM(54)='RLPT'
      NAM(55)='RLAB'
      NAM(56)='RRLOC'
      NAM(57)='RSE'
      NAM(58)='RC'
    NAM(59)='BOC'
    NAM(60)='SPH'
    NAM(61)='P2AYH'
    NAM(62)='P2AYF'
    NAM(63)='CTPTH'
    NAM(64)='CTPTF'
    NAM(65)='BUL'
    NAM(66)='XOK'
    NAM(67)='PHBS'
    NAM(68)='P2DY'
    NAM(69)='P2EY'
    NAM(70)='P2FY'
    NAM(71)='PAHC'
    NAM(72)='CYA'
    NAM(73)='DYA'
    NAM(74)='PAAC'
    NAM(75)='PCBM'
    NAM(76)='SEPT'
    NAM(77)='PUEPD'
      RESNAM(1)='URC'
      RESNAM(2)='UYC'
      RESNAM(3)='UCI'
      RESNAM(4)='URL'
      RESNAM(5)='USE'
      RESNAM(6)='UC'
      RESNAM(7)='CURC'
      RESNAM(8)='CUYC'
```

```
      RESNAM(9)='CUCI'
      RESNAM(10)='CUL'
      RESNAM(11)='CUSE'
      RESNAM(12)='CUC'
      DO 100 I=1,80
100   ICHANG(I)=0
      RETURN
200   TYPE 201
201   FORMAT(' VHC EQUALS ZERO. NO COMPUTATION DONE.')
      TYPE 222
      GO TO 50
210   TYPE 211
211   FORMAT(' CPT EQUALS ZERO. NO COMPUTATION DONE.')
      TYPE 222
      GO TO 50
```

C <BENDER>RAIL.FOR;112 Thu 5-Feb-81 3:10PM PAGE 1:18

```
220   TYPE 221
221   FORMAT(' CTPDT EQUALS ZERO. NO COMPUTATION DONE.')
      TYPE 222
222   FORMAT(' VHC,CPT, AND CTPDT CANNOT EQUAL ZERO BECAUSE THEY',/
      1,' APPEAR IN A DIVISION.')
      END
C
C
C
CC    LIST CONTENTS OF HELP FILE
      SUBROUTINE MAPDU
      COMMON /FAC/IBRIEF,MODBAS(4),NRES
      1,MATCH,LEVEL,NFILE,NPARM,IMODEL(4),IDIR(4,20)
      1/HLP/MAP(200),MAXHLP
      DO 20 I=1,200
      IF(MAP(I).LE.8) GO TO 20
      TYPE 10,I
10    FORMAT(/,' LEVEL=',I3)
      J=I
      CALL AHELP(J)
20    CONTINUE
      MATCH=1
```

```
      RETURN
      END
C
C
C
C     CALL PROGRAM, DO DIFFERENCING
      SUBROUTINE DIFF
      DIMENSION XRES(20)
      COMMON /FAC/IBRIEF,MODBAS(4),NRES
     1,MATCH,LEVEL,NFILE,MPARM,IMODEL(4),IDIR(4,20)
     1/DAT/PARAM(80),RESULT(20),ICHANG(80)
C
C     STORE CURRENT VALUES
      NFILE=NFILE+2
      J=NFILE-1
      CALL ASTORE(J)
      CALL STRCHG(J+1)
C
C     COMPUTE BASE CASE
      CALL BLOAD
5     CALL PRGRM
      DO 10 I=1,NRES
10    XRES(I)=RESULT(I)
C
C     COMPUTE CURRENT MODEL
      CALL ALOAD(J)
15    CALL PRGRM
      CALL LDCHNG(J+1)
      CALL ADEL(J+1)
      CALL ADEL(J)
```



Railroad Yard Simulation Model: Description and Computer Program User's Manual (Final Report), US DOT, FRA, LE Wittig, CE Waldman and EK Bender, 1981 -18-Economics & Finance