PB89-124309

U.S. Department of Transportation

Federal Highway Administration

Publication No. FHWA-IP-88-20
April 1988

Safety Resource Allocation Programs and Input Processor - Users Manual

Research, Development, and Technology
Turner-Fairbank Highway Research Center
6300 Georgetown Pike
McLean, Virginia 22101-2296

Technical Report Documentation Page

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
FHWA-IP-88-20	PB89-124300/AS	

4. Title and Subtitle	5. Report Date
Safety Resource Allocation Programs and Input Processor -- Users Manual	April 1988
	6. Performing Organization Code
7. Author(s)	8. Performing Organization Report No.
Charles C. Liu and Hobih Chen	
9. Performing Organization Name and Address	10. Work Unit No. (TRAIS)
SRA Technologies, Inc. 4700 King Street, Suite 300 Alexandria, Virginia 22302	NCP 3A9C0053
	11. Contract or Grant No.
	DTFH61-86-C-00006
12. Sponsoring Agency Name and Address	13. Type of Report and Period Covered
Federal Highway Administration Office of Implementation 6300 Georgetown Pike McLean, Virginia 22101-2296	Users Manual April 1986 - March 1988
	14. Sponsoring Agency Code

15. Supplementary Notes

FHWA Contract Manager: Peter J. Hatzi (HRT-20)

16. Abstract

This report is the users manual for the SRAP (Safety Resource Allocation Programs) microcomputer program package on IBM-PC and compatible microcomputers.

The SRAP package contains three computerized methodologies--incremental benefit-cost analysis, integer programming, and dynamic programming--developed by the FHWA to aid highway safety planning decisions by prioritizing projects based on their costs and benefits. The three models maximize total net accident savings under a given budget constraint by selecting the optimal mix of accident locations and the preferred countermeasure alternatives at those locations.

Also included in SRAP is an interactive input processor to assist users in creating and modifying input data files. The input processor, through a menu-driven, full-screen editing interface, relieves the user of the "clerical" requirements of data coding, such as card types, card order, data fields, and data formats.

A related report documenting the field testing and implementation experience with SRAP in the State of Iowa is Safety Resource Allocation Programs -- Implementation Technique (Report number FHWA-TS-88-10).

17. Key Words	18. Distribution Statement
Resource allocation, Highway safety, Benefit-cost, Cost-effectiveness, Integer programming, Dynamic programming, Microcomputer software	No restrictions. This document is available to the public through the National Technical Information Services, Springfield, Virginia, 22161.

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of Pages	22. [illegible]
Unclassified	Unclassified	62	A04

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

METRIC (SI*) CONVERSION FACTORS

APPROXIMATE CONVERSIONS TO SI UNITS

LENGTH

Symbol	When You Know	Multiply By	To Find	Symbol
in	inches	2.54	millimetres	mm
ft	feet	0.3048	metres	m
yd	yards	0.914	metres	m
mi	miles	1.61	kilometres	km

AREA

Symbol	When You Know	Multiply By	To Find	Symbol
in²	square inches	645.2	millimetres squared	mm²
ft²	square feet	0.0929	metres squared	m²
yd²	square yards	0.836	metres squared	m²
mi²	square miles	2.59	kilometres squared	km²
ac	acres	0.395	hectares	ha

MASS (weight)

Symbol	When You Know	Multiply By	To Find	Symbol
oz	ounces	28.35	grams	g
lb	pounds	0.454	kilograms	kg
T	short tons (2000 lb)	0.907	megagrams	Mg

VOLUME

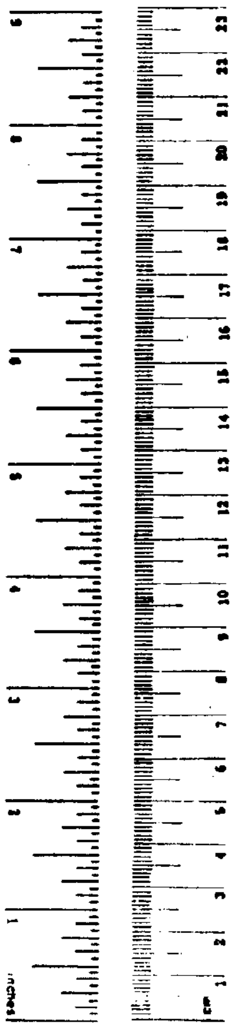
Symbol	When You Know	Multiply By	To Find	Symbol
fl oz	fluid ounces	29.57	millilitres	mL
gal	gallons	3.785	litres	L
ft³	cubic feet	0.0328	metres cubed	m³
yd³	cubic yards	0.0765	metres cubed	m³

NOTE: Volumes greater than 1000 L shall be shown in m³.

TEMPERATURE (exact)

Symbol	When You Know	Multiply By	To Find	Symbol
°F	Fahrenheit temperature	5/9 (after subtracting 32)	Celsius temperature	°C

* SI is the symbol for the International System of Measurements



APPROXIMATE CONVERSIONS TO SI UNITS

LENGTH

Symbol	When You Know	Multiply By	To Find	Symbol
mm	millimetres	0.039	inches	in
m	metres	3.28	feet	ft
m	metres	1.09	yards	yd
km	kilometres	0.621	miles	mi

AREA

Symbol	When You Know	Multiply By	To Find	Symbol
mm²	millimetres squared	0.0016	square inches	in²
m²	metres squared	10.764	square feet	ft²
km²	kilometres squared	0.39	square miles	mi²
ha	hectores (10 000 m²)	2.53	acres	ac

MASS (weight)

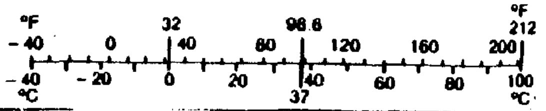
Symbol	When You Know	Multiply By	To Find	Symbol
g	grams	0.0353	ounces	oz
kg	kilograms	2.205	pounds	lb
Mg	megagrams (1 000 kg)	1.103	short tons	T

VOLUME

Symbol	When You Know	Multiply By	To Find	Symbol
mL	millilitres	0.034	fluid ounces	fl oz
L	litres	0.264	gallons	gal
m³	metres cubed	35.315	cubic feet	ft³
m³	metres cubed	1.308	cubic yards	yd³

TEMPERATURE (exact)

Symbol	When You Know	Multiply By	To Find	Symbol
°C	Celsius temperature	9/5 (then add 32)	Fahrenheit temperature	°F



These fac[illegible] requirement of FHWA Order 5190.1A.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

An important step in highway safety planning is the establishment of project priorities. Through prioritizing the safety improvement projects, one attempts to allocate a limited resource or budget among various competing countermeasure alternatives in a way that maximizes total net accident savings. This prioritizing process can be simply stated in a question as: "where and which safety improvement or accident countermeasures should be installed?"

Three computerized methodologies--incremental benefit-cost analysis, integer programming, and dynamic programming--were developed by the Federal Highway Administration (FHWA) in 1983 to aid highway safety planning decisions (and to answer the above question) by prioritizing projects based on their costs and benefits. The three computer models, termed collectively as the "Safety Resource Allocation Programs (SRAP)," were aimed at maximizing total net accident savings under a given budget constraint by selecting the optimal mix of accident locations and the preferred countermeasure alternatives at those locations.

The safety resource allocation models were originally written to run on a mainframe computer. They have been converted to run under the DOS operating system on IBM-PC and compatible microcomputers. In addition, a user-friendly input processor was developed to assist users of the models in the creation and modification of input data files. Both the models and the input processor were subsequently integrated to form a single program package called "SRAP." This document serves as a Users Manual to SRAP.

Manual Organization

The users manual consists of four chapters and two appendixes. In chapter 2, a brief introduction is made of the concept and formulation of the resource allocation models, including a discussion on the input data requirements. Chapter 3 documents the design, operation, and capability of the SRAP program package. The microcomputer hardware needed to run the program and the recommended program setup also are presented. In chapter 4, a tutorial is given which leads the user through both the data editing and model analysis stages using a sample test problem. The chapter concludes with a discussion on outputs from the three optimization models.

CHAPTER 2
SAFETY RESOURCE ALLOCATION MODELS

A recently completed Federal Highway Administration Study of cost-effectiveness methods documented three improved resource allocation models for use in selecting accident countermeasures and locations for highway safety programs. [1] The three models--incremental benefit-cost analysis (INCBEN), integer programming (INTPROG), and dynamic programming (DYNPROG)--contain improved optimization techniques for determining project priority within an available budget. They differ from the commonly practiced simple benefit-cost method in two respects. First, multiple countermeasure alternatives are explicitly formulated and evaluated at each high-accident location and are carried forward to the optimization stage. Secondly, the optimization techniques allow for simultaneous determination of preferred locations and preferred alternatives at those locations to obtain the best system- or program-wide solution. This chapter briefly describes the methodologies and data requirements of the three improved resource allocation models.

Model Overview

In all three models (incremental benefit-cost analysis, integer programming, and dynamic programming), the resource allocation problem is formulated as an optimization problem where one attempts to find the best combination of countermeasure alternatives and accident locations for improvement, under the constraint of a given budget of initial project costs. Selection of locations and the appropriate alternative (which may be "null" or "do nothing") at each location is made on the basis of the present value of annual net benefits and the initial cost of each countermeasure alternative. The same objective function is used in all three formulations: To maximize the present worth of net benefits over all selected alternatives. It is noted that in estimating the net benefits, annual maintenance, operating, and repair costs, in addition to salvage value, should be specifically included as "disbenefits." This ensures that the optimization models maximize these net benefits subject to a constraint on total initial cost (i.e., "budget"). If, on the other hand, annual maintenance, operating, and repair costs and salvage value are to be lumped with the construction cost, the cost constraint would not be the "budget" for initial costs, but would be a budget for present worth of all costs less salvage value.

The present worth of net benefits for each alternative is calculated using the following formula:

$$B = \sum_{i=1}^{SL} [(ACR_i + OUB_i - MC_i - RC_i) / (1 + r)^i] + [SV / (1 + r)^{SL}] \qquad (1)$$

where:

B =	present worth of net benefits over the service life of the alternative;
SL =	service life of the alternative, in years;
r =	discount rate;
ACR_i =	expected reduction in accident costs from employing the alternative, in year i;
OUB_i =	other expected user benefits (savings in vehicle operating and time costs, motorist comfort, etc.) from employing the alternative, in year i;
MC_i =	increase in annual maintenance and operating costs from employing the alternative (excluding RC_i defined below), in year i;
RC_i =	annual increase in repair costs from employing the alternative, in year i; and
SV =	salvage value of the alternative at the end of its life.

The optimization algorithms used in the three models are detailed in reference 1 and the original program documentation, references 2 through 4. The incremental benefit-cost method has the advantage of ranking, from best to worst, all increments of expenditures, instead of specifying the best group of projects for a given budget. The integer programming and the dynamic programming models, on the other hand, employ two common Operations Research techniques in solving the budget allocation problem. All three models have been tested against the simple benefit-cost method and were found to achieve up to 35 to 40 percent improvement in benefits over the simple benefit-cost method. (1) The models also were successfully installed at and operated by a State highway agency during a 1-1/2 year period. That implementation experience of the State is documented in reference 5.

Data Requirements

Input data required by the three safety resource allocation programs are similar and include the following:

- The number of hazardous locations to be considered.
- The overall budget available for safety projects, in dollars.
- The number of countermeasure alternatives to be considered at each location.

and for each alternative at each location:

- The initial construction cost, in dollars.
- The present worth of annual net benefits, in dollars.

Except for the last item--the present worth of net benefits for each alternative which should be estimated as described in the previous section--all other input items can usually be obtained in a straightforward manner. The user is advised to follow his/her agency's existing guidelines and practice in collecting and preparing the input information. Reference 1 can be consulted for additional information on the calculation of reduction in accident cost.

For the integer programming model (INTPROG) only, three additional data items are required in the input stream. They are listed below:

- The lower bound on the total benefits expected, in dollars.

- A print option to show calculation in the algorithm's LP (Linear Program) relaxation.

- A print option to trace the search for the optimal solution.

The two print options generally are not activated, since the output produced by either or both options is typically bulky and is of theoretical interest only. The specification of the total benefits lower bound, however, is strictly required. This bound sets the threshold value above which iteration results are printed in the output. In other words, iterations (rounds of project selection, each resulting in a better solution as project selection approaches the optimum) are not printed until the initial set of selected projects yields at least as many benefits as specified by this lower bound. While the bound, theoretically, can be arbitrarily chosen, prior experience with model INTPROG indicated that setting it at twice the budget level is usually satisfactory, in the sense that the number of iterations will be limited, but not so restricted that project selection cannot attain an optimum solution for the specified budget level. (3) However, if the bound is set too high, then the optimal set of projects for the specified budget level cannot be reached. In this case, model INTPROG must be rerun with a lower value of the bound.

In the original mainframe version of the safety resource allocation models, all input data has to be coded according to a specific, rigid card format. In addition, the input cards need to be arranged in a prescribed sequence. These requirements make the data file creation very difficult and time-consuming if it is to be error free. Under the SRAP package, as will be explained later in the manual, an input processor is available which relieves the user of all the "clerical" requirements of running the models. This enables a user to spend his/her time more productively in data collection and reduction as well as in output analysis.

CHAPTER 3
THE SRAP MICROCOMPUTER SYSTEM

SRAP is an integrated microcomputer program package which contains FHWA's three safety resource allocation programs and an interactive input processor for developing and modifying input data files. The package, with its menu structure and full-screen data entry feature, is extremely friendly and easy to use. No prior microcomputer experience is required from the user.

This chapter details the program design, operation, and capability of SRAP. The microcomputer hardware required to run the program and the recommended program setup also are presented.

Program Design

Basically, the SRAP system is composed of four independent programs working under an integrated, coordinated environment. The four programs are: The input processor program and the three safety resource allocation programs (incremental benefit-cost analysis, integer programming, and dynamic programming). Under the system, a user develops a data set, performs an analysis using any of the three resource allocation models, modifies the data set, and repeats the analysis as desired, all without leaving the system. The "shell" environment provided by SRAP shields the user from the host operating system of the microcomputer. This enables a user with no knowledge of DOS (Disk Operating System) to learn and operate the package with minimal effort and in minimal time. In addition, the SRAP system is self-contained. Once one enters the system, all data manipulation and model analysis tasks are performed from within the system. The SRAP shell is in control at all times until the user decides to quit the system, at which time the control is returned to the host operating system or DOS. An overview of the SRAP system structure is given in figure 1. The operation of SRAP and the flow between the component programs are discussed later.

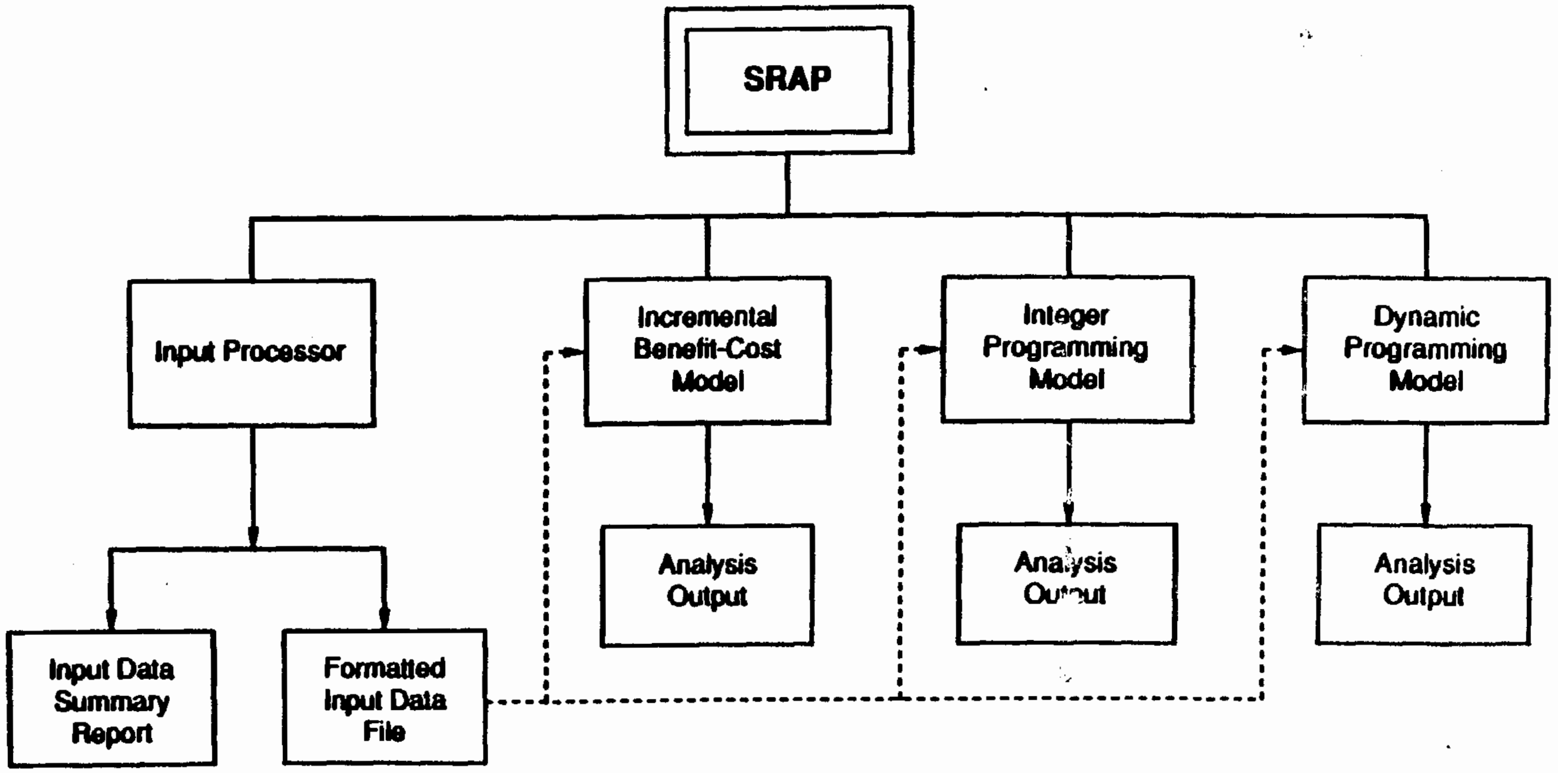

SRAP
Input Processor
Incremental Benefit-Cost Model
Integer Programming Model
Dynamic Programming Model
Analysis Output
Analysis Output
Analysis Output
Input Data Summary Report
Formatted Input Data File

Figure 1. SRAP program structure overview.

The three resource allocation models were originally written in FORTRAN for the mainframe computer. They were converted to run under the DOS operating system on IBM-PC and compatible microcomputers. Since the microcomputer version of the models is essentially identical to the mainframe version, the same input data file will produce exactly the same output. The conversion effort, however, did modify the FORTRAN source codes of the three models to conform to the newer ANSI FORTRAN-77 standard. In addition, enhancement was made to allow the models to accommodate varying problem sizes through an added PARAMETER statement for redimensioning arrays. The resulting resource allocation models not only maintain all existing capabilities of the mainframe version, but are significantly more portable and accessible to potential users.

The interactive input processor was developed to aid the user of the models in creating and modifying input data files. It relieves the user of the "clerical" requirements of data coding, such as card types, card order, data fields, and data formats. These input coding schemes are totally transparent to the user. The input processor was programmed in Turbo Pascal and has the following features:

1. User-friendly interface.

   The input processor has a user-friendly interface through the use of "full-screen editing." This method presents the user with a well-labelled screen template that contains blanks (or input fields) to be filled with data. Free movement between data fields and immediate modification of data before acceptance by the program also are provided by the method.

2. Capability of both creating and modifying data files.

   In addition to creating new data files interactively, the input processor allows a user to make changes to existing data files without entering all of the data again. This is an important timesaving feature.

3. Compatibility to mainframe data files.

   The input processor is able to read and write data files in the format of the existing mainframe programs. For large problems with many accident locations, if the user desires, he/she may upload or transfer the data file to the mainframe and run the resource allocation programs there.

4. Multiwindow, multicolor display.

   The program is designed on multiwindow technology which is able to clearly display the hierarchy of the input process on the screen. It can work on both color and monochrome monitors.

5. Convenient summary report.

   The program can generate an input data summary report in a more readable format than the output echo from the resource allocation models. It also provides additional identification fields for the user to store such information as the name of the user, run ID number, date of run, etc. This identification information is all printed on the summary report.

6. Error checking.

   All input data entered by the user are checked against the allowable range and type of data for each input field. Since the input processor does all the actual coding of the input data file for the analysis models, the chances of having typographical errors are minimized.

7. File management capability.

   A comprehensive file management system is provided in the input processor. A user can retrieve and save data files from within the input processor, on any drive or path of the microcomputer system. The program is intelligent in that it warns against such possible user errors as overwriting an existing data file or exiting the program without saving the data. Handy utilities such as displaying directory and changing drive or directory path specification also are provided in the input processor.

The current version of SRAP has the capability of analyzing up to 150 accident locations per given run. Each location can contain up to seven accident countermeasure alternatives. The only exception is the dynamic programming (DYNPROG) model, which is limited to no more than 85 accident locations per run, due to the larger memory requirement of the DYNPROG model. In case the user needs to execute the DYNPROG model with more than 85 locations, a mainframe version may be used. This limitation should pose no problem for the typical user, as the three models generally choose almost the same set of projects. In addition, the total benefits from each of the models are usually within about 1/2 of 1 percent of each other. (1)

Program Operation

Once entering SRAP, the user simply responds to questions or enters data into fields on the screen. The program is run as described in the Program Execution section of chapter 4. The user initially is presented with a menu. Menu options are selected by moving the highlight bar to the desired menu choice and hitting the carriage return. The program will branch to the desired section of the program. The input processor and each of the three resource allocation models are the major options available in this menu.

Inside the input processor, data entry screens are displayed which contain descriptions of the data to be entered and data fields. The data field will usually contain the current value--either that being modified or a default value. The program automatically moves between the data fields on the screen. The current data field is always highlighted. The user enters only a carriage return to accept the value shown, or enters a new value over the old one followed by a carriage return. At any time, the user can skip or move around the fields on the screen by depressing the cursor keys (up, down, left, and right arrows). The movement between different data entry screens are facilitated through the "PgUp" and "PgDn" keys. At all times, the command keys or options available to the user are displayed at the bottom of the screen. A full description of the input processor can be found in chapter 4.

Two types of output are generated by the input processor. One is the data summary report, while the other is a fixed format data file for running any of the three resource allocation models. After exiting the input processor and returning to the main menu, the user can choose to execute any one model by answering the program prompts for input and output files. The execution begins and produces an output file on disk, screen, or printer, as the user requests. The program flow is as depicted in figure 1. A sample session through SRAP is given in chapter 4.

Equipment Needs

The microcomputer hardware required to operate the SRAP program is listed in table 1. Except for the higher memory need (640 K), all other requirements are easily found on typical microcomputer systems. Memory need was increased due to the shell design, which accommodates the presence of both the input processor program and the safety resource allocation models.

As a reference, run time statistics for the SRAP package were collected and are presented in table 2. These statistics are the actual execution times (for two sample data files prepared by the input processor) for each of the safety resource allocation models. The two sample problems tested were of different sizes. The large problem contained 80 high-hazard locations and a total of 146 countermeasure alternatives spread among the locations. The small problem had 24 accident locations and 47 countermeasure alternatives. The test computer was a Kaypro Professional Computer, model PC-10, which is IBM-PC compatible but has a higher CPU speed of 8.0 MHz.

Table 1. Microcomputer resource requirements for software use.

Component	Requirement
Computer	IBM-PC/XT/AT or compatible
Disk Operating System (DOS)	Version 2.0 or higher
Main Memory	640 K
Disk Drive	One 5.25-inch floppy drive or hard-disk drive
Monitor	Monochrome or color
Printer	Capable of 132-column printing and compatible with the computer

Table 2. SRAP run time statistics.

	Problem Size	
Model	80 Locations with 146 Alternatives	24 Locations with 47 Alternatives
INCBEN	40 sec.	9 sec.
INTPROG	4 min. 10 sec.	22 sec.
DYNPROG	17 min. 55 sec.	2 min. 28 sec.

As can be seen in table 2, model DYNPROG generally required more time than the other two. However, even for the large problem with 80 locations, the run time was still reasonable and acceptable. Model INCBEN required the least run time and caused practically no wait for the user at all. It is noted that these run times did not include printing the output. Rather, the output was routed to the hard disk during the tests. For users with a more advanced computer such as the IBM-AT class machine or a computer equipped with the math coprocessor (8087 or 80287 chip), the run times can be reduced further. In general, once the user has all the input information prepared, a typical size problem (approximately 50 locations) can be entered through the input processor and executed in less than 1 hour, including printing the output.

Program Distribution and Setup

The SRAP package is distributed on a single 360-K floppy diskette. The diskette contains all necessary program files in executable form together with a sample data file. This section describes the contents of each file and how the user can set up a working diskette for use in running SRAP.

The following files are present on the distribution diskette:

1. SRAP.COM: execution file for the SRAP shell and the interactive input processor.
2. INCBEN.EXE: execution file for the analysis program, incremental benefit-cost technique (INCBEN).
3. INTPROG.EXE: execution file for the analysis program, integer programming technique (INTPROG).
4. DYNPROG.EXE: execution file for the analysis program, dynamic programming technique (DYNPROG).
5. MENU_01.TPL
   S_INPUT.TPL
   G_01.TPL: three screen template files for the input processor.

6. MENU_01.COL S_01.COL G_01.COL:	three screen attribute files for color monitors.
7. MENU_01.MON S_01.MON G_01.MON:	three screen attribute files for monochrome monitors.
8. TEST.SAF:	a sample input data file.

The SRAP package is supplied on a DOS formatted diskette that does not contain DOS itself. Therefore, to create a working diskette that is bootable, the user must copy the appropriate files to a bootable diskette. The suggested method follows. All user input is underlined for clarity. The first setup is for microcomputers with two diskette drives. The second is for those with at least one fixed disk and one floppy drive.

Setup 1. Two Drive System

1. Booting the system. Place the DOS operating system diskette in drive A and turn on the computer. A DOS diskette should be available with a user's microcomputer. After the computer boots, it may ask for time and date (depending on configuration) and will eventually display the system prompt:

   ```
   A>
   ```

2. Formatting a blank diskette. Place a blank, unformatted diskette in drive B and type the following:

   ```
   A>FORMAT B:/S
   ```

   This command will format the diskette drive B and transfer the system files, creating a bootable diskette.

3. Copying SRAP files. Replace the DOS diskette in drive A with the SRAP distribution diskette. Leave the just formatted diskette in drive B. Then type the following:

   ```
   A>COPY *.* B:
   ```

   which will copy all files from the distribution diskette to the bootable working diskette.

4. Backing up the working diskette. This step is optional. The user may wish to back up the working copy of the diskette in case

anything happens to the working copy. The backup may be created by following steps 1 through 3 again, or by using the DISKCOPY command, as described in the DOS users manual. In any case, the backup and the original copy should be stored in a safe place.

Setup 2. Fixed Disk System

This setup assumes that the SRAP package will be stored on the fixed disk. If the user plans on running the program from the floppy drive, then the instructions for the first setup should be followed, as may suit the exact system configuration.

1. Booting the system. Turn on the computer. The fixed disk must be partitioned to operate under DOS. After the computer boots, it may ask for time and date (depending on configuration) and will eventually display the system prompt:

   ```
   C>
   ```

2. Creating sub-directory. This step is optional. However, most fixed disk users will probably want to create (or make) a sub-directory for SRAP. To do this, from the root directory, or lower level sub-directory of the user's choice (see DOS manual for more information about sub-directory structures), type

   C><u>MD SRAP</u> (or whatever name the user wishes to use in lieu of "SRAP", assuming from the root directory), then

   C><u>CD SRAP</u> (to "C"hange to that "D"irectory).

3. Copying SRAP files. After inserting the distribution diskette in drive A, type the following command to copy all the files from the SRAP distribution diskette to the current directory on the fixed disk:

   C><u>COPY A:*.*</u>

4. Backing up the SRAP diskette. The user should back up the distribution diskette in case anything happens to the original copy. The backup may be created by using the DISKCOPY routine in the DOS user's manual. The backup should be stored in a safe place.

After completing either of the above setup procedures, the user is ready to run SRAP. The program execution is presented in chapter 4.

CHAPTER 4
A SRAP TUTORIAL

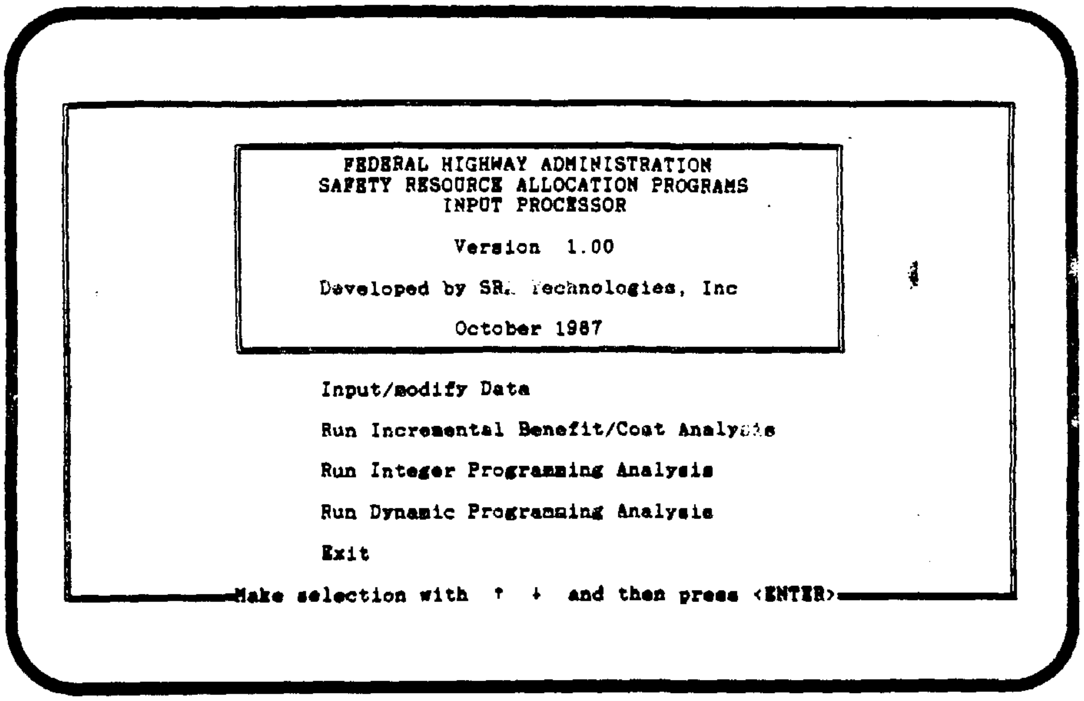
This chapter details the operation of the SRAP microcomputer program package. A sample session is presented which leads the user through both the data editing and model analysis stages of a sample test problem. Output from the optimization models is then discussed.

Program Execution

To run SRAP, put the working diskette (prepared as described earlier) in the default drive and type "SRAP" followed by a carriage return (the <Enter> key). For users with a fixed disk, no diskette insertion is necessary, but be sure the default (i.e., current) drive or sub-directory is the one that contains the SRAP package. All files except the sample data set from the distribution diskette must be present on the default drive to execute the program. After entering the "SRAP" command, the user is presented with a screen as shown in figure 2.

The screen contains a program logo showing the version number of the program and the SRAP main menu. The boxed logo area is colored blue if the user is using a color monitor. From this point on, the user is completely under the SRAP shell and does not interact with DOS at all. To exit SRAP, simply choose the "Exit" option of the menu by moving the highlight bar to rest on the option and following with a carriage return.

Two comments are in order here regarding the two sets of screen attribute files (file sets 6 and 7) described earlier in chapter 3. SRAP was designed to automatically recognize a user's display type through the appropriate screen buffer address in memory. If a user's system contains the color graphics display, file set 6 (with the ".COL" ex[illegible]ion) will be used by SRAP to display colored screens. On the other ha[illegible] file set 7 (with the ".MON" extension) will be chosen by SRAP for sy[illegible]s with monochrome



```
                FEDERAL HIGHWAY ADMINISTRATION
              SAFETY RESOURCE ALLOCATION PROGRAMS
                       INPUT PROCESSOR

                        Version  1.00

           Developed by SRA Technologies, Inc

                        October 1987

           Input/modify Data

           Run Incremental Benefit/Cost Analysis

           Run Integer Programming Analysis

           Run Dynamic Programming Analysis

           Exit
      Make selection with  ↑  ↓  and then press <ENTER>
```

Figure 2. Terminal screen: SRAP main menu.

display. Therefore, if the user can be certain that SRAP will be run on only one (color or monochrome) particular type of system, only the corresponding set of screen attribute files (either set 6 or 7) needs to be present on the SRAP working diskette. The other set can be deleted.

Secondly, for a certain class of composite monitors without color capability, SRAP may choose the color screen attribute files (set 6) and display screens in different shades. Although the program still functions, the screens' readability may be degraded. In this case, the user may want to remove the screen shading by swapping the contents of the two sets of screen attribute files. This may be done by using the following DOS commands:

<u>DEL *.COL</u>	(to "DEL"ete the three original color screen attribute files), and
<u>REN *.MON *.COL</u>	(to "REN"ame the three monochrome attribute files as the new color screen attribute files).

In any case, the original sets of color and monochrome screen attribute files should be kept safely on the distribution or the backup diskette.

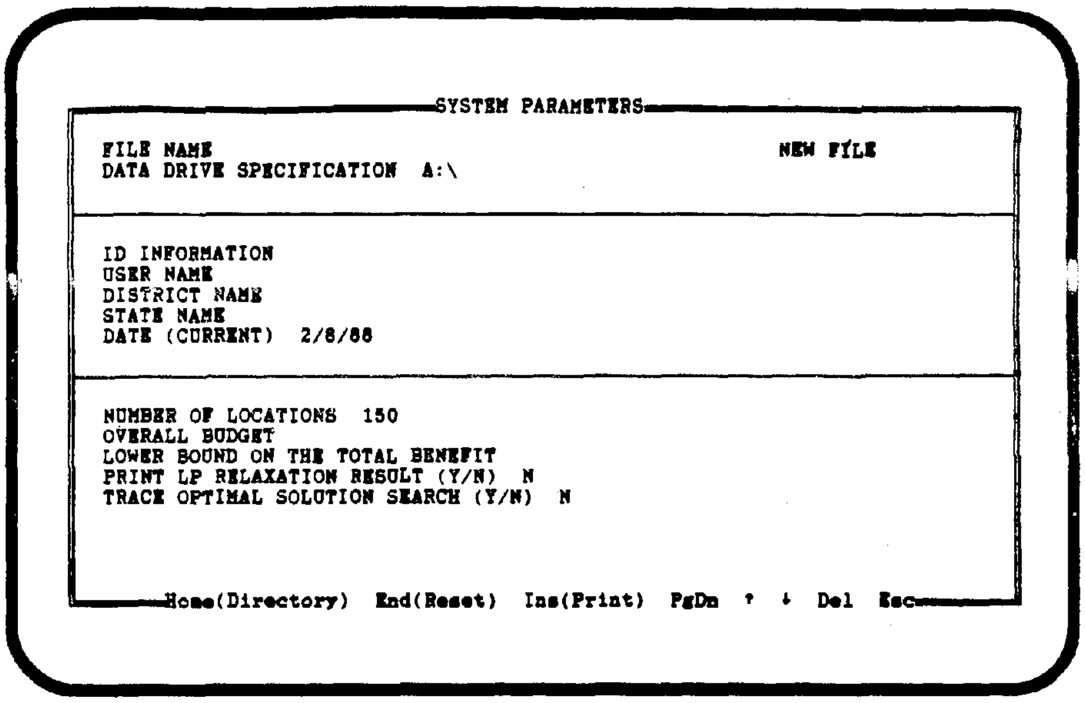
Sample Session

This section presents a tutorial session to familiarize users with the operation of SRAP. The tutorial also allows a user to identify and study the necessary input data requirements before running the program. The session simulates the modification of the sample data set, TEST.SAF, supplied alon with the package. All data entry screens are shown in sequence as th modifications are being made, and a new data file, TEST1.SAF, is create These terminal screens are reproduced exactly as they appear at some poin during execution. Color and boldface, however, is not shown.

Once entering the "SRAP" command, the first screen a user sees is the main menu as shown in figure 2. Initially, the highlight bar rests on the first option of the menu, "Input/modify Data." This is the option that contains the input processor for data creation and editing. In the menu, there are three other options for activating each of the three safety resource allocation models for analysis runs, and one last option for exiting the SRAP program.

As indicated in the last line of the screen display, the user makes a menu selection by using the up or down arrow key to highlight his choice of menu option, followed by an <Enter> key. Almost every screen in the program has this "help line" at the bottom of the screen. Typically, the various command keys available to the user for that particular screen are displayed in the help line.

Continuing the sample session, the user chooses the "Input/modify Data" option by simply hitting the <Enter> key. Figure 3 shows the "System Parameters" screen displayed immediately after entering the input processor.



```
                              SYSTEM PARAMETERS
 FILE NAME                                                      NEW FILE
 DATA DRIVE SPECIFICATION  A:\

 ID INFORMATION
 USER NAME
 DISTRICT NAME
 STATE NAME
 DATE (CURRENT)  2/8/88

 NUMBER OF LOCATIONS  150
 OVERALL BUDGET
 LOWER BOUND ON THE TOTAL BENEFIT
 PRINT LP RELAXATION RESULT (Y/N)  N
 TRACE OPTIMAL SOLUTION SEARCH (Y/N)  N

     Home(Directory)  End(Reset)  Ins(Print)  PgDn  ↑  ↓  Del  Esc
```

Figure 3. Terminal screen: System parameters.

There are only two levels of screens inside the input processor. The "System Parameters" screen is in the first level. This screen contains general data that are applicable to the entire problem and are not accident location specific. The data are divided into three parts.

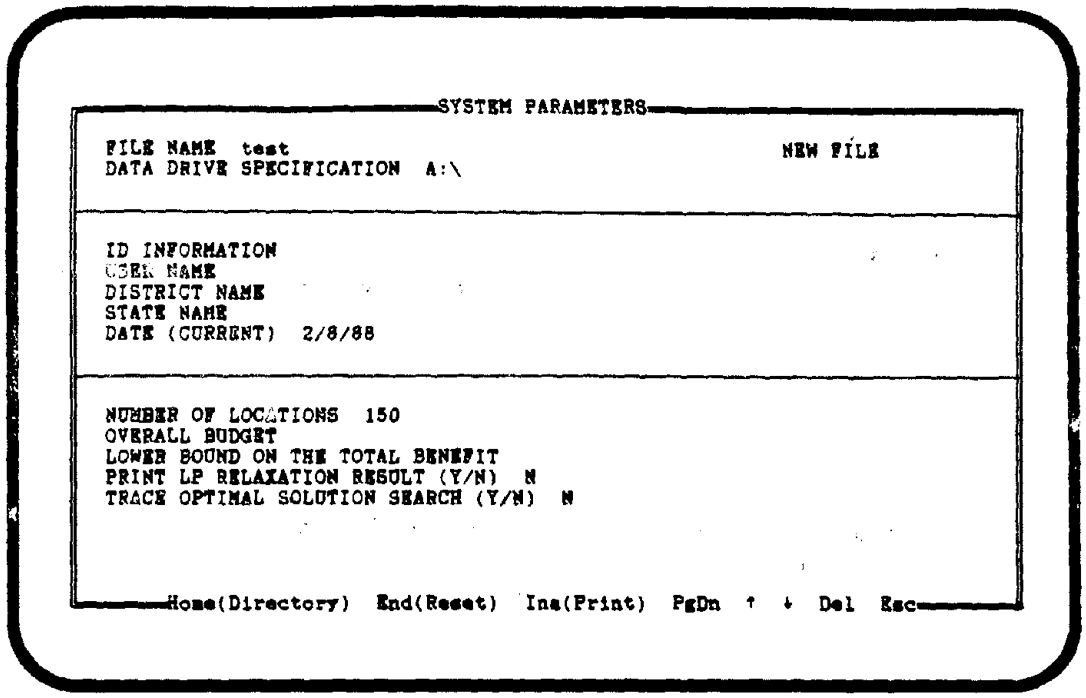
The first part contains two data fields: the file name and the data drive specification. The second part is for users to record identification information unique to the data set or run. Data required for each of the fields are self-explanatory. The third part contains important input data regarding the total number of accident locations to be studied and the overall budget in dollars. The last three fields in this part are specific to the integer programming model (INTPROG). Their use has been described in the "Data Requirements" section of chapter 2.

As can be seen in figure 3, most of the data fields are blank at this point, but several fields do have default (i.e., preset) values. The data drive specification field contains the "A" drive label as the SRAP working diskette was executed from drive A. Had the user booted the program from a subdirectory on drive C, the field might display "C:\SRAP\." The date field contains "2/8/88," which was the current date on the microcomputer system. SRAP automatically accesses the computer's internal clock to obtain this information. A user running the sample session sees his/her current date displayed if the system clock is set properly. Three data fields in the third part of the screen contain default values. The number of accident locations is set at 150, which is the maximum number that can be accommodated by the current version of SRAP. The two print options for model INTPROG are set as "N."

Normally, if a user is creating a new data file, he/she starts entering information by completing the data fields. The fact that a new data set is being created is indicated by the "NEW FILE" notice given at the end of the file name field. In this sample session, however, an existing data set, TEST.SAF, from the distribution diskette will be retrieved and modified.

The SRAP program has its own convention in naming data files. All data sets created or accessed by SRAP will have a file name of up to eight characters plus the ".SAF" extension. The ".SAF" extension is strictly mandatory in order for the program to distinguish a SRAP data file from other files residing on the same drive. Inside the SRAP shell, however, there is no need for the user to attach the ".SAF" extension in any file name specification. This is done automatically by the program.

Continuing the sample session, the user sees on the screen that a highlight bar rests on the file name field upon entering the screen (although the bar can not be shown in figure 3). To retrieve an existing data file from the current drive (drive A, in this case), the user enters its file name in the field. He/she types "test" as shown in figure 4 and hits the <Enter> key. Note that no file name extension is necessary and, in fact, any extension entered will not be accepted in the file name field.

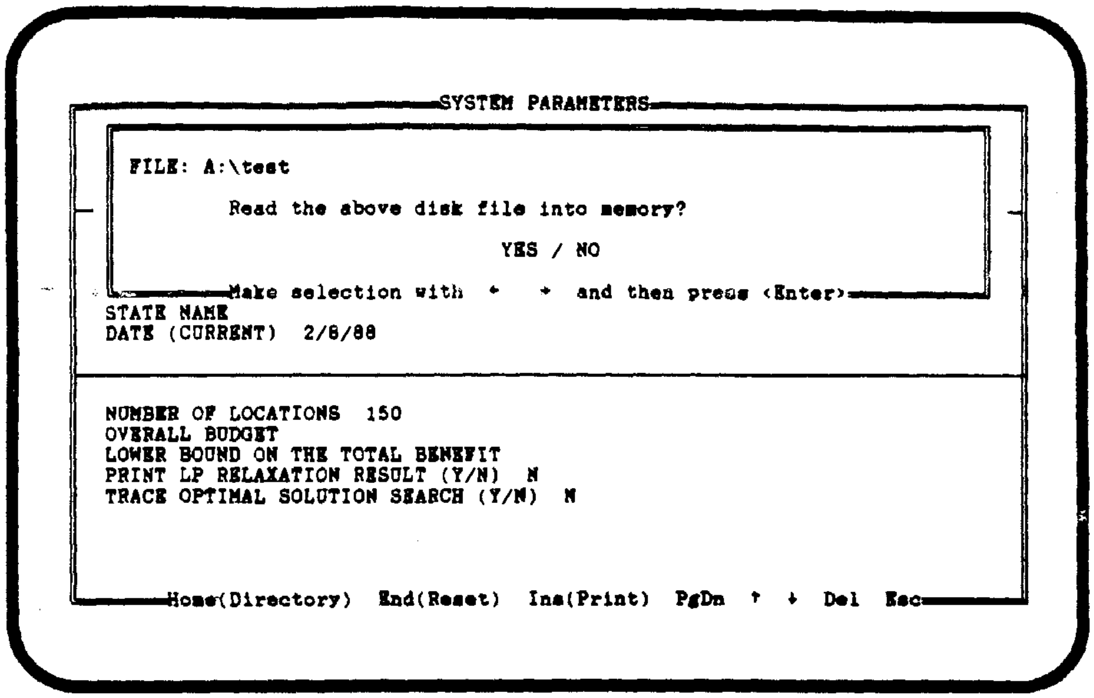


```
SYSTEM PARAMETERS
FILE NAME  test                                              NEW FILE
DATA DRIVE SPECIFICATION  A:\

ID INFORMATION
USER NAME
DISTRICT NAME
STATE NAME
DATE (CURRENT)  2/8/88

NUMBER OF LOCATIONS  150
OVERALL BUDGET
LOWER BOUND ON THE TOTAL BENEFIT
PRINT LP RELAXATION RESULT (Y/N)  N
TRACE OPTIMAL SOLUTION SEARCH (Y/N)  N

Home(Directory)  End(Reset)  Ins(Print)  PgDn  ↑  ↓  Del  Esc
```

Figure 4. Terminal screen: Entering input file name.

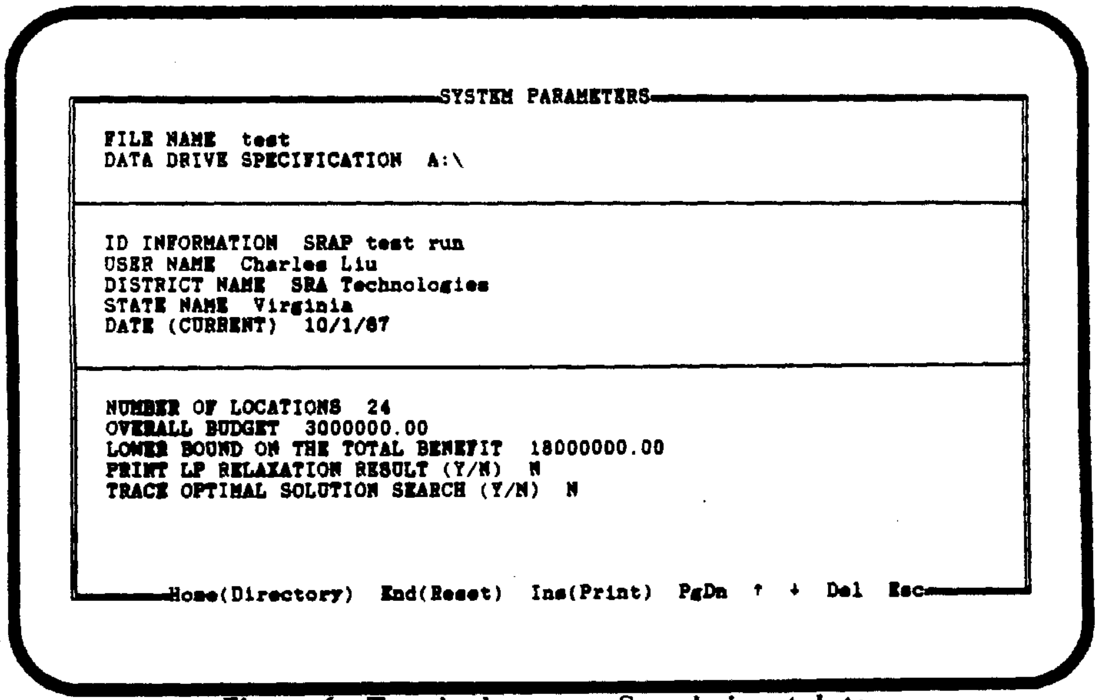
Upon accepting the "test" file name, a pop-up window appears, which overlays part of the original screen as shown in figure 5. This indicates SRAP has checked the current drive A for the existence of the file named "TEST.SAF" and that it is now confirming with the user the retrieval of the data set. The user chooses the correct action by using the left or right arrow key, or he/she simply types "Y" or "N" from the keyboard to answer the question. In this case, the user answers "Yes" to continue.

After reading the input file, the "System Parameters" screen is restored as in figure 6. Note that all data fields now contain data from the sample file "TEST.SAF". Also note that the "NEW FILE" tag has disappeared from the end of the file name field, because SRAP now holds the old or existing file "TEST.SAF" in memory. If, for example, the user had specified an incorrect file name "test1" to retrieve, the program would simply accept the name "test1" and continue to display the "NEW FILE" notice. No file retrieval would occur at all. The checking performed by SRAP not only ascertains the existence of



```
SYSTEM PARAMETERS

FILE: A:\test

        Read the above disk file into memory?

                        YES / NO

    Make selection with  ←  →  and then press <Enter>
STATE NAME
DATE (CURRENT)  2/8/88

NUMBER OF LOCATIONS  150
OVERALL BUDGET
LOWER BOUND ON THE TOTAL BENEFIT
PRINT LP RELAXATION RESULT (Y/N)  N
TRACE OPTIMAL SOLUTION SEARCH (Y/N)  N

Home(Directory)  End(Reset)  Ins(Print)  PgDn  ↑  ↓  Del  Esc
```

Figure 5. Terminal screen: File reading check.



```
SYSTEM PARAMETERS

FILE NAME  test
DATA DRIVE SPECIFICATION  A:\

ID INFORMATION  SRAP test run
USER NAME  Charles Liu
DISTRICT NAME  SRA Technologies
STATE NAME  Virginia
DATE (CURRENT)  10/1/87

NUMBER OF LOCATIONS  24
OVERALL BUDGET  3000000.00
LOWER BOUND ON THE TOTAL BENEFIT  18000000.00
PRINT LP RELAXATION RESULT (Y/N)  N
TRACE OPTIMAL SOLUTION SEARCH (Y/N)  N

Home(Directory)  End(Reset)  Ins(Print)  PgDn  ↑  ↓  Del  Esc
```

Figure 6. Terminal screen: Sample input data.

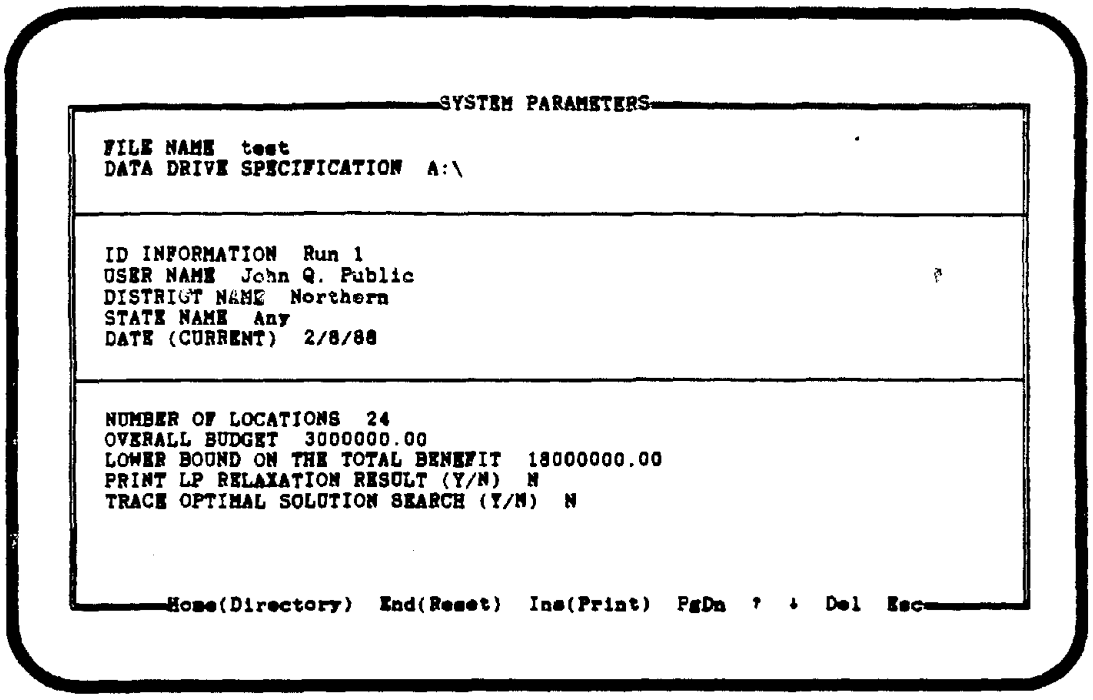
a data set, it also prevents the user from accidentally assigning duplicate file names.

The user is now ready to modify the sample data file. To do this, he/she moves the highlight bar to the desired data field and enters the new information. The movement between data fields is accomplished through the up and down arrow keys. These two keys appear in the "help line" at the bottom of the screen. There are six additional command keys in the help line. They will be discussed as the tutorial progresses.

In this sample session, the user will only modify the data fields in the second part of the screen. First, he/she uses the arrow key to highlight the first field (ID Information) in this part. Then, he/she types in a new ID of the user's choice (say, "Run 1"). Note that as soon as the first character is entered, the old entry "SRAP test run" disappears entirely. Another way to delete the old entry in a data field is to use the command key <Del>. This key will clear the existing data and leave a blank data field. After entering the new data, the user should always depress the <Enter> key to signal the completion of a data item. Unless the entry contains an error, it will be accepted by the program and the highlight bar will automatically move down to the next data field. The user proceeds to modify the other data fields in this part.

The data field "Date" contains a special feature. If the user types a "C" to this field, the program will access the system clock and display the system date automatically. Of course, the user can enter the date manually, as before. After the modifications of all data fields in the second part, the screen may appear as in figure 7.

Only the second part of the screen was modified in this sample session because the information contained there does not affect the actual analysis of the resource allocation models. This facilitates the later discussion on model outputs. The data editing process described here, however, applies to all data fields and should serve as an excellent illustration to the users.

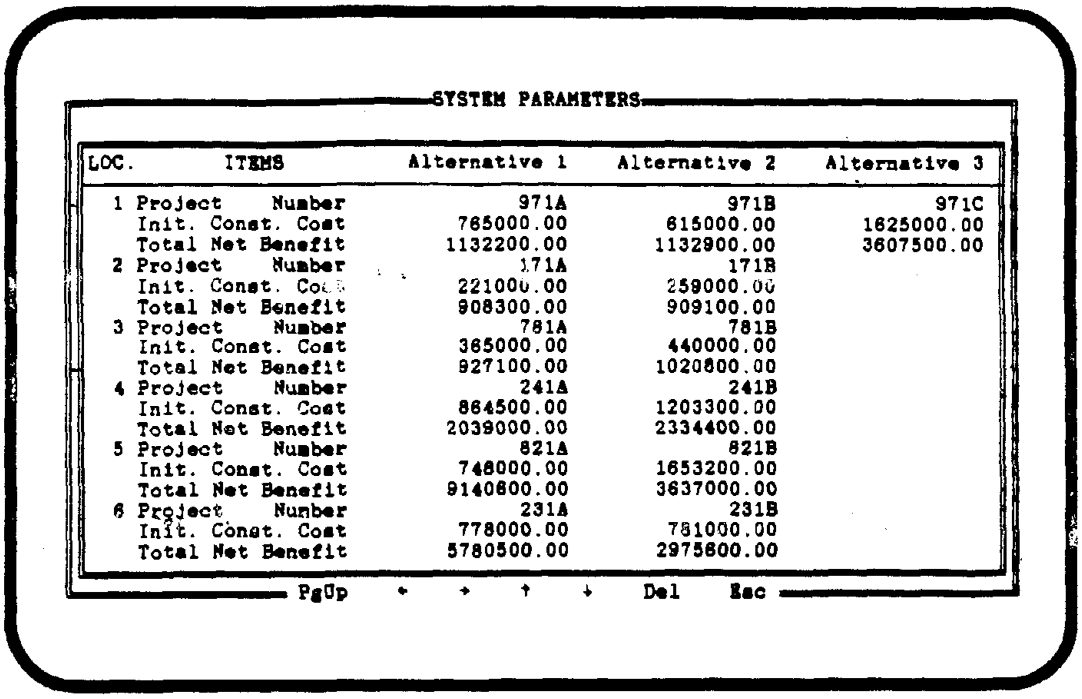


```
SYSTEM PARAMETERS
FILE NAME  test
DATA DRIVE SPECIFICATION  A:\

ID INFORMATION  Run 1
USER NAME  John Q. Public
DISTRICT NAME  Northern
STATE NAME  Any
DATE (CURRENT)  2/8/88

NUMBER OF LOCATIONS  24
OVERALL BUDGET  3000000.00
LOWER BOUND ON THE TOTAL BENEFIT  18000000.00
PRINT LP RELAXATION RESULT (Y/N)  N
TRACE OPTIMAL SOLUTION SEARCH (Y/N)  N

Home(Directory)  End(Reset)  Ins(Print)  PgDn  ↑  ↓  Del  Esc
```

Figure 7. Terminal screen: Modified input data.

Before continuing the sample session, it is noted that each of the data fields in the input processor was programmed to accept only one type (numeric or alphameric) of data. For example, the "Number of Locations" field can only accept numerical values. If a user tries to enter any alphabet, it will be rejected.

Since the user has completed the modification of the "System Parameters" screen, the next step is to modify the data pertaining to each countermeasure alternative at all accident locations. Hitting the <PgDn> key at this point activates the second level of screen as shown in figure 8. The second level screen contains the project number, initial construction cost (in dollars), and present worth of total annual net benefits (in dollars) of all countermeasure alternatives at all accident locations. The screen appears as a "window" overlaying the first level "System Parameters" screen.

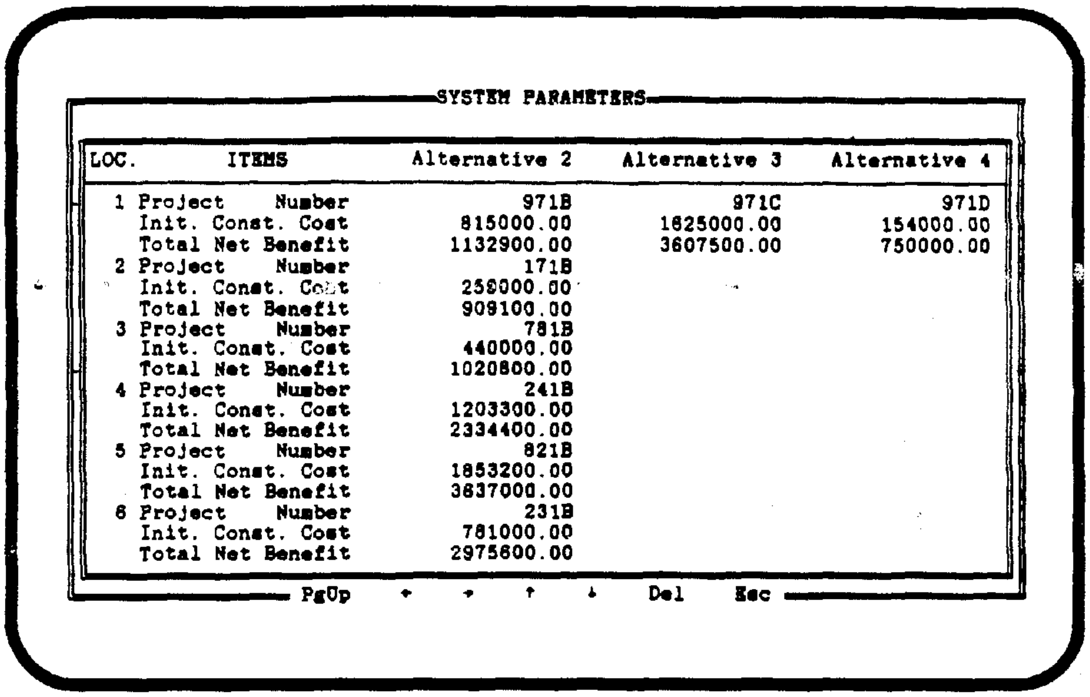


```
                         SYSTEM PARAMETERS
LOC.        ITEMS           Alternative 1   Alternative 2   Alternative 3

  1 Project     Number               971A            971B            971C
    Init. Const. Cost           765000.00       615000.00      1625000.00
    Total Net Benefit          1132200.00      1132900.00      3607500.00
  2 Project     Number               171A            171B
    Init. Const. Cost           221000.00       259000.00
    Total Net Benefit           908300.00       909100.00
  3 Project     Number               781A            781B
    Init. Const. Cost           365000.00       440000.00
    Total Net Benefit           927100.00      1020800.00
  4 Project     Number               241A            241B
    Init. Const. Cost           864500.00      1203300.00
    Total Net Benefit          2039000.00      2334400.00
  5 Project     Number               821A            821B
    Init. Const. Cost           748000.00      1653200.00
    Total Net Benefit          9140800.00      3637000.00
  6 Project     Number               231A            231B
    Init. Const. Cost           778000.00       781000.00
    Total Net Benefit          5780500.00      2975600.00
              PgUp    ←     →     ↑     ↓     Del    Esc
```

Figure 8. Terminal screen: Countermeasure alternatives data, display 1.

Data fields in the window are organized in the form of a matrix. Starting from the top, each accident location occupies three rows for the three data items: project number, cost, and benefit, respectively. The countermeasure alternatives at each location are represented in columns across the matrix. Initially, the top left field "971A" (the project number for Alternative 1 at Location 1) is highlighted. The user may use all four arrow keys (left, right, up, and down) to move the highlight bar to any matrix "cell" he/she wants to modify. The window has been designed to "scroll" in all four directions as directed by the arrow keys. For example, by hitting the right arrow key three times, the window will appear as in figure 9, showing alternatives 2 through 4.

In essence, this data screen is organized as a conventional "spreadsheet" with a total of seven columns for up to seven alternatives at each location. The number of rows in the spreadsheet depends on the number of accident locations specified in the first level screen. This large spreadsheet, however, is viewed, at any moment, through a window containing six locations and three

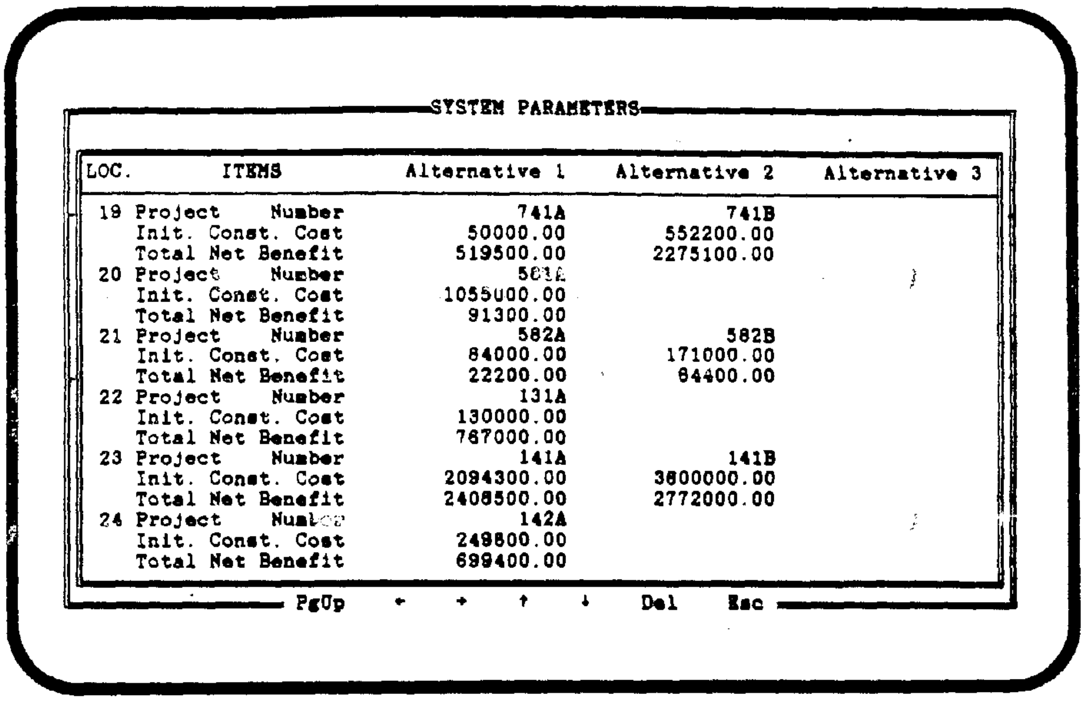


```
SYSTEM PARAMETERS
LOC.        ITEMS             Alternative 2    Alternative 3    Alternative 4

  1 Project     Number                 971B             971C             971D
    Init. Const. Cost             815000.00       1625000.00        154000.00
    Total Net Benefit            1132900.00       3607500.00        750000.00
  2 Project     Number                 171B
    Init. Const. Cost             259000.00
    Total Net Benefit             909100.00
  3 Project     Number                 781B
    Init. Const. Cost             440000.00
    Total Net Benefit            1020800.00
  4 Project     Number                 241B
    Init. Const. Cost            1203300.00
    Total Net Benefit            2334400.00
  5 Project     Number                 821B
    Init. Const. Cost            1853200.00
    Total Net Benefit            3837000.00
  6 Project     Number                 231B
    Init. Const. Cost             781000.00
    Total Net Benefit            2975600.00
                PgUp    ←    →    ↑    ↓    Del    Esc
```

Figure 9. Terminal screen: Countermeasure alternatives data, display 2.

alternatives each. The scrolling capability enables the user to zoom in on any portion of the spreadsheet. Another example of the window is given in figure 10, which displays the data for locations 19 through 24. Since the sample data set contains 24 locations, no more downward scrolling will be allowed at this point.

The specification of project number is arbitrary and any unique alphameric designation will suffice. However, the numbering scheme used in the sample data, as suggested in reference 2, is strongly recommended. In this scheme, one assigns a different three-digit integer to each location, e.g., 101, 102, 103, ... Then one assigns a unique letter to each alternative at each location, e.g., A, B, C, ... The concatenation of a location number and alternative letter becomes a project number, e.g., 101A, 101B, 101C, ..., 102A, 102B, 102C, ...

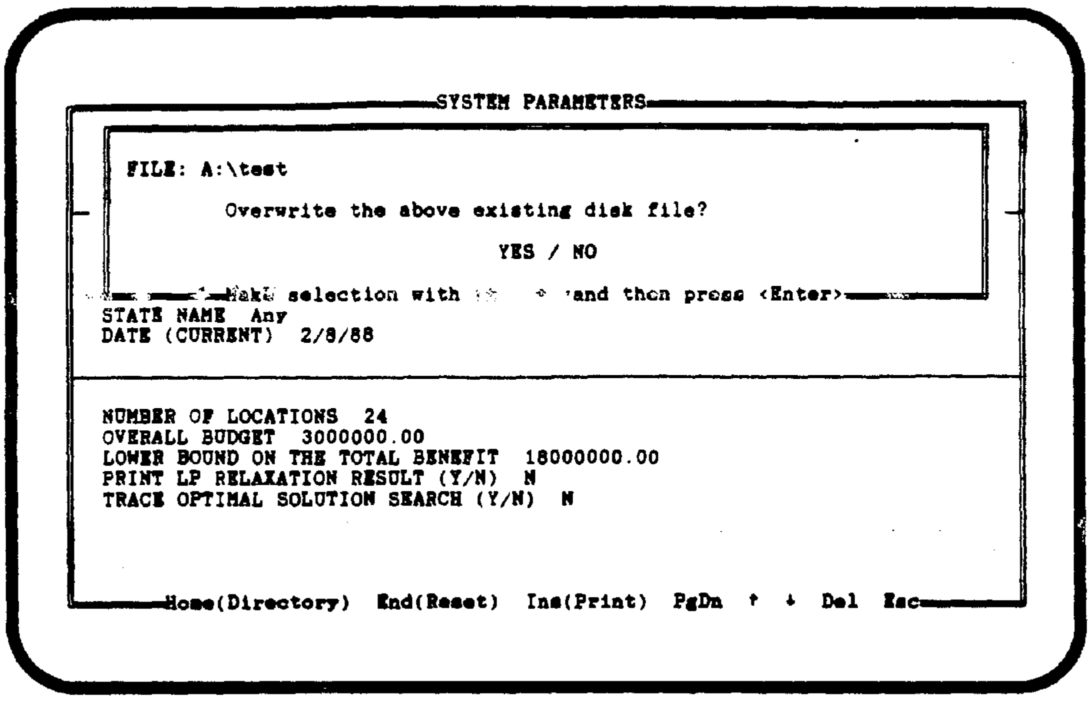


```
                         SYSTEM PARAMETERS
LOC.        ITEMS              Alternative 1   Alternative 2   Alternative 3

 19 Project     Number                  741A            741B
    Init. Const. Cost               50000.00       552200.00
    Total Net Benefit              519500.00      2275100.00
 20 Project     Number                  581A
    Init. Const. Cost             1055000.00
    Total Net Benefit               91300.00
 21 Project     Number                  582A            582B
    Init. Const. Cost               84000.00       171000.00
    Total Net Benefit               22200.00        84400.00
 22 Project     Number                  131A
    Init. Const. Cost              130000.00
    Total Net Benefit              767000.00
 23 Project     Number                  141A            141B
    Init. Const. Cost             2094300.00      3800000.00
    Total Net Benefit             2408500.00      2772000.00
 24 Project     Number                  142A
    Init. Const. Cost              249800.00
    Total Net Benefit              699400.00

              PgUp    ←     →     ↑     ↓    Del    Esc
```

Figure 10. Terminal screen: Countermeasure alternatives data, display 3.

No data modification will be performed for this level of screen in the sample session. In actual use, the user may use the same editing or modification procedure as described earlier. Since there are only two levels of screen in the input processor, the user has completed the data modification task for the sample data set. To return to the first level, he/she simply hits the <PgUp> key. The "System Parameters" screen similar to that of figure 7 is restored.

The use of other command keys in the help line at the bottom of the "System Parameters" screen will now be explained. The <Esc> key is used to exit the input processor in order to return to the main menu. In certain situations, however, the program might prompt the user for verification of file name and warn against leaving the input processor without saving an edited data set. As a demonstration, the user should depress the <Esc> key at this point. A pop-up window will appear as shown in figure 11.

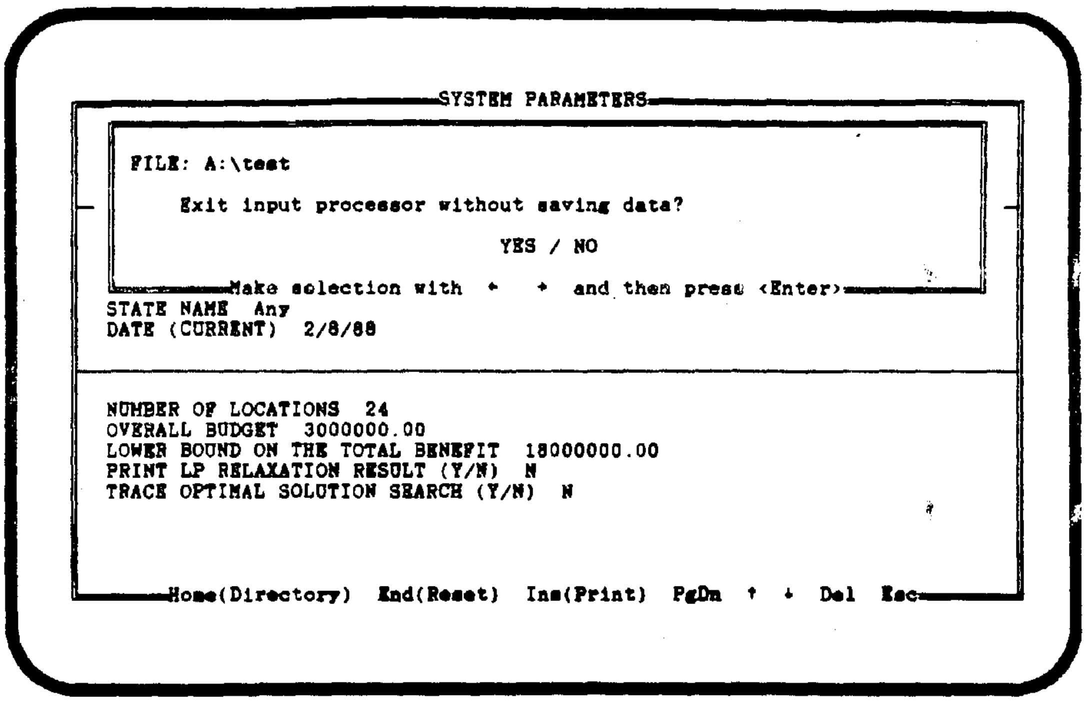


```
                         SYSTEM PARAMETERS
   FILE: A:\test

           Overwrite the above existing disk file?

                              YES / NO

        Make selection with ← → and then press <Enter>
STATE NAME  Any
DATE (CURRENT)  2/8/88

NUMBER OF LOCATIONS  24
OVERALL BUDGET  3000000.00
LOWER BOUND ON THE TOTAL BENEFIT  18000000.00
PRINT LP RELAXATION RESULT (Y/N)  N
TRACE OPTIMAL SOLUTION SEARCH (Y/N)  N

     Home(Directory)  End(Reset)  Ins(Print)  PgDn  ↑  ↓  Del  Esc
```

Figure 11. Terminal screen: Exit check number 1.

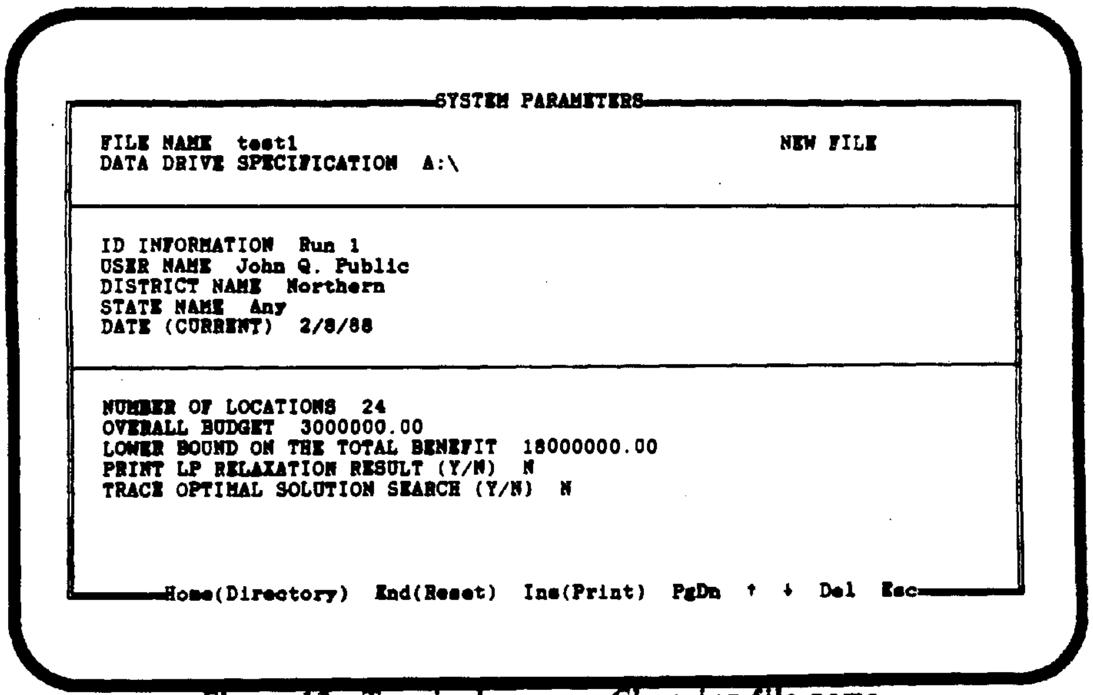
The window contains a question asking the user if the existing file named "TEST" on the default drive could be overwritten. In other words, does the user intend to replace the contents of the old file with the new data currently displayed on screen (and in memory)? In this sample session, the user answers "No" to preserve the contents of the sample data set. The program immediately responds with another question in the window as shown in figure 12. Since the user has not saved the modified data set, he/she answers "No" to the second question also.

The correct way to save the modified data file here is to give the file a distinct new name. To accomplish this, the user moves the highlight bar to the file name field and enters a new name followed by the <Enter> key. The name "test1" will be used in the sample session to designate the modified data set. After entering the "test1" name, the user sees a "System Parameters" screen as shown in figure 13.



```
SYSTEM PARAMETERS

FILE: A:\test

    Exit input processor without saving data?

                                YES / NO

Make selection with ←  → and then press <Enter>
STATE NAME  Any
DATE (CURRENT)  2/8/88

NUMBER OF LOCATIONS  24
OVERALL BUDGET  3000000.00
LOWER BOUND ON THE TOTAL BENEFIT  18000000.00
PRINT LP RELAXATION RESULT (Y/N)  N
TRACE OPTIMAL SOLUTION SEARCH (Y/N)  N

Home(Directory)  End(Reset)  Ins(Print)  PgDn  ↑  ↓  Del  Esc
```

Figure 12. Terminal screen: Exit check number 2.



```
SYSTEM PARAMETERS

FILE NAME  test1                                              NEW FILE
DATA DRIVE SPECIFICATION  A:\

ID INFORMATION  Run 1
USER NAME  John Q. Public
DISTRICT NAME  Northern
STATE NAME  Any
DATE (CURRENT)  2/8/88

NUMBER OF LOCATIONS  24
OVERALL BUDGET  3000000.00
LOWER BOUND ON THE TOTAL BENEFIT  18000000.00
PRINT LP RELAXATION RESULT (Y/N)  N
TRACE OPTIMAL SOLUTION SEARCH (Y/N)  N

Home(Directory)  End(Reset)  Ins(Print)  PgDn  ↑  ↓  Del  Esc
```

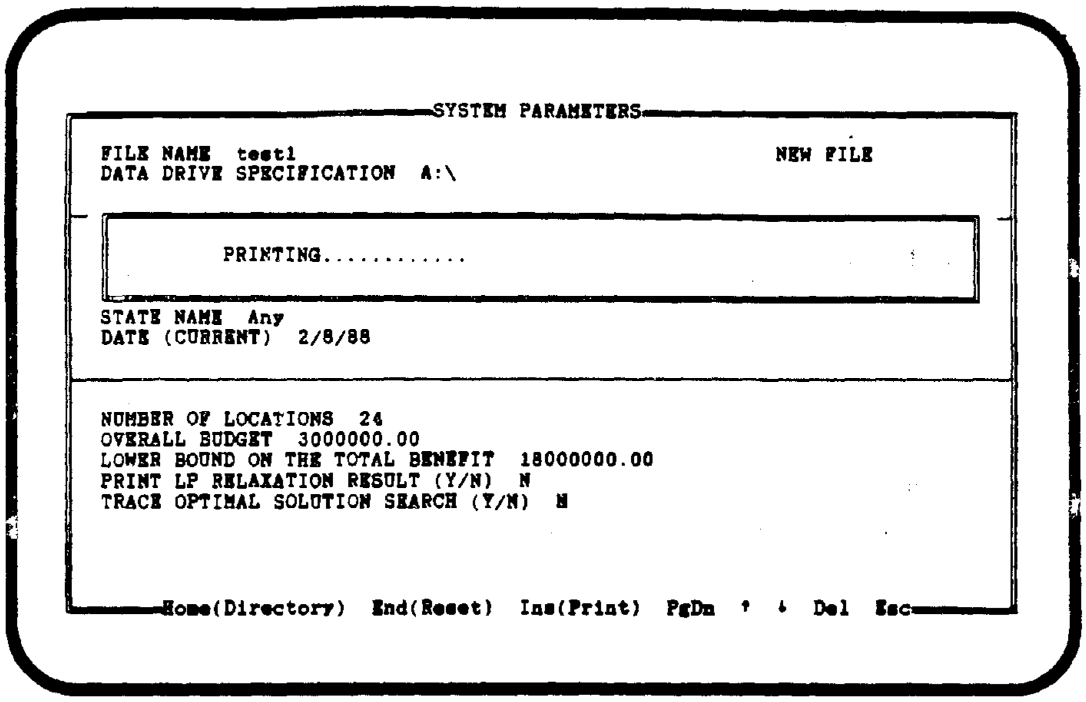
Figure 13. Terminal screen: Changing file name.

At this point, the user should notice that first, a "NEW FILE" tag is attached to the end of the file name field; and secondly, that the name "test1" was accepted by the program without any warning message. Had there been a file named "TEST1.SAF" residing on the default drive, a message such as the one shown in figure 5 would have been given by the program. At this moment, although the file "test1" has not been physically saved on the drive, all necessary checks have been performed and, as soon as the <Esc> key is pressed, the file will be saved automatically. The file name field is, in essence, the key to all file retrieving and saving operations.

The <Esc> key should not be pressed at this time as the sample session continues with the discussion of other command keys in the help line. The <Ins> key is for instructing the program to print a data summary report. The user simply hits the <Ins> key after making certain the printer is turned on. A "printing" message pops up in a window as shown in figure 14. The message disappears as the printing is completed. The data summary report presents all input data including identification information in a clear, concise manner. It serves as a hard-copy summary of the problem being analyzed. The data summary report for this "test1" data file is given in appendix A.

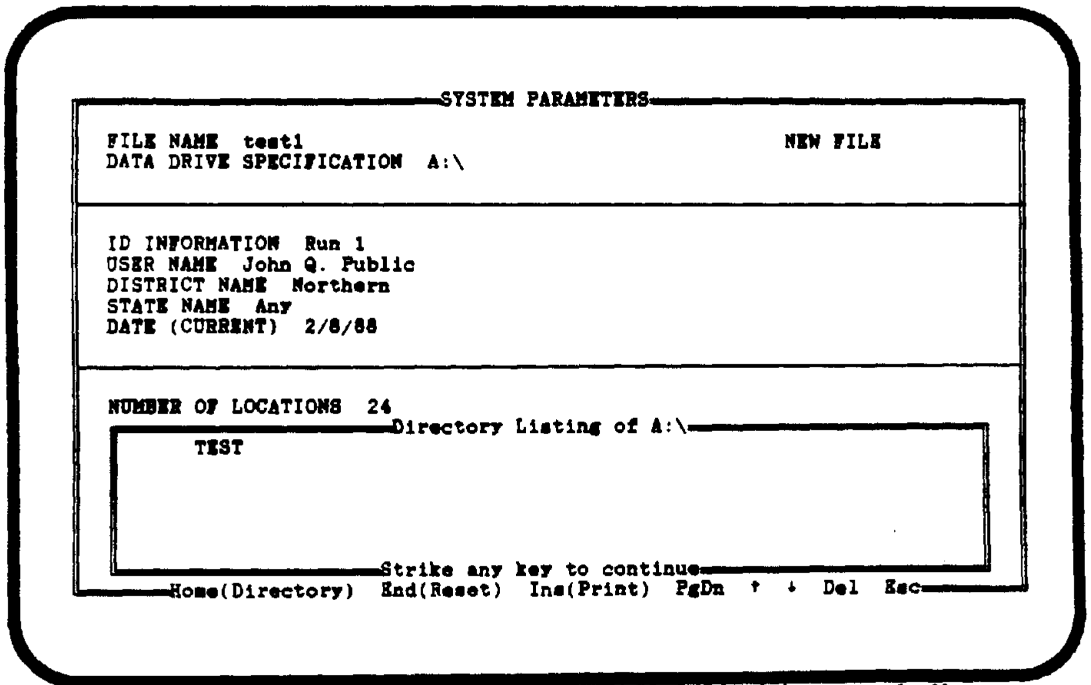
The <Home> key is for the user to catalog the default drive or sub-directory. Pressing this key displays a pop-up window containing a directory of all data files currently residing on the drive. Note that only data files with the ".SAF" extension will be listed in the directory. All other user files not related to the SRAP program will not be included. Figure 15 shows the directory listing activated by the <Home> key. As the "TEST.SAF" file is the only data file currently on the default drive, it is the one that appears in the listing.

The last command key in the help line is the <End> key. This key resets all data fields to their default values (mostly blank fields). It is used when a user wants to discard all current entries in the data fields to start a problem afresh. The user may press the <End> key now and be presented with the pop-up window of figure 16. A "No" answer should be given to continue the sample session. Had the user answered "Yes" to the reset prompt, a blank



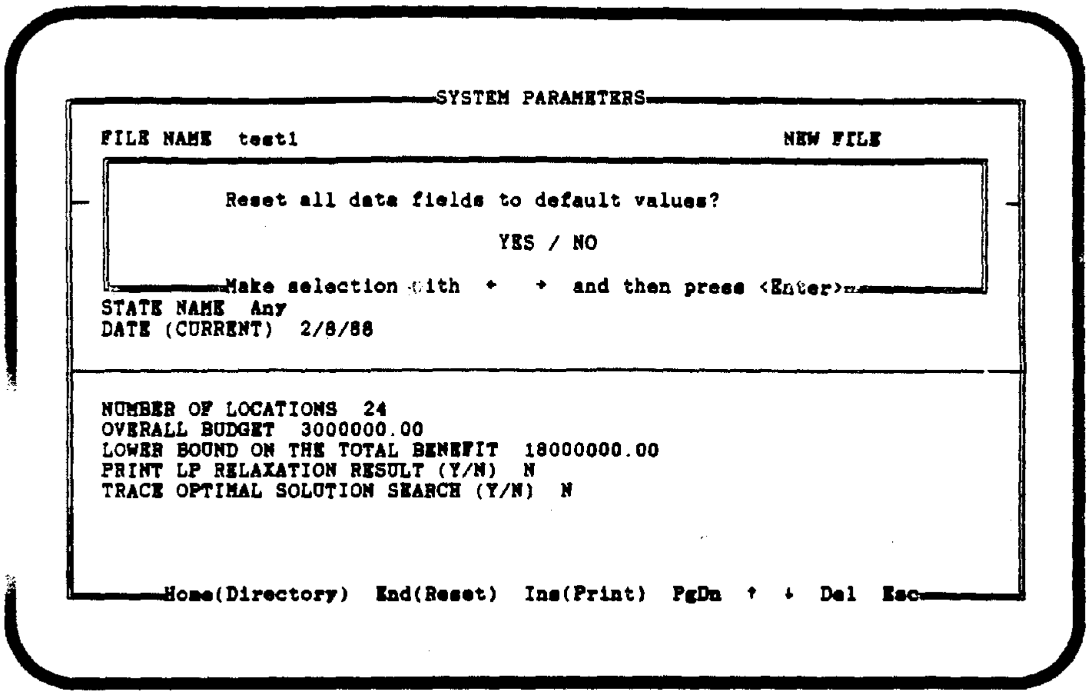
```
─────────────────SYSTEM PARAMETERS────────────────────
 FILE NAME  test1                                    NEW FILE
 DATA DRIVE SPECIFICATION  A:\

     PRINTING............

 STATE NAME  Any
 DATE (CURRENT)  2/8/88

 NUMBER OF LOCATIONS  24
 OVERALL BUDGET  3000000.00
 LOWER BOUND ON THE TOTAL BENEFIT  18000000.00
 PRINT LP RELAXATION RESULT (Y/N)  N
 TRACE OPTIMAL SOLUTION SEARCH (Y/N)  N

────Home(Directory)  End(Reset)  Ins(Print)  PgDn  ↑  ↓  Del  Esc────
```

Figure 14. Terminal screen: Printing data summary report.



```
─────────────────SYSTEM PARAMETERS────────────────────
 FILE NAME  test1                                    NEW FILE
 DATA DRIVE SPECIFICATION  A:\

 ID INFORMATION  Run 1
 USER NAME  John Q. Public
 DISTRICT NAME  Northern
 STATE NAME  Any
 DATE (CURRENT)  2/8/88

 NUMBER OF LOCATIONS  24
 ───────────────Directory Listing of A:\──────────────
       TEST

 ───────────────Strike any key to continue────────────
────Home(Directory)  End(Reset)  Ins(Print)  PgDn  ↑  ↓  Del  Esc────
```

Figure 15. Terminal screen: Cataloging the default drive or sub-directory.



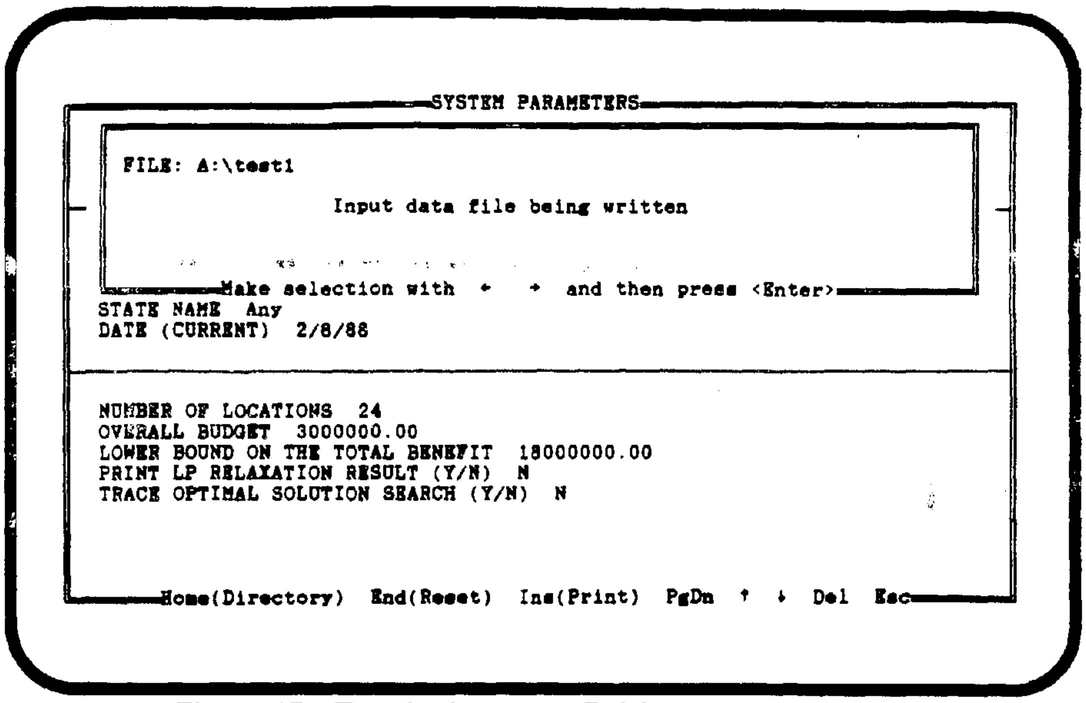
```
                      SYSTEM PARAMETERS
FILE NAME  test1                                        NEW FILE

          Reset all data fields to default values?

                            YES / NO

          Make selection with  ←   →  and then press <Enter>
STATE NAME  Any
DATE (CURRENT)  2/8/88

NUMBER OF LOCATIONS  24
OVERALL BUDGET  3000000.00
LOWER BOUND ON THE TOTAL BENEFIT  18000000.00
PRINT LP RELAXATION RESULT (Y/N)  N
TRACE OPTIMAL SOLUTION SEARCH (Y/N)  N

    Home(Directory)  End(Reset)  Ins(Print)  PgDn  ↑  ↓  Del  Esc
```

Figure 16. Terminal screen: Data fields reset (clearing memory).

"System Parameters" screen like the one in figure 3 would have been displayed and all current data would have been lost.

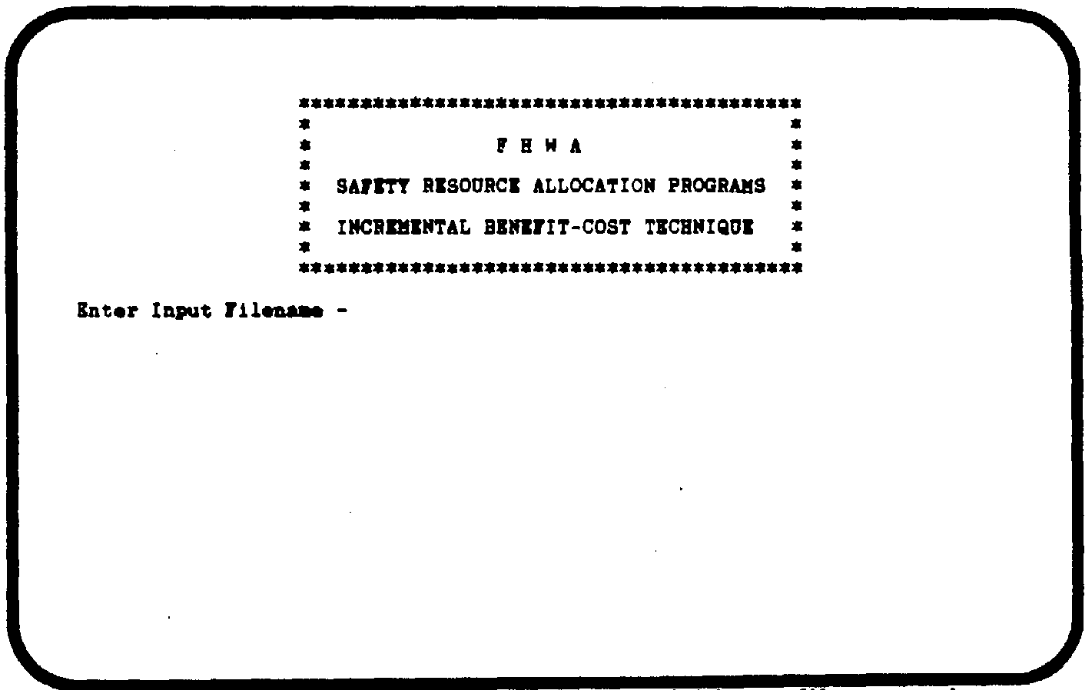
The sample session has now completed all discussions with the input processor. The user presses the <Esc> key to exit the input processor and return to the main menu. During the exiting process, a pop-up window appears informing the user of the saving of the modified data set named "test1." This is illustrated in figure 17.

From the main menu, the next logical step is to select one of the resource allocation models to perform the analysis. In this sample session, the user should choose the incremental benefit-cost analysis model (INCBEN) by highlighting the option followed by the <Enter> key. A screen as shown in figure 18 appears.



```
                         SYSTEM PARAMETERS
  FILE: A:\test1

               Input data file being written

     Make selection with ←  → and then press <Enter>
  STATE NAME  Any
  DATE (CURRENT)  2/8/88

  NUMBER OF LOCATIONS  24
  OVERALL BUDGET  3000000.00
  LOWER BOUND ON THE TOTAL BENEFIT  18000000.00
  PRINT LP RELAXATION RESULT (Y/N)  N
  TRACE OPTIMAL SOLUTION SEARCH (Y/N)  N

  Home(Directory)  End(Reset)  Ins(Print)  PgDn  ↑  ↓  Del  Esc
```

Figure 17. Terminal screen: Exiting input processor.



```
              *******************************************
              *                                         *
              *                F H W A                  *
              *                                         *
              *   SAFETY RESOURCE ALLOCATION PROGRAMS   *
              *                                         *
              *   INCREMENTAL BENEFIT-COST TECHNIQUE    *
              *                                         *
              *******************************************
Enter Input Filename -
```

Figure 18. Terminal screen: Analysis model input file prompting.

The screen contains a program logo for the INCBEN model and an input file name prompt. It is asking the user to enter the name of the data file to be analyzed. The user types in the name "test1" followed by a carriage return. A second prompt appears as shown in figure 19.

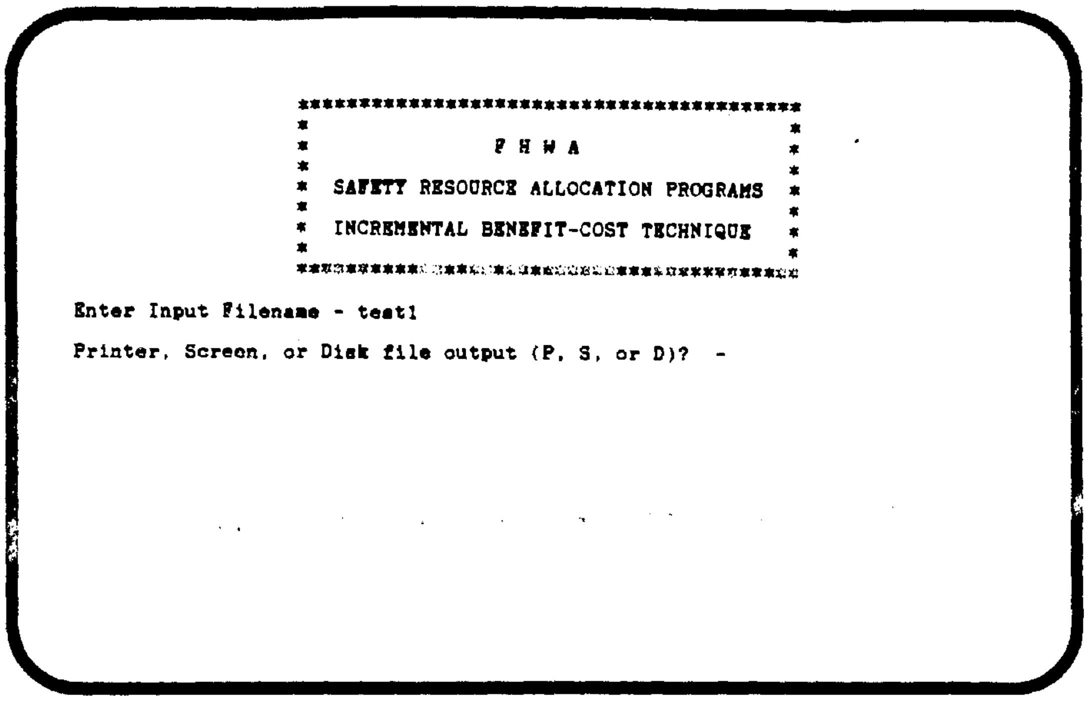
The user may specify either of three different dispositions for the model output. The output can be printed from the printer, displayed on screen, or routed (and saved) to a drive as a disk file. After the user enters the desired output option, the model analysis commences and the "Program is running" message is displayed. Figure 20 shows the screen display when a printer output option is chosen. At the end of model run, a "Stop - Program terminated" message is given and the user is prompted to hit any key to return to the main menu.

The output generated by model INCBEN is 132-column wide. If the user's printer is of the narrow-carriage type, he/she should preset the printer to print in the "compressed" mode. The other two models, INTPROG and DYNPROG, produce outputs that can be accommodated in the regular 80-column print mode.

If the user chooses the disk file output option, he is prompted by the program to enter an output file name. This is illustrated in figure 21. Note that the output file will be saved on the default drive or sub-directory unless the file name contains a different drive specification.

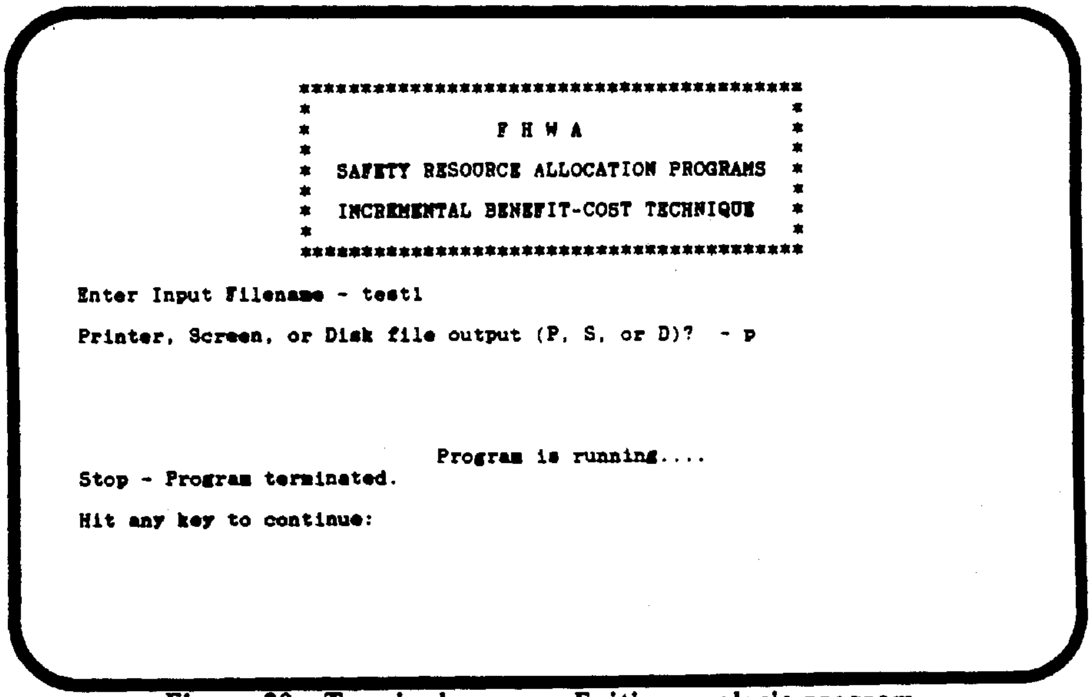
All input and output file names entered by the user are rigorously checked by the program to prevent any error. For example, figure 22 shows a case in which the program warns the user about the error of specifying a nonexistent data file named "test2." It gives the user the option of re-entering a new name or quitting the analysis altogether. Other checks for errors such as duplicate file names also are performed by SRAP.

Before concluding the sample session, a final point should be noted regarding the user's printer configuration with his/her microcomputer system. SRAP is designed to function as if the user's printer is connected to the



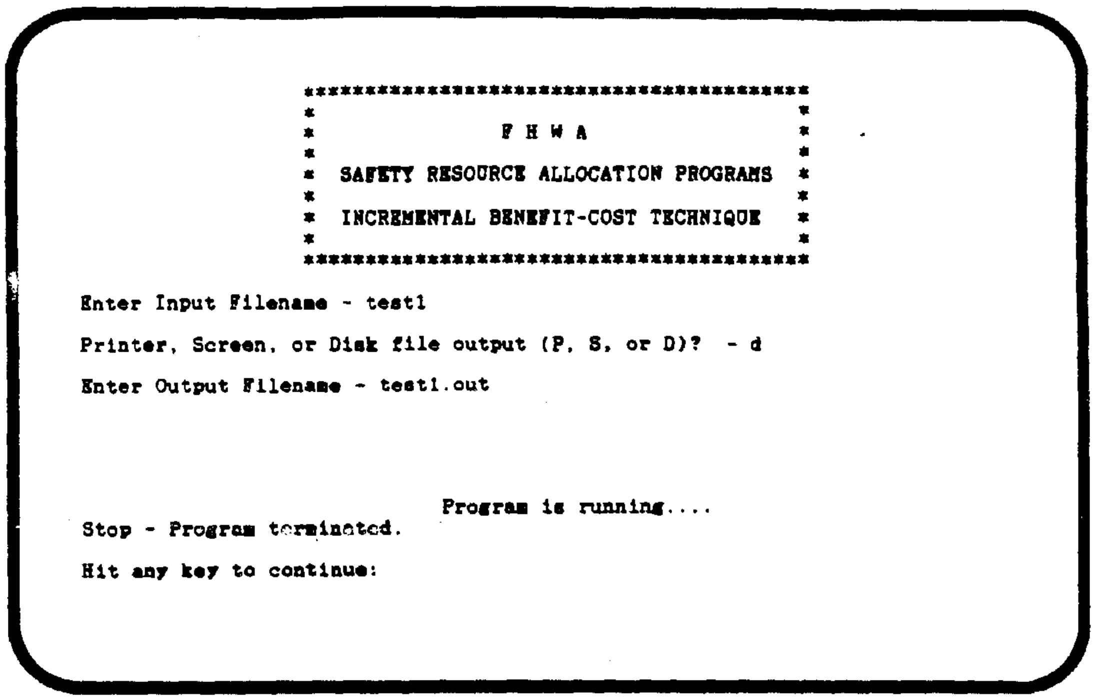
```
                    ******************************************
                    *                                        *
                    *                 F H W A                *
                    *                                        *
                    *  SAFETY RESOURCE ALLOCATION PROGRAMS   *
                    *                                        *
                    *  INCREMENTAL BENEFIT-COST TECHNIQUE    *
                    *                                        *
                    ******************************************

Enter Input Filename - test1

Printer, Screen, or Disk file output (P, S, or D)?  -
```

Figure 19. Terminal screen: Analysis model output file prompting.



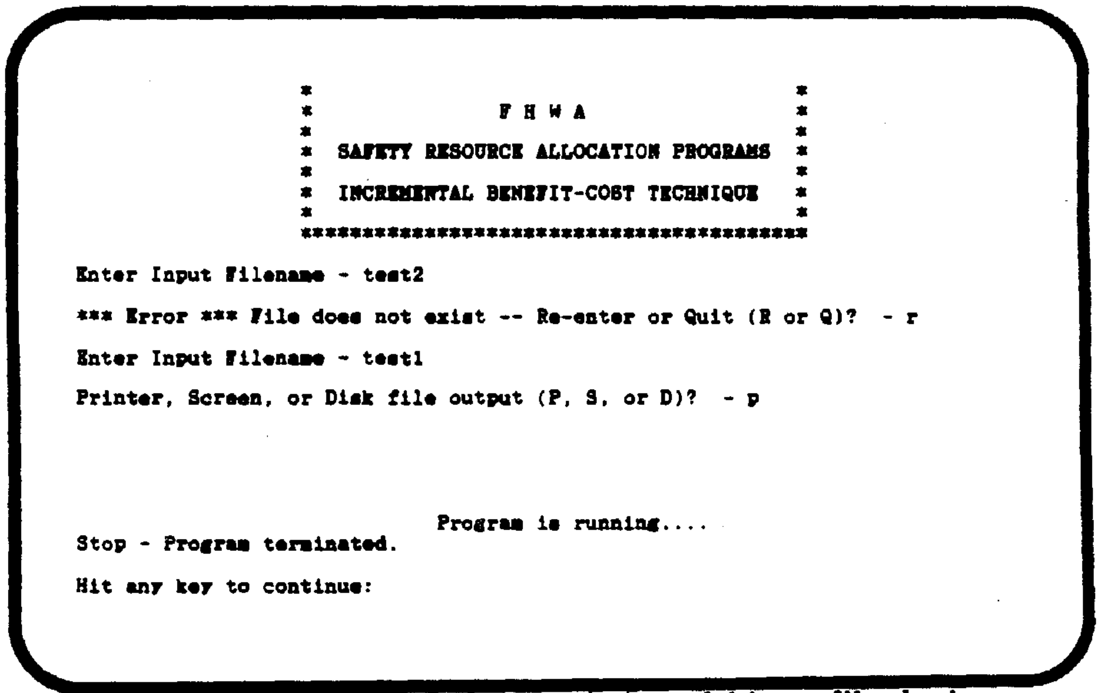
```
                    ******************************************
                    *                                        *
                    *                 F H W A                *
                    *                                        *
                    *  SAFETY RESOURCE ALLOCATION PROGRAMS   *
                    *                                        *
                    *  INCREMENTAL BENEFIT-COST TECHNIQUE    *
                    *                                        *
                    ******************************************

Enter Input Filename - test1

Printer, Screen, or Disk file output (P, S, or D)?  - p



                                  Program is running....
Stop - Program terminated.

Hit any key to continue:
```

Figure 20. Terminal screen: Exiting analysis program.



```
                    *****************************************
                    *                                       *
                    *                F H W A                *
                    *                                       *
                    *  SAFETY RESOURCE ALLOCATION PROGRAMS  *
                    *                                       *
                    *  INCREMENTAL BENEFIT-COST TECHNIQUE   *
                    *                                       *
                    *****************************************
Enter Input Filename - test1
Printer, Screen, or Disk file output (P, S, or D)?  - d
Enter Output Filename - test1.out



                              Program is running....
Stop - Program terminated.
Hit any key to continue:
```

Figure 21. Terminal screen: Disk file output specification.



```
                    *                                       *
                    *                F H W A                *
                    *                                       *
                    *  SAFETY RESOURCE ALLOCATION PROGRAMS  *
                    *                                       *
                    *  INCREMENTAL BENEFIT-COST TECHNIQUE   *
                    *                                       *
                    *****************************************
Enter Input Filename - test2
*** Error *** File does not exist -- Re-enter or Quit (R or Q)?  - r
Enter Input Filename - test1
Printer, Screen, or Disk file output (P, S, or D)?  - p



                              Program is running....
Stop - Program terminated.
Hit any key to continue:
```

Figure 22. Terminal screen: Analysis model input file check.

system through the parallel printer port number 1 or LPT1 (which is the most typical configuration). If, however, the user's system is configured otherwise, he/she will have to reassign the output device or port using the DOS "MODE" command. For example, for a system with a printer connected through the serial communication port number 1 (COM1), the following DOS command will redirect all SRAP printer output.

```
MODE LPT1:=COM1
```

The DOS users manual should be consulted for more information. In any case, if the user has obtained the data summary report or model output during the sample session, no configuration change will be needed. The output from the INCBEN model for the sample test problem is given in appendix B. Runs using the other two analysis models can be done similarly and are not presented in the sample session.

Output Interpretation

Output from each of the three safety resource allocation models is briefly discussed below. The discussion refers specifically to the output generated for the sample test problem (which is on the distribution diskette and has been described in the tutorial). Due to space limitation, only the output from model INCBEN is reproduced in appendix B. The user should execute the other two models (INTPROG and DYNPROG) and obtain their outputs before reading this section. All three outputs discussed are for the safety evaluation problem summarized in appendix A. A total of 24 high-hazard locations were included in the problem.

Incremental Benefit-Cost Analysis (INCBEN)

The output from INCBEN consists of six parts, as shown in appendix B. They are as follows:

1. Listing of input data, showing the total number of locations (24 in this example), the available budget ($3,000,000), and each countermeasure alternative along with its respective initial cost and present value of annual net benefits over its service life ($765,000 and $1,132,200, respectively, for Alternative A at the first location). This portion of the output simply prints out the input data.

2. Listing of projects deleted from the array of alternatives due to equal initial costs, but not greater net benefits when comparing one alternative at a given location with the immediately preceding alternative at that location. In this example, there are no such projects. However, had there been, say, an alternative at Location 1 with a cost of $765,000 (equal to that of Alternative 971A) and benefits of $1,132,200 or less, then that alternative would have been deleted, leaving 971A in the array.

3. Results of the incremental benefit-cost procedure, where project costs, benefits, incremental costs, incremental benefits, and incremental and average benefit-cost ratios are calculated. If no average benefit-cost ratio is calculated for a particular project, then "0" is shown for the average benefit-cost ratio for that project.

4. Listing of projects deleted by the procedure due to incremental benefit-cost ratios not greater than one.

5. Listing of remaining alternatives ranked by incremental benefit-cost ratios, with the cumulative cost of projects. For some alternatives, the symbol "**" appears in the cumulative cost column. This symbol indicates that the incremental cost for such an alternative is not included in cumulative cost because the incremental cost of a higher-cost alternative (with a greater incremental benefit-cost ratio) at the same location has already been included in cumulative cost. This avoids "double counting" of costs.

For example, the highest-ranked alternative at Location 16 is project 777B, which is brought into the array of acceptable projects at a cumulative cost of $2,243,900. Further down the list of projects, project 777A is encountered. Since the incremental cost of a higher-cost and higher-ranked alternative (777B) at the same location has already been included in the cumulative cost, the incremental cost of 777A is not added to cumulative cost. (The incremental benefit-cost calculation procedure has included the incremental cost of the lower-cost and lower-ranked alternative at the same location, 777A, in the incremental cost of 777B.) Therefore, the "**" symbol appears in the cumulative cost column for 777A.

On the other hand, take the example of project 741A, which is the highest-ranked alternative at Location 19 and is brought into the list at a cumulative cost of $1,139,900. Further down the list, project 741B is found. Since Alternative 741A has already been ranked, and 741B is a higher-cost alternative than 741A at the same location, the cumulative cost increases by the incremental cost of 741B, which is the difference between the initial costs of 741B and 741A, or $552,200 - $50,000 = $502,200. In this case, no "**" symbol would apply to project 741B.

6. Listing of optimal solution of selected projects for the available budget, based on the ranking described in part 5. The project selection process is such that, once an alternative at a given location is selected, a lower-cost alternative at that location will not be accepted, although a higher-cost alternative at that location may be accepted in lieu of the previously chosen alternative. For example, 777B is the highest-ranked alternative at Location 16 and, since the budget is sufficient, it is brought into the solution. However, when the lower-cost 777A is encountered further down the list, 777A is passed over since the higher-cost and higher-ranked 777B has already been selected.

For a $3,000,000 budget, projects shown through 991A are selected, with a cumulative cost of $2,878,900. The next project, 971D, has an incremental cost of $154,000, and thus, will not fit within the remaining budget. The next project which will fit within the remaining budget is 772A, with an incremental cost of $65,000, leaving $56,100 left in the budget. Of the projects left in the list, only one, 901B, with an incremental cost of $42,500, can be included at this time. However, since 901B is a higher-cost alternative than 901A that has already been selected at that location, 901A is dropped from the solution in favor of 901B. This leaves an unspent budget of $13,600 and a total cumulative benefit of $28,969,100.

At this point, a switching rule is used to check whether the last project selected can be dropped and one or more projects substituted, within the remaining budget, thereby producing greater total benefits. In this example, dropping the last selected project (901B) will leave a remaining budget of $88,000, allowing 772B (with an incremental cost of $50,000) to be brought into the solution. Project 772B is, in fact, the highest-ranked alternative among the ones remaining in the list which can fit within $88,000. The inclusion of 772B in the solution also forces out the previously selected Alternative 772A at the same location. Since dropping 901B and adding 772B decreases total net benefits by $2,112,000 (equal to 901B's benefit of $2,461,000 less 772B's benefit of $349,000, as seen in the listing of input data), the original solution (containing 901B) is the optimal solution.

It should be noted that, had there been one or more additional projects costing no more than the remaining budget of $38,000 (after dropping 901B and adding 772B), then these projects would have been added to the solution. Then the total net benefits of the original set of projects including 901B would have been compared with the set including 772B and the other accepted projects (but not 901B), and the set yielding more benefits would have been selected.

Integer Programming (INTPROG)

The output from INTPROG consists of three parts. They are as follows:

1. Listing of input data, as explained under model INCBEN. In addition, a lower bound on total net benefits from project selection is set at $18,000,000.

2. Intermediate calculations, depending on the values of the two print options. In this example, neither option is activated so that no intermediate calculations prior to the first selection iteration are printed. The listing of the unconverged solutions from intermediate iterations, before reaching the optimal solution (i.e., benefit-maximizing set of projects), is given by the program. Iteration 1 in this example results in a set of projects costing $2,309,900 and yielding total net benefits of $25,396,000. After another iteration, however, more projects are selected, giving benefits of $28,621,400 for a cost of $2,878,900. The third iteration continues to add projects to the solution, spending more of the budget and adding more to cumulative benefits. The higher that benefit lower bound is set, the fewer will be the number of iterations.

3. Optimal solution, or that set of projects which yields maximum total annual net benefits for the available budget. In this example, benefits of $28,970,400 are obtained by spending $2,993,900 of the available budget of $3,000,000. If the solution for the last iteration indicates that the total cost is 0.0, then this indicates that the lower bound was set too high. In other words, the set bound exceeds the total benefits that can be produced for the available budget. In this event, the bound should be set at a lower value and the program then rerun.

Dynamic Programming (DYNPROG)

The output from model DYNPROG consists of two parts. They are as follows:

1. Listing of input data, as explained under model INCBEN.

2. Selection of projects, showing the optimal policy or set of projects which maximizes total annual net benefits for the available budget. In this example, benefits of $28,969,100 are obtained for an expenditure of $2,986,400.

APPENDIX A
TEST DATA SUMMARY REPORT

```
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$                                        $
$       FEDERAL HIGHWAY ADMINISTRATION   $
$     SAFETY RESOURCE ALLOCATION PROGRAMS $
$             INPUT PROCESSOR            $
$                                        $
$              Version  1.00             $
$                                        $
$     DEVELOPED BY SRA TECHNOLOGIES, INC. $
$                                        $
$               OCTOBER 1987             $
$                                        $
$           DATA SUMMARY REPORT          $
$                                        $
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$


---------------------
SYSTEM-WIDE PARAMETERS
---------------------

     ID Information: Run 1
     User Name      : John Q. Public
     District Name  : Northern
     State Name     : Any
     Date           : 2/8/88

     Number of Locations                   : 24
     Overall Budget ($)                    : 3000000.00
     Lower Bound on the Total Benefit ($): 18000000.00
     Print LP Relaxation Result            : No
     Trace Optimal Solution Search         : No


----------------
BENEFIT/COST DATA
----------------

Location   Alt.   Project    Initial Cost ($)     Total Net Benefit ($)
========================================================================
    1        1      971A         765000.00              1132200.00
             2      971B         815000.00              1132900.00
             3      971C        1625000.00              3607500.00
             4      971D         154000.00               750000.00
------------------------------------------------------------------------
    2        1      171A         221000.00               908300.00
             2      171B         259000.00               909100.00
------------------------------------------------------------------------
    3        1      781A         365000.00               927100.00
             2      781B         440000.00              1020800.00
========================================================================
```

Location	Alt.	Project	Initial Cost ($)	Total Net Benefit ($)
4	1	241A	864500.00	2039000.00
	2	241B	1203300.00	2334400.00
5	1	821A	748000.00	9140600.00
	2	821B	1653200.00	3637000.00
6	1	231A	778000.00	5780500.00
	2	231B	781000.00	2975600.00
7	1	841A	1725000.00	2777300.00
	2	841B	2641700.00	3064400.00
8	1	771A	556000.00	795100.00
	2	771B	572000.00	1538700.00
9	1	651A	66000.00	440200.00
	2	651B	169100.00	270600.00
10	1	851A	3100000.00	4867000.00
	2	851B	3350000.00	4857500.00
	3	851C	4150000.00	4855500.00
11	1	772A	65000.00	279200.00
	2	772B	115000.00	349000.00
12	1	773A	1461200.00	4407300.00
	2	773B	2750000.00	8827500.00
13	1	774A	200000.00	604000.00
	2	774B	290000.00	597400.00
14	1	775A	260000.00	3211000.00
	2	775B	310000.00	4808100.00
15	1	776A	2650000.00	7950000.00
16	1	777A	160000.00	1062400.00
	2	777B	326000.00	2314600.00
17	1	991A	439000.00	2458400.00
	2	991B	660000.00	2277000.00
18	1	901A	31900.00	2392500.00
	2	901B	74400.00	2461000.00
19	1	741A	50000.00	519500.00
	2	741B	552200.00	2275100.00

Location	Alt.	Project	Initial Cost ($)	Total Net Benefit ($)
20	1	581A	1055000.00	91300.00
21	1	582A	84000.00	22200.00
	2	582B	171000.00	64400.00
22	1	131A	130000.00	767000.00
23	1	141A	2094300.00	2408500.00
	2	141B	3600000.00	2772000.00
24	1	142A	249800.00	699400.00

APPENDIX B
TEST DATA OUTPUT FROM PROGRAM INCBEN

```
*****************************************
*                                       *
*               F H W A                 *
*                                       *
*  SAFETY RESOURCE ALLOCATION PROGRAMS  *
*                                       *
*  INCREMENTAL BENEFIT-COST TECHNIQUE   *
*                                       *
*****************************************


                INPUT DATA
                **********


THE NUMBER OF LOCATIONS =    24
THE BUDGET LEVEL        =   3000000.00


    LOC   PROJ NO           COST          BENEFIT


     1      971A         765000.00      1132200.00

     1      971B         815000.00      1132900.00

     1      971C        1625000.00      3607500.00

     1      971D         154000.00       750000.00


     2      171A         221000.00       908300.00

     2      171B         259000.00       909100.00


     3      781A         365000.00       927100.00

     3      781B         440000.00      1020800.00


     4      241A         864500.00      2039000.00

     4      241B        1203300.00      2334400.00


     5      821A         748000.00      9140600.00

     5      821B        1653200.00      3637000.00


     6      231A         778000.00      5780500.00

     6      231B         781000.00      2975600.00
```

7	841A	1725000.00	2777300.00
7	841B	2641700.00	3064400.00
8	771A	556000.00	795100.00
8	771B	572000.00	1538700.00
9	651A	66000.00	440200.00
9	651B	169100.00	270600.00
10	851A	3100000.00	4867000.00
10	851B	3350000.00	4857500.00
10	851C	4150000.00	4855500.00
11	772A	65000.00	279200.00
11	772B	115000.00	349000.00
12	773A	1461200.00	4407300.00
12	773B	2750000.00	8827500.00
13	774A	200000.00	604000.00
13	774B	290000.00	597400.00
14	775A	260000.00	3211000.00
14	775B	310000.00	4808100.00
15	776A	2650000.00	7950000.00
16	777A	160000.00	1062400.00
16	777B	326000.00	2314600.00
17	991A	439000.00	2458400.00
17	991B	660000.00	2277000.00

18	901A	31900.00	2392500.00
18	901B	74400.00	2461000.00
19	741A	50000.00	519500.00
19	741B	552200.00	2275100.00
20	581A	1055000.00	91300.00
21	582A	84000.00	22200.00
21	582B	171000.00	64400.00
22	131A	130000.00	767000.00
23	141A	2094300.00	2408500.00
23	141B	3600000.00	2772000.00
24	142A	249800.00	699400.00

```
PROJECTS OF SAME COST BUT LESS BENEFIT DELETED
**********************************************


REF   PROJ NO              COST             BENEFIT

              NO PROJECT IS DELETED
```

AN INCREMENTAL BENEFIT-COST ANALYSIS

LOC	PROJ NO	COST	BENEFIT	INC COST	INC BENEFIT	INC BC-RATIO	AVG BC-RATIO
1	971D	154000.00	750000.00	154000.00	750000.00	4.87	.00
1	971C	1625000.00	3607500.00	1471000.00	2857500.00	1.94	.00
2	171A	221000.00	908300.00	221000.00	908300.00	4.11	.00
3	781A	365000.00	927100.00	365000.00	927100.00	2.54	.00
3	781B	440000.00	1020800.00	75000.00	93700.00	1.25	.00
4	241A	864500.00	2039000.00	864500.00	2039000.00	2.36	.00
5	821A	748000.00	9140600.00	748000.00	9140600.00	12.22	.00
6	231A	778000.00	5780500.00	778000.00	5780500.00	7.43	.00
7	841A	1725000.00	2777300.00	1725000.00	2777300.00	1.61	.00
8	771A	556000.00	795100.00	556000.00	795100.00	1.43	.00
8	771B	572000.00	1538700.00	16000.00	743600.00	46.47	2.69
9	651A	66000.00	440200.00	66000.00	440200.00	6.67	.00
10	851A	3100000.00	4867000.00	3100000.00	4867000.00	1.57	.00
11	772A	65000.00	279200.00	65000.00	279200.00	4.30	.00
11	772B	115000.00	349000.00	50000.00	69800.00	1.40	.00
12	773A	1461200.00	4407300.00	1461200.00	4407300.00	3.02	.00
12	773B	2750000.00	8827500.00	1288800.00	4420200.00	3.43	3.21

13	774A	200000.00	604000.00	200000.00	604000.00	3.02	.00
14	775A	260000.00	3211000.00	260000.00	3211000.00	12.35	.00
14	775B	310000.00	4808100.00	50000.00	1597100.00	31.94	15.51
15	776A	2650000.00	7950000.00	2650000.00	7950000.00	3.00	.00
16	777A	160000.00	1062400.00	160000.00	1062400.00	6.64	.00
16	777B	326000.00	2314600.00	166000.00	1252200.00	7.54	7.10
17	991A	439000.00	2458400.00	439000.00	2458400.00	5.60	.00
18	901A	31900.00	2392500.00	31900.00	2392500.00	75.00	.00
18	901B	74400.00	2461000.00	42500.00	68500.00	1.61	.00
19	741A	50000.00	519500.00	50000.00	519500.00	10.39	.00
19	741B	552200.00	2275100.00	502200.00	1755600.00	3.50	.00
22	131A	130000.00	767000.00	130000.00	767000.00	5.90	.00
23	141A	2094300.00	2408500.00	2094300.00	2408500.00	1.15	.00
24	142A	249800.00	699400.00	249800.00	699400.00	2.80	.00

PROJECTS DELETED
****

LOC	PROJ NO	COST	BENEFIT	INC COST	INC BENEFIT	INC BC-RATIO
1	971A	765000.00	1132200.00	611000.00	382200.00	.63
1	971B	815000.00	1132900.00	661000.00	382900.00	.58
2	171B	259000.00	909100.00	38000.00	800.00	.02
4	241B	1203300.00	2334400.00	338800.00	295400.00	.87
5	821B	1653200.00	3637000.00	905200.00	-5503600.00	-6.08
6	231B	781000.00	2975600.00	3000.00	-2804900.00	-934.97
7	841B	2641700.00	3064400.00	916700.00	287100.00	.31
9	651B	169100.00	270600.00	103100.00	-169600.00	-1.65
10	851B	3350000.00	4857500.00	250000.00	-9500.00	-.04
10	851C	4150000.00	4855500.00	1050000.00	-11500.00	-.01
13	774B	290000.00	597400.00	90000.00	-6600.00	-.07
17	991B	660000.00	2277000.00	221000.00	-181400.00	-.82
20	581A	1055000.00	91300.00	1055000.00	91300.00	.09
21	582A	84000.00	22200.00	84000.00	22200.00	.26
21	582B	171000.00	64400.00	171000.00	64400.00	.38
23	141B	3600000.00	2772000.00	1505700.00	363500.00	.24

SELECTION OF PROJECTS

PROJ NO	COST	BENEFIT	INC COST	BC-RATIO	CUM COST
901A	31900.00	2392500.00	31900.00	75.00	31900.00
775B	310000.00	4808100.00	310000.00	15.51	341900.00
775A	260000.00	3211000.00	260000.00	12.35	**
821A	748000.00	9140600.00	748000.00	12.22	1089900.00
741A	50000.00	519500.00	50000.00	10.39	1139900.00
231A	778000.00	5780500.00	778000.00	7.43	1917900.00
777B	326000.00	2314600.00	326000.00	7.10	2243900.00
651A	66000.00	440200.00	66000.00	6.67	2309900.00
777A	160000.00	1062400.00	160000.00	6.64	**
131A	130000.00	767000.00	130000.00	5.90	2439900.00
991A	439000.00	2458400.00	439000.00	5.60	2878900.00
971D	154000.00	750000.00	154000.00	4.87	3032900.00
772A	65000.00	279200.00	65000.00	4.30	3097900.00
171A	221000.00	908300.00	221000.00	4.11	3318900.00
741B	552200.00	2275100.00	502200.00	3.50	3821100.00
773B	2750000.00	8827500.00	2750000.00	3.21	6571100.00
774A	200000.00	604000.00	200000.00	3.02	6771100.00
773A	1461200.00	4407300.00	1461200.00	3.02	**
776A	2650000.00	7950000.00	2650000.00	3.00	9421100.00
142A	249800.00	699400.00	249800.00	2.80	9670900.00
771B	572000.00	1538700.00	572000.00	2.69	10242900.00
781A	365000.00	927100.00	365000.00	2.54	10607900.00
241A	864500.00	2039000.00	864500.00	2.36	11472400.00
971C	1625000.00	3607500.00	1471000.00	1.94	12943400.00
901B	74400.00	2461000.00	42500.00	1.61	12985900.00
841A	1725000.00	2777300.00	1725000.00	1.61	14710900.00
851A	3100000.00	4867000.00	3100000.00	1.57	17810900.00
771A	556000.00	795100.00	556000.00	1.43	**
772B	115000.00	349000.00	50000.00	1.40	17860900.00

```
  781B        440000.00      1020800.00        75000.00          1.25    17935900.00
  141A       2094300.00      2408500.00      2094300.00          1.15    20030200.00
THE PREFERRED SOLUTION OF PROJECTS FOR A FIXED BUDGET OF       3000000.00   IS :


PROJ NO          COST          BENEFIT

  821A        748000.00      9140600.00
  231A        778000.00      5780500.00
  651A         66000.00       440200.00
  772A         65000.00       279200.00
  775B        310000.00      4808100.00
  777B        326000.00      2314600.00
  991A        439000.00      2458400.00
  901B         74400.00      2461000.00
  741A         50000.00       519500.00
  131A        130000.00       767000.00

THE TOTAL COST IS            2986400.00
THE TOTAL BENEFIT IS        28969100.00
THE EXCESS BUDGET IS           13600.00
```

REFERENCES

1. McFarland, W.F. and Rollins, J.B., Cost-Effectiveness Techniques for Highway Safety: Resource Allocation--Final Report, Report No. FHWA/RD-84/011, Federal Highway Administration, Washington, DC, June 1985.

2. McFarland, W.F., et al, Documentation for Incremental Benefit-Cost Technique--Program INCBEN, Texas Transportation Institute, Texas A&M University System, College Station, Texas, December 1983.

3. McFarland, W.F., et al, Documentation for Integer Programming Technique--Program INTPROG, Texas Transportation Institute, Texas A&M University System, College Station, Texas, December 1983.

4. Subramanian, B., et al, Documentation for Dynamic Programming Technique--Program DYNPROG, Texas Transportation Institute, Texas A&M University System, College Station, Texas, December 1983.

5. Liu, C.C., Safety Resource Allocation Programs--Implementation Technique, Report No. FHWA-TS-88-019, Federal Highway Administration, Washington, D.C., April 1988.

B173
11