5/9/96

SANDIA REPORT
SAND96–0842 • UC–505
Unlimited Release
Printed April 1996

A User's Guide to SABLE 2.0: The Sandia Automated Boolean Logic Evaluation Software

K. M. Hays, G. D. Wyss, S. L. Daniel

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185 and Livermore, California 94550
for the United States Department of Energy
under Contract DE-AC04-94AL85000

SF2900Q(8-81)

SAND96-0842
Unlimited Release
Printed April, 1996

Distribution
Category UC-505

A User's Guide to SABLE 2.0: The Sandia Automated Boolean Logic Evaluation Software

K.M. Hays, G.D. Wyss, and S.L. Daniel
Risk Assessment and Systems Modeling Department
Sandia National Laboratories
Albuquerque, NM 87185

Abstract

This document is a reference guide for the Sandia Automated Boolean Logic Evaluation software (SABLE) version 2.0 developed at Sandia National Laboratories. SABLE 2.0 is designed to solve and quantify fault trees on IBM - compatible personal computers using the Microsoft Windows operating environment. SABLE 2.0 consists of a Windows user interface combined with a fault tree solution engine that is derived from the well - known SETS fault tree analysis code. This manual explains the fundamentals of solving fault trees and shows how to use the Windows SABLE 2.0 interface to specify a problem, solve the problem, and view the output.

Acknowledgments

The authors wish to recognize and thank Mike Baird (formerly of Sandia National Laboratories) for his work to develop the SABLE Windows user interface and for his efforts to document, modernize and otherwise upgrade the original SETS software into the more useable SABLE FORTRAN analysis engine. We would also like to thank Desmond Stack of Los Alamos National Laboratories, as well as Marty Fuentes and Todd Jones of Sandia for their technical contributions in testing and providing suggestions for improvements to the software. We would also like to thank Dave Carlson and Kevin Maloney of Sandia's Assessment Technologies Department, and Allen Camp of Sandia's Risk Assessment and Systems Modeling Department for their technical and managerial support in this project.

Much of this document is derived from material that was developed for a two-day training course on the SEATREE and SABLE software. We appreciate the many comments and suggestions from course participants that helped improve both the course and this manual. Finally, we would like to thank Donnie Whitehead and Kevin Maloney for their review of this manuscript in preparation for publication.

Acronyms

DNF	Disjunctive Normal Form
ISTS	Independent SubTreeS
LANL	Los Alamos National Laboratory
LHS	Latin Hypercube Sampling
MDI	Multiple Document Interface
SABLE	Sandia Automated Boolean Logic Evaluation Software
SIPRA	Sandia Integrated Probabilistic Risk Assessment
SNL	Sandia National Laboratories
TEMAC3	Top Event Matrix Analysis Code
UNM	University of New Mexico

Contents

A USER'S GUIDE TO SABLE 2.0: The Sandia Automated Boolean Logic Evaluation Software

1 Installation Instructions

1.1 Machine Requirements

To install SABLE 2.0, verify that your computer meets the following minimum hardware and software requirements:

- A 386 IBM - compatible PC or higher.
- 387 or higher math co-processor (the WEITEK math co-processor *is not* an acceptable substitute).
- 8 MB of RAM, preferably 12 MB.
- Approximately 1 MB of free hard drive space.
- DOS Version 5.0 or later.
- Microsoft Windows 3.1 or higher.

The installation program will ensure that your computer meets the above requirements before installing SABLE. *NOTE: Although only 1 MB of hard drive space is required for the SABLE software, remember that SABLE generates large output files. To run SABLE you will need more than 1 MB of free hard drive space.*

1.2 Installing SABLE 2.0

To install SABLE:

- Run Windows.
- Place the SABLE 2.0 Installation Disk #1 in Drive A or Drive B.
- Choose the "Run" command from the Program Manager file menu. Type

  ```
  a:\setup
  ```
  or
  ```
  b:\setup
  ```

  in the Command line edit box and choose "OK". This starts the SABLE 2.0 setup program.

- You will be asked where you want to install SABLE. By default, all programs in Sandia's risk assessment code suite will be installed in "C:\SIPRA". You may change this directory; however, *you must install each of Sandia's risk assessment codes in the same directory*.

- After installation, you must modify your AUTOEXEC.BAT file to include the following line:

```
SET SIPRA_PGM=C:\SIPRA
```

  where "C:\SIPRA" is the directory in which SABLE was installed.

- Reboot your computer.

NOTE: Do not redistribute the SABLE Installation software without written permission from Department 6412; these secondary users would not be registered users and could not be contacted for upgrades.

1.3 Trouble Shooting

In case of problems, please contact:

Risk Assessment and Systems Modeling Department
Organization 6412, MS 0747
PO Box 5800
Sandia National Laboratories
Albuquerque, NM 87185

Phone: (505) 844 - 0056
Fax: (505) 844 - 3321

Appendix B contains a "SABLE Problem Reporting Form" that can be filled out and sent via mail or fax to the Risk Assessment and Systems Modeling Department.

2 Fault Tree Analysis Overview

The SABLE 2.0 Fault Tree Analysis package is but one part of a larger suite of codes developed by Sandia National Laboratories in conjunction with Los Alamos National Laboratory and the University of New Mexico for use by risk analysts. The process of performing a fault tree analysis consists of several steps, and this brief tutorial explains how the various Sandia software products work together to form a complete fault tree risk assessment package. Figure 2-1 shows how the various codes in the risk assessment package work together.

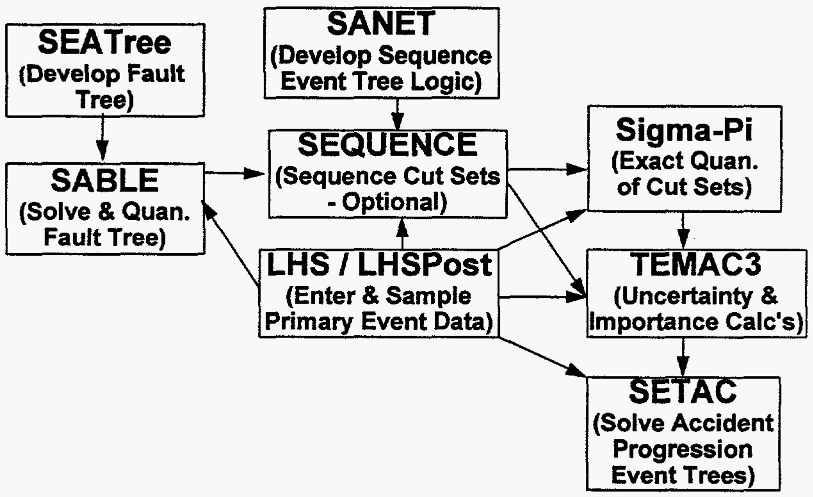

SEATree (Develop Fault Tree)
SANET (Develop Sequence Event Tree Logic)
SABLE (Solve & Quan. Fault Tree)
SEQUENCE (Sequence Cut Sets - Optional)
Sigma-Pi (Exact Quan. of Cut Sets)
LHS / LHSPost (Enter & Sample Primary Event Data)
TEMAC3 (Uncertainty & Importance Calc's)
SETAC (Solve Accident Progression Event Trees)

Figure 2-1: Sandia's Risk Analysis Code Suite

2.1 Developing a Fault Tree

The fault tree analysis begins with the construction of one or more fault trees to represent the system(s) being modeled. Sandia's SEATREE software provides a friendly Windows-based facility for building and graphing fault trees. Fault tree construction occurs in a collapsible list-based view (not unlike the familiar collapsible directory listing found in the Windows "File Manager" application). This view allows a user to enter new events and subtrees, cut and paste subtrees and tree structures, and import fault tree modules from libraries of common components. Users can then switch to an on-screen galley page layout system to develop publication-ready fault tree graphics. SEATREE exports fault tree logic in ASCII-formatted files that are fully compatible with the SABLE fault tree analysis engine.

2.2 Developing Quantification Information

If the fault tree analysis is to be quantitative, probability or frequency data must be entered for each primary event in the fault tree. Sandia's software code suite currently supports three different methods for entering this information.

SEATREE. The SEATREE package (described above for fault tree development) has a data storage mode which allows a user to enter quantification information in a spreadsheet-like format. SEATREE features warn the user of missing values. SEATREE exports the quantification information in ASCII-formatted files that are fully compatible with the SABLE fault tree analysis engine.

LHS. If an integrated fault tree uncertainty analysis is to be performed, it is important that any point estimate quantification information used by SABLE is consistent with the sampled uncertainty data generated by LHS for use in the TEMAC3 cut set importance analysis code. SABLE is able to read the point estimate block in the LHS output file and convert it to a traditional SABLE value block internally.

Text Editor. The format of the quantification data required by SABLE is very simple. Thus, a user could use any ASCII text editor to enter the quantification for a small problem, or to make minor modifications to specific values in the file exported by SEATREE for a larger problem.

2.3 Solving a Fault Tree

The SABLE software described in this manual is used to solve and quantify a fault tree. SABLE is capable of solving multiple fault trees simultaneously to generate minimal cut sets and global independent subtrees. SABLE can perform point estimate cut set quantification using the rare event approximation, and can truncate cut sets that fall below a user-defined probabilistic cut-off level. The cut sets generated by SABLE can be imported directly into the cut set importance analysis code TEMAC3.

2.4 Fault Tree Importance Analysis

Once a fault tree has been solved and minimal cut sets have been obtained, it is important to extract as much useful information as possible from these results. Customers typically want answers to questions such as:

- Which are the most likely combinations of events that can lead to system failure?
- To which primary events is the top event frequency most sensitive?
- Are there some components that do not make a significant contribution to system reliability?
- Which components should I concentrate on improving to achieve the greatest reliability improvement?
- Which events are associated with the greatest fraction of the total failure frequency?
- How sure are we that our results are "close" to the true answer (or, what is the uncertainty in our answer)?

These questions are answered by a cut set uncertainty and sensitivity analysis. These functions are performed by Sandia's LHS code (to sample statistical distributions) and TEMAC3 (developed by Los Alamos, Sandia, and the University of New Mexico to calculate the cut set and event sensitivity and uncertainty results). TEMAC3 can read the cut sets directly from SABLE as well as the same LHS output file that may have been input to SABLE. TEMAC3 can generate output that is suitable for direct inclusion in event tree analyses as performed by Sandia's SETAC package.

2.5 Quantification of Cut Sets

Each cut set generated by a fault tree analysis consists of a group of primary events that is joined by the logical AND operator. If these primary events are independent of one another, the probability of the cut set can be computed as the product of the individual event probabilities. There is no approximation involved in this calculation aside from any assumptions that were used in generating the individual event probabilities. However, special note must be taken in the development of the fault tree to ensure that the assumption of independence is actually true for all primary events.

A cut set expression is a group of cut sets that is joined together by the logical OR operator. To compute the exact probability that results from an OR operation, one must apply the following formula:

$$P(A\ \underline{\textit{or}}\ B) = P(A) + P(B) - P(A\ \underline{\textit{and}}\ B).$$

If A and B are both "rare events" (i.e., they have small probabilities), then the final cross term P(A *and* B) becomes very small. If we neglect this term, we obtain the "rare event approximation," which can be stated as:

$$P(A\ \underline{\textit{or}}\ B) \cong P(A) + P(B).$$

A related formula known as the "minimum cut set upper bound," which neglects similar terms, can be stated as:

$$P(A \ \underline{or}\ B) \cong 1 - \{ [1 - P(A)] * [1 - P(B)] \}.$$

This formula is essentially the rare event approximation computed in complement space. It is esoterically more pleasing than the rare event approximation because it can not produce a result that is greater than unity, but it is not really more accurate. Both approximation techniques break down when there are probabilities greater than approximately 0.1, especially if the fault tree results contain a significant number of cut sets with only one or two events each. Most fault tree analysis software, including SABLE and TEMAC3, implement one of these approximations due to the large amount of computational effort and computer memory required to identify and properly account for the cross product terms in the exact solution. Note that each of these approximations will *overpredict* the actual probability of the cut set expression. Thus, while they yield results that are *conservative*, one has no idea of exactly *how* conservative unless an *exact* quantification of the cut set expression can be achieved.

In certain classes of problems such as lethality studies and success state studies (where it is expected that the probabilities in the cut sets are large), these approximations can break down and yield highly erroneous results. The Sigma Pi code has been incorporated into the Sandia code suite to allow an analyst to compute the *exact* probability of a top event from a cut set expression, should that computation be required for a particular analysis. It is also instructive to run Sigma Pi on routine cut set expressions in order to determine exactly how conservative the results obtained from other codes may be.

2.6 Advanced Fault Tree Analysis Topics

Fault tree analysis is a powerful tool by itself, but it becomes even more powerful when it is used in conjunction with event tree analysis techniques. This is typically done in two ways: through the use of sequence event trees, and by feeding quantitative fault tree analysis results into a follow-on event tree analysis.

Developing Sequence Event Trees. In large fault tree analyses it is often difficult to build a single fault tree for the entire analysis because of the multiplicity of ways that individual systems can interact. Many risk analysts find it easier to develop one fault tree for each system and to use an event tree representation to model the interactions between systems. These event trees can be constructed using Sandia's SANET event tree graphics software. In such a model, each event in the event tree represents the success or failure of one system. Each path through the event tree is called a *sequence*, and represents one possible combination of system successes and failures that can occur in the analysis. The analyst examines each sequence to determine whether it leads to a favorable or an unfavorable result. Each "unfavorable" sequence is solved to determine which cut sets contribute to the unfavorable condition.

Solving Sequences. Typically there are several sequences in each event tree that lead to unfavorable outcomes. In the interest of minimizing the amount of effort required to find cut sets for these sequences, the following method has been developed.

1. Before sequence solution begins, solve all of the individual system fault trees individually for cut sets.
2. For each sequence, use a logical AND operation to combine the cut sets for all of the systems that are found to be in the "failed" state on the event tree path that defines the sequence. Thus, the unfavorable sequence outcome is only achieved when all of these systems fail.
3. Consider the systems that are found to be in the "successful" state on the event tree path that defines the sequence. Since these systems are *known* to be successful (by the definition of the sequence), we must assure that none of the cut sets for our sequence will inadvertently imply that this system has failed. This is done by deleting any cut sets from the sequence that would make such an implication.

Solving sequences using the Sandia fault tree analysis software is a three part process.

1. SEATREE and SABLE are used to develop and solve the initial system fault trees for system cut sets.
2. SANET is used to develop the event trees that describe the interactions between the systems (i.e., to develop the "sequence logic").
3. The SEQUENCE code takes the cut sets from SABLE and the sequence logic from SANET and performs the solution algorithm described above using the SABLE analysis engine.

Using Fault Tree Results in an Event Tree Analysis. It is not unusual for a question in an event tree analysis to ask, "Does system X work?" It makes sense to find the probability for such an event using a fault tree model. To accomplish this one would simply construct and solve the fault tree model "off-line" and transfer the result into the event tree analysis using SANET. If a point estimate event tree analysis is being performed and the systems are independent, one can simply quantify the fault tree in SABLE and copy the single value into the SANET event tree model input. If the event tree is being analyzed as part of an uncertainty analysis, one would perform an uncertainty analysis on the fault tree results using LHS and TEMAC3 to obtain uncertainty input for the event tree question. Sandia's SETAC event tree analysis software reads TEMAC3 output directly, so it is not necessary to manually transfer this data into the event tree model.

3 Outline of the SABLE Interface

This overview of SABLE explains how SABLE is constructed, the purpose of the Analysis Summary Screen, how the Solve button works, and explains various file handling capabilities of SABLE.

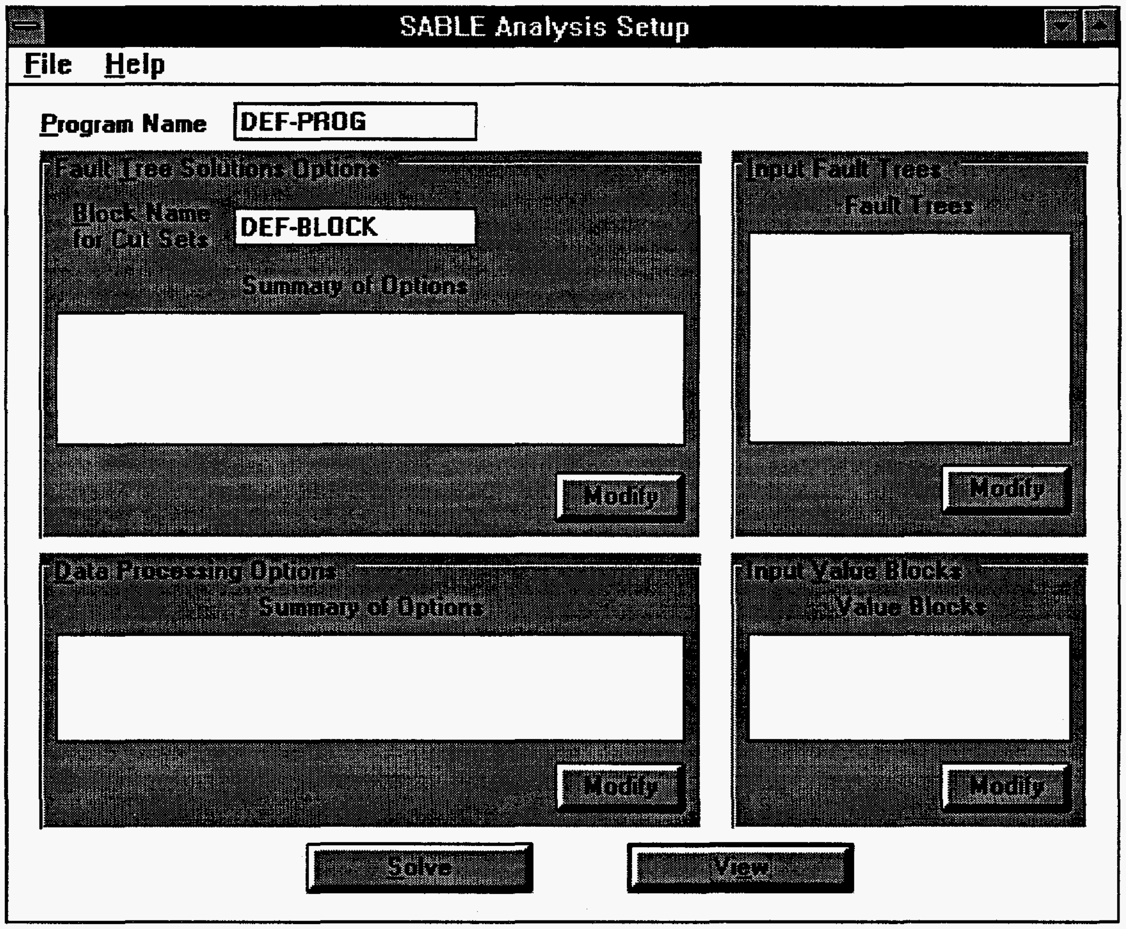
3.1 What SABLE Does

SABLE is designed to provide a friendly facility for solving fault trees. It takes as its input ASCII-formatted files that contain fault tree logic and quantification information and processes those files to obtain cut set solutions. At the user's option, these cut sets can be quantified and printed in order of decreasing frequency.

SABLE consists of two major parts: the Windows user interface and the FORTRAN analysis engine. The interface is designed to provide a friendly environment to specify input files, fault trees and solution methods, values and quantification schemes, and data processing and output options. The analysis engine provides a powerful tool to solve and quantify fault trees and generate reports of the results. The interface writes files containing SABLE user code which provide instructions for the SABLE FORTRAN analysis engine to execute in solving the designated fault trees.

3.2 Analysis Summary Screen

The Analysis Summary Screen, shown in Figure 3-1, is the first screen which appears when you invoke the SABLE interface.



Program Name: DEF-PROG

Block Name for Cut Sets: DEF-BLOCK

Figure 3-1: The Analysis Summary Screen

This screen gives an overview of the currently selected options, which include the fault trees and value blocks selected and all solution, quantification, and reporting options. Options cannot be changed on this screen. You must press the Modify buttons to bring up detail screens. The only changes that can be made on this screen are the program and block names. You can use this screen to browse the summary information to determine whether the currently selected options will cause SABLE to perform the fault tree solution that you desire. When you are satisfied with the options that you have selected, press the "Solve" button to cause SABLE to run the fault tree analysis. You may then view the results by pressing the "View" button.

3.3 The Solve Button

The Solve button is located at the bottom of the Analysis Summary Screen. When then Solve button is pressed, SABLE performs the following operations:

- SABLE checks the options and names specified by the user to assure that they are complete and consistent.

- SABLE writes a "user code" file that contains instructions for the analysis engine.
- SABLE invokes the analysis engine and waits in the background for it to finish (the analysis engine runs in a DOS window).
- SABLE checks for run-time errors in the analysis engine output.

If a run time error occurs, a diagnostic message is displayed. SABLE will tell you where the results and/or error files can be found for viewing and printing.

3.4 File Handling

The following options are available through the File menu on the Analysis Setup screen.

Starting a New Analysis. When you first enter SABLE, you are ready to start a new analysis. If you have already selected options and wish to start over with a new analysis, select "File | New" from the menu on the Analysis Setup screen. A verification screen will appear; you may either continue with the existing analysis or start a new one.

Opening a Pre-existing Analysis. To open a pre-existing analysis, select "File | Open" from the menu on the Analysis Setup screen. A standard Windows dialog box will allow you to browse through drives, directories, and files until you select the file you wish to open. Files containing a SABLE analysis usually have a *.PGM extension.

Saving an Analysis. To save an existing analysis, select "File | Save" from the menu on the Analysis Setup screen. If you have previously named the analysis, the file will be saved automatically. If you have not previously named the analysis, a standard Windows dialog box will allow you to specify the name of the analysis and its path.

If you wish to save an analysis under a different name, select "File | Save As" from the menu on the Analysis Setup screen. A standard Windows dialog box will allow you to specify the name of the analysis and its path.

Viewing a File. To view a file, either select "File | View" from the menu on the Analysis Setup screen or press the "View" button on the bottom of the Analysis Setup screen. Both actions will activate a standard Windows dialog box, allowing you to select a file for viewing.

Setting the Default Directory. The default directory is the starting location for file searches and specifications in the standard Windows dialog box. To set the default working directory, select "File | Default Directory" from the menu on the summary screen. A screen will appear which allows you to scroll through the directories and disk drives on your machine. Double-click on the desired directory and press "OK" to specify the new default directory.

4 Fault Trees

This section describes the various types of files that can contain fault trees and how to specify fault trees by invoking the Fault Tree Selection screen.

4.1 Types of Files that can Contain Fault Trees

There are three types of files that can contain fault trees. They can be found in any combination.

SEATREE Files. These files usually end with a file extension of *.SET. Only one fault tree exists in each file. You can select fault trees from multiple files to be solved in a single SABLE run.

ASCII Files. These files are created manually using an ASCII text editor to enter fault tree logic. There may be many fault trees in each file, and trees from multiple files may be solved in a single SABLE run.

SABLE Block Files. These binary files usually have a file extension of *.BLK or *.SBF. They may contain many fault trees in each file. SABLE can only use one block file in each run. If you select fault trees to be solved from a SABLE block file, SABLE will not allow you specify a different block file name on the "Data Processing Options" screen.

4.2 Fault Tree Selection Screen

You must tell SABLE where to find the fault trees to solve. From the Analysis Setup screen, go to the area of the screen marked "Input Fault Trees" and press the "Modify" button. This will bring up the Fault Tree Selection screen, shown in Figure 4-1.

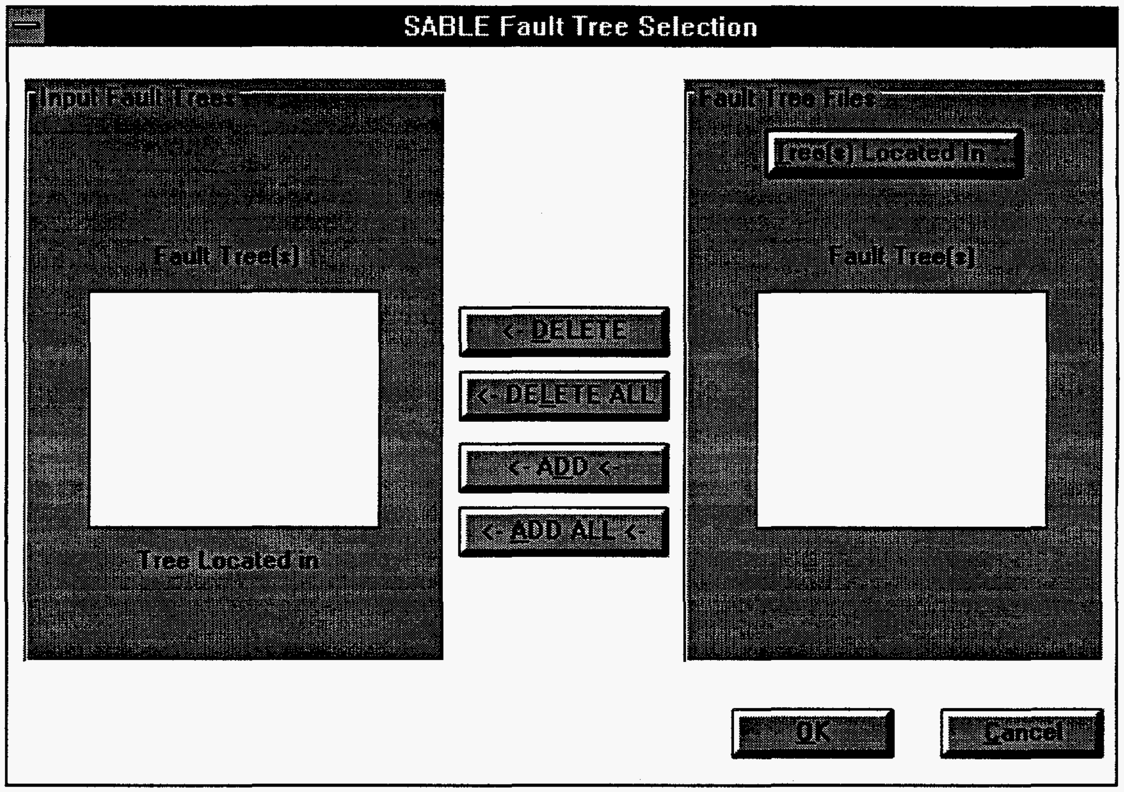


Figure 4-1: The Fault Tree Selection Screen

This screen contains two columns. The column on the left contains the names of the fault trees that have already been selected for analysis. Unless a name is deleted from the column, SABLE will attempt to solve it. The column on the right contains the names of all fault trees in the currently selected file. The buttons between the columns allow you to either add fault trees to the solve list or delete fault trees from the solve list.

Selecting a Fault Tree File. To select a fault tree file, press the "Tree(s) Located In . . ." button above the right hand column. This will bring up a Windows common dialog box for selecting an input file.

Adding Fault Trees to the List. Once you have selected a file, the names of the fault trees in the file are listed in the column on the right. To select a tree to solve, either double click the tree name, or select it and press the "Add" button. To add all the trees in the file, press the "Add All" button. The name(s) of the selected tree(s) will be placed in the column on the left.

Deleting Fault Trees from the List. To remove a fault tree from the solve list, either double click the name in the column on the left, or select the name and press the "Delete" button. To remove all fault trees from consideration, press the "Delete All" button.

Seeing Fault Tree File Names. To see the name of the input file that contains a specific fault tree, select the fault tree in the left hand column. The file name corresponding to the selected tree appears below the column containing the list of selected fault trees.

5 Quantification Values

This section describes the various types of files that contain quantification information, how to include value blocks using the Value Block Selection screen, and how to create and/or edit a value block using the SABLE Value Block Editor.

5.1 Types of Files that Can Contain Quantification Information

Four main file types can contain value block information. Value blocks can be in any combination of files from the following forms.

SEATREE Files. These files have a file extension of *.EDT. They usually contain only one value block per file. Value blocks from multiple SEATREE, ASCII, and LHS files may be used in a single SABLE run.

ASCII Files. These files are created manually using an ASCII text editor to enter quantification data. They may have many value blocks per file. Value blocks from multiple SEATREE, ASCII, and LHS files may be used in a single SABLE run.

LHS Files. LHS files have only one value block per file. Value blocks from multiple SEATREE, ASCII, and LHS files may be used in a single SABLE run.

SABLE Value Block Files. These binary files usually have an extension of *.VAL or *.SVF. They may contain multiple value blocks in each file. SABLE can use only one value block file in each run. If you select value blocks from a SABLE value block file, SABLE will not allow you to specify a different value block file name on the "Data Processing Options" screen.

5.2 Value Block Selection Screen

For SABLE to quantify your analysis, you must specify one or more value blocks. From the Analysis Setup screen, go to the area marked "Input Value Blocks" and press the "Modify" button. This will bring up the Value Block Selection screen, shown in Figure 5-1.

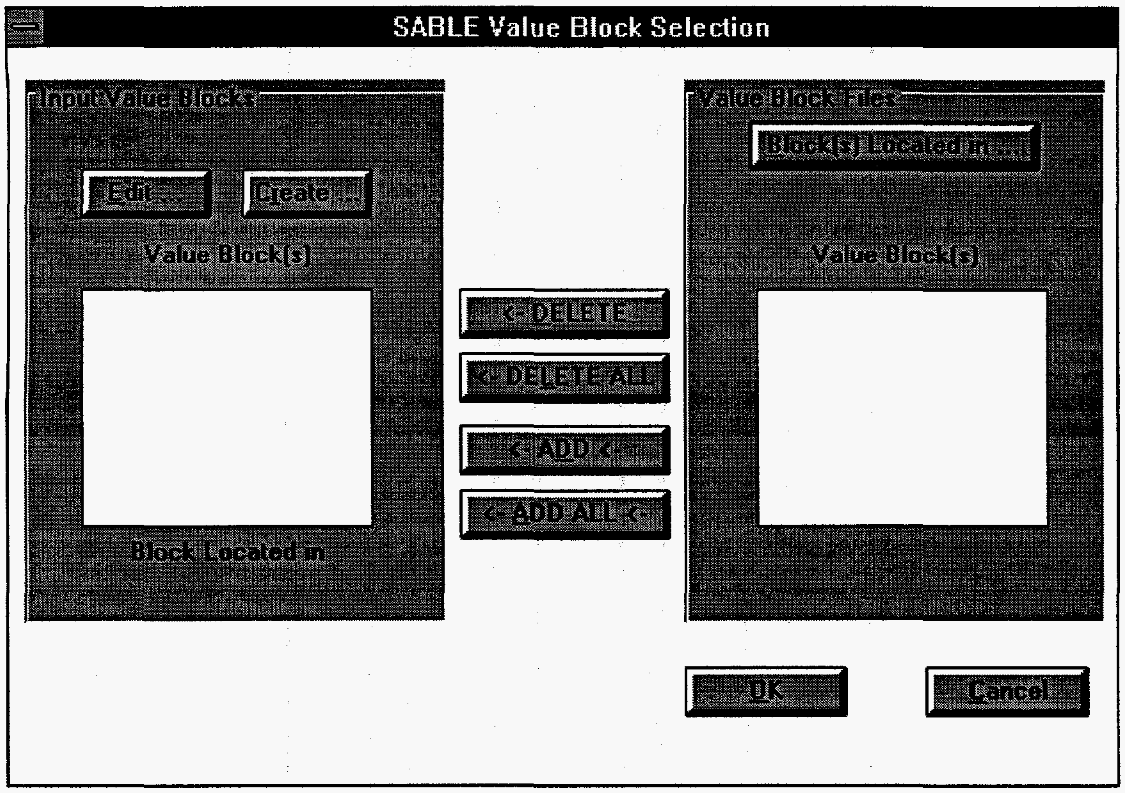


Figure 5-1: The Value Block Selection Screen

This screen is similar to the Fault Tree Selection screen. The value blocks to be processed are listed in the left column, and the value blocks in the currently selected file are listed in the right column.

Selecting a Quantification Input File. To select a value block file, press the "Block(s) Located In . . ." button above the right hand column. This will bring up a Windows common dialog box for selecting an input file.

Adding Value Blocks to the List. Once you have selected a file, the names of the value blocks in the file are listed in the column on the right. To select a value block for use in cut set quantification, either double click the value block name, or highlight (select) it and press the "Add" button. To add all the blocks in the file, press the "Add All" button. The name(s) of the selected block(s) will be placed in the column on the left.

Deleting Value Blocks from the List. To remove a value block from the input list, either double click the name in the column on the left or select the name and press the "Delete" button. To remove all blocks from consideration, press the "Delete All" button.

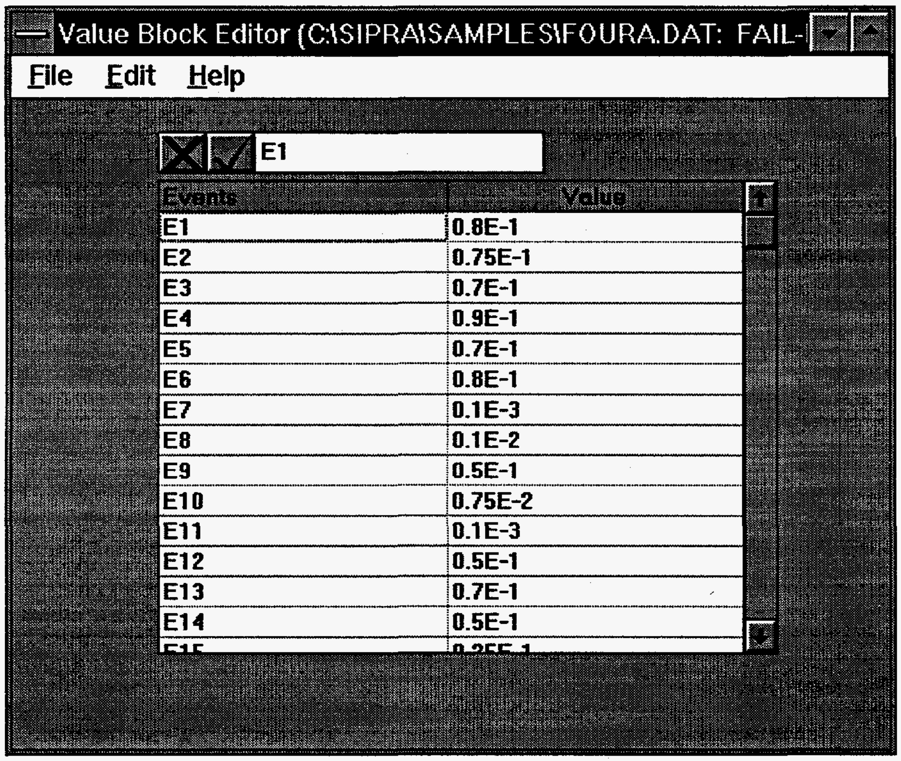
Seeing Value Block File Names. To see the input file that contains a specific value block, select the block in the left hand column. The file name corresponding to the selected value block appears below the column containing the list of selected blocks.

5.3 The SABLE Value Block Editor

The SABLE Value Block Editor is accessed from the "Create" and "Edit" buttons on the Value Block Selection Screen.

The "Create" button is clicked to begin the process of building a new value block. A screen will appear for you to provide a value block name that follows the SABLE naming conventions. Pressing "OK" will bring up the SABLE Value Block Editor Screen, shown in Figure 5-2; a "Cancel" button is provided if you wish to abort the creation of a new value block.

To edit an existing value block, you must first select a value block from the list of value blocks listed on the left-hand side of the Value Block Selection Screen and then click the "Edit" button. Editing occurs in a spreadsheet-like format, with each row containing an event name and the value for that event.



Events	Value
E1	0.8E-1
E2	0.75E-1
E3	0.7E-1
E4	0.9E-1
E5	0.7E-1
E6	0.8E-1
E7	0.1E-3
E8	0.1E-2
E9	0.5E-1
E10	0.75E-2
E11	0.1E-3
E12	0.5E-1
E13	0.7E-1
E14	0.5E-1

Figure 5-2: Value Block Editor Screen

Whether you are creating or editing a value block, the process of maneuvering in the SABLE Value Block Editor is the same.

5.3.1 The Editing Box

The editing box, located above the grid on the Value Block Editor screen, is the location for naming events, defining values, and changing text. To the left of this box are two buttons: the "Cancel" button ("X") and the "OK" button (check mark). The Cancel button will cancel any changes made in the editing box; the OK button will save any changes to the current selection you are modifying. Pressing "Enter" or "Return" will also activate the OK button. The Value Block Editor will check your entry after you have pressed the OK button to ensure that the name or value conforms to SABLE standards.

5.3.2 Inserting Event Names and Values

To insert a new event, select "Edit | Insert" from the menu on the Value Block Editor screen. A small dialog box, shown in Figure 5-3, will appear asking how many values you wish to add.

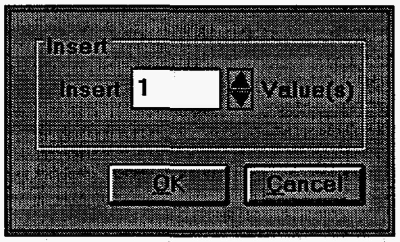


Figure 5-3: Insertion Dialog Box

Either type the number of values you wish to insert or increase/decrease the number with the spinner on the right. The default value is "1". Press "OK" to insert the row(s) or press "Cancel" to cancel the operation.

You may also access the editing window, shown in Figure 5-4, by clicking the right mouse button when on the grid.

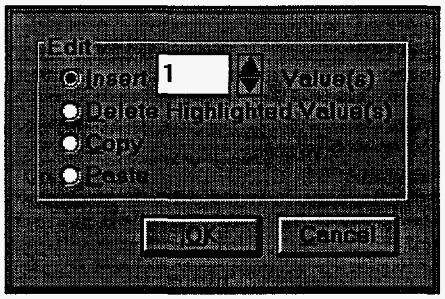


Figure 5-4: Editing Window

Either type the number of values you wish to insert or increase/decrease the number with the spinner on the right. The default value is "1". Press "OK" to insert the value(s) or press "Cancel" to cancel the operation.

Both of the above operations will insert blank rows. You must then type the event name and event value using the editing box. To enter the name and value, select the grid entry and type the information; the text will be sent to the editing box.

5.3.3 Changing the Name or Value of an Existing Event

To change an existing name or value, click on the cell containing the information you wish to change. The text in the grid will be placed in the editing box. You may either start typing, which will delete the text in the editing box, or place the cursor in the editing box where you wish to begin typing. *NOTE: The text in the grid will not change unless you press the OK button or press "Return".*

5.3.4 Deleting Events

To delete an event and its associated value from the value block, highlight the event you wish to delete and select "Edit | Delete" from the menu bar. You may select multiple events. A confirmation message will appear ensuring you wish to delete the highlighted event(s).

You may also use the Editing window to delete an event(s). Select the event(s) you wish to delete and press the right mouse button. The Editing window will appear; select "Delete Highlighted Value(s)" and press "OK".

Note: If a value is already included in a value block FILE, deleting an event will not erase the value IN THE FILE since SABLE will ignore values for events that do not exist in the fault tree. Therefore, the delete option is mainly used as an editing tool when adding and changing values in a block. There is no undo option for deletions or value changes!

5.3.5 Copying an Event Value

To copy an event value, highlight the row that contains the value you wish to copy and select "Edit | Copy" from the menu on the Value Block Editor screen. The value will be copied into an internal buffer and will remain there until you copy another value.

You may also use the Editing window to copy a value. Select the value you wish to copy and press the right mouse button. The Editing window will appear; select "Copy" and press "OK".

5.3.6 Pasting an Event Value

To paste an event value, select the row(s) where you wish to paste a value and select "Edit | Paste" from the menu on the Value Block Editor screen. The value will be pasted into the selected row(s).

You may also use the Editing window to paste a value. Select the location of the value you wish to paste and press the right mouse button. The Editing window will appear; select "Paste" and press "OK".

NOTE: You must first copy a value before pasting.

5.3.7 Finding an Event Name or Value

To find an event name or value, select "Edit | Find" from the menu on the Value Block Editor screen. The Find dialog box, shown in Figure 5-5, will appear.

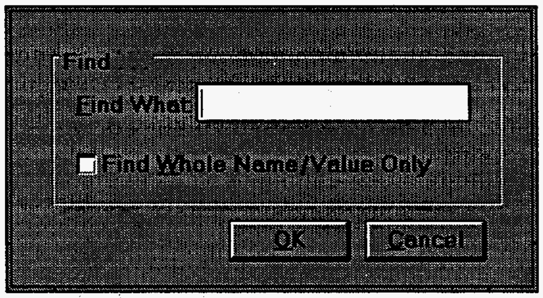


Figure 5-5: Find Dialog Box

Type the text string you wish to find. If you select the "Find Whole Name/Value Only," the program will search for only whole words or values matching what you have typed. The search process begins at the currently selected row. If the text is found, the row containing the text will be highlighted. If no occurrence is found, you will be asked if you wish to start searching from the beginning of the value block.

If you have selected multiple events (by highlighting a group of rows with the mouse), the program will only search in the highlighted section. If no occurrence of the text is found, you will be asked if you wish to search the entire block.

5.3.8 Data Format Type

The data associated with each event in a SABLE Value Block can be displayed in either scientific or floating pointing format.

The scientific format can be in two forms:

(1) X.XXXXXXE-XX
or
(2) XE-XX

Floating point entries must be of the form:

X.XXXXXXX

The leading digit is required even if it is zero.

Some examples of the formats for scientific and floating point values are:

1E-5
1.5E-10
0.257595E-2
1.0
0.5
0.12345

Some examples of invalid formats are:

.5 *(no leading zero)*
1.E-2 *(no number following the decimal point)*
1.01 *(value greater than one)*

5.3.9 Saving the Value Block and Returning to SABLE

To save the changes, select "File | Save" from the menu. From the File menu, choose "Exit" to return to SABLE when you have completed your editing.

6 How Fault Trees are Solved

This section discusses the theory of fault tree solution and how to select various options pertaining to the solution of the selected fault trees.

6.1 Theory of Fault Tree Solution

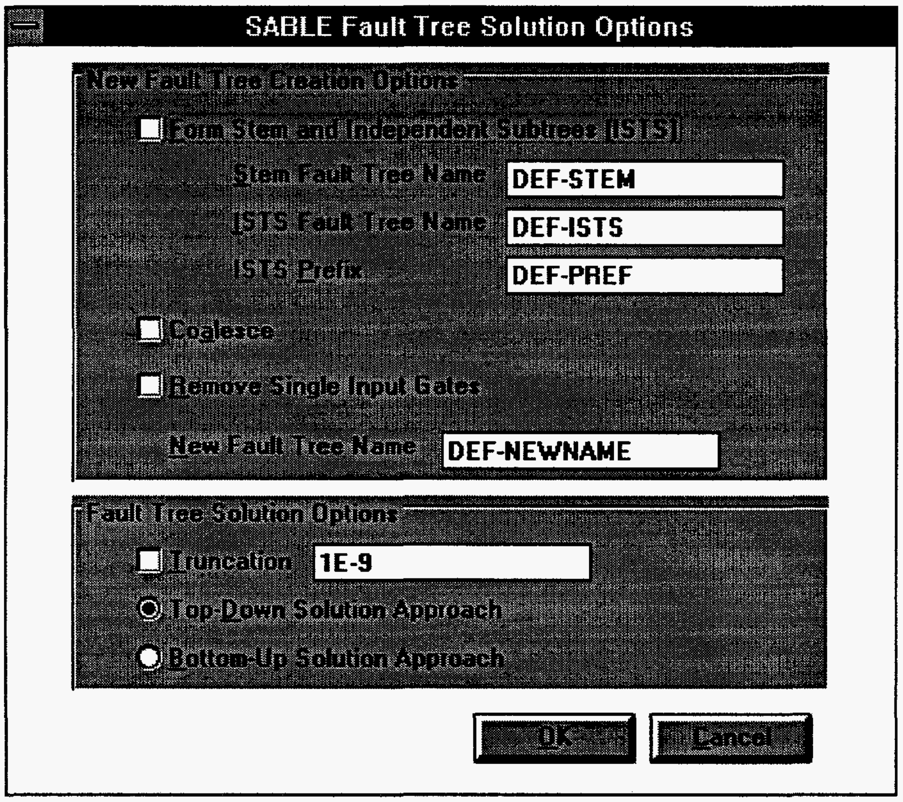
The SABLE fault tree analysis code takes several steps in preparing fault trees for solution and executing the solution process. First, SABLE brings all of the fault trees to be solved into memory at the same time. This allows SABLE to assemble large trees from constituent pieces and to check the trees for consistency. If one tree has a "developed event" with the same name as a gate in another tree, SABLE assumes that these are different representations of the same event and substitutes the actual gate logic for the developed event. SABLE also assures that, if there is more than one development for a particular event name, all of the developments are consistent.

Once the fault trees are assembled and checked, SABLE can change the structure of the fault tree to increase solution speed. In this process, which is controlled through user input, single input gates can be removed from the trees. The trees may also be modified through a process known as "coalescing." The coalescing process looks for cascaded gates of the same type and replaces them with a single gate. Thus, if SABLE finds an AND gate whose children are also all AND gates, it replaces the AND gate *and its children* with a logically equivalent structure that contains a single AND gate. The same process occurs for OR gates. Transfer gates, i.e. a multiple occurrence event, cannot be coalesced.

The final step SABLE can take in preparing fault trees for solution is to identify and remove independent subtrees (ISTS). An IST is a subtree whose basic events are found only within that subtree (i.e., no primary event in the subtree exists anywhere in any of the fault trees except in that subtree, hence it is "independent" of the rest of the fault tree structure). SABLE prunes the IST from the fault tree stems and replaces them with developed events. This allows SABLE to work on the stem portion separately from the ISTS and reduces the computational complexity of the problem dramatically.

Now that the fault trees are prepared, SABLE uses the following method to accomplish the fault tree solution. First, SABLE solves any IST, quantifies the IST results (if necessary), and saves the cut sets and values for later use. SABLE then solves and, if necessary, quantifies the stem fault trees using only the values for the IST developed events. Finally, SABLE will substitute the IST cut sets into the stem cut sets to obtain the final global cut set expressions for the fault trees. Note that the user may optionally tell SABLE not to perform this final substitution if they are comfortable reading cut sets that contain IST names instead of cut sets that contain only primary events.

You must tell SABLE how the fault trees will be solved. From the Analysis Setup screen, go to the area of the screen marked Fault Tree Solution Options and press the Modify button. Figure 6-1 shows the Fault Tree Solution Options Screen.


SABLE Fault Tree Solution Options

New Fault Tree Creation Options

[ ] Form Stem and Independent Subtrees (ISTS)

Stem Fault Tree Name: DEF-STEM

ISTS Fault Tree Name: DEF-ISTS

ISTS Prefix: DEF-PREF

[ ] Coalesce

[ ] Remove Single Input Gates

New Fault Tree Name: DEF-NEWNAME

Fault Tree Solution Options

[ ] Truncation: 1E-9

[✓ Top-Down Solution Approach]

Bottom-Up Solution Approach

Figure 6-1: The Fault Tree Solution Options Screen

6.2 Form Stem and Independent Subtrees

SABLE will have an easier time solving larger fault trees if the independent subtrees (ISTS) are pruned from the stem fault tree and solved separately. To cause this to occur, you must select the "Form Stem and Independent Subtrees" option on the Fault Tree Solution Options screen. If this option is selected, you must also select to remove single input gates. You must also provide a stem fault tree name, an IST fault tree name, and an IST prefix. Default names are provided.

6.3 Coalesce Gates

Selecting the "Coalesce" option on the Fault Tree Solution Options screen will combine cascading AND gates and cascading OR gates to form equivalent logical structures containing fewer gates. If this option is selected, you must also select to remove single input gates.

6.4 Remove Single Input Gates

Selecting the "Remove Single Input Gates" option on the Fault Tree Solution Options screen will cause SABLE to restructure the fault tree to remove all gates that do not have more than one input. This reduces the number of gates in the fault tree and allows SABLE to solve it in a more efficient manner. This option must be selected if "Form Stem and Independent Subtrees" is selected. If you want to remove single gates without forming ISTS, you must specify a name for the fault tree created by this process. A default name is provided.

6.5 Truncation

Even simple fault trees can sometimes generate a large number of cut sets. Many of these cut sets will have values (probabilities or frequencies) that are extremely small and, hence, are of little value to the analyst. In order to both speed the solution process and reduce the amount of clutter in the program output, an analyst may wish to delete (truncate) all cut sets that do not meet a particular minimum value criterion.

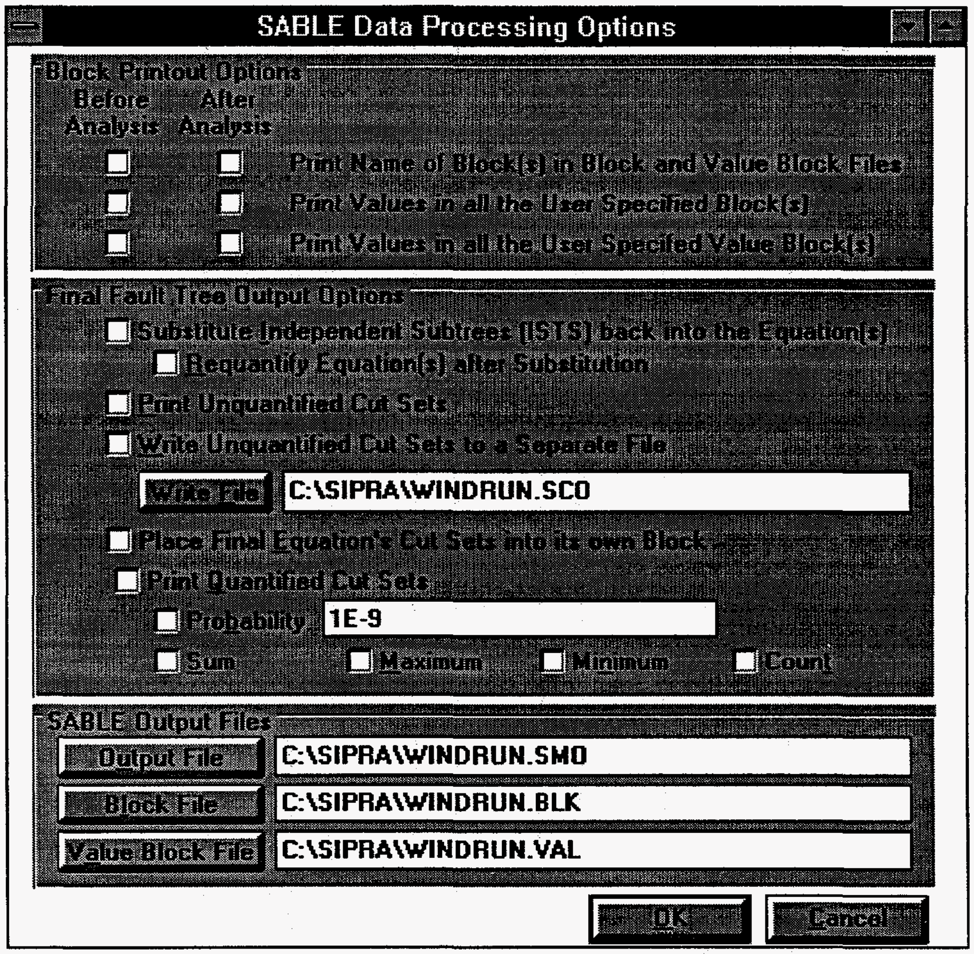
To enable cut set truncation, select the "Truncation" option check box on the Fault Tree Solution Options screen and enter the probability or frequency below which cut sets are not to be retained. Truncation is performed based on the value calculated for each cut set individually. Any cut set with a value less than the value specified will be truncated. Note that, since truncation is based on cut set quantification, you must provide quantification information (a value block) if you want SABLE to perform truncation.

6.6 Top-Down Versus Bottom-Up Solution

An objective for SABLE is to keep the number of intermediate cut sets small in order to dramatically reduce the solution time. Each solution method (top-down vs. bottom-up) causes SABLE to solve the fault tree in smaller pieces to keep the number of intermediate cut sets small. Selecting top-down causes SABLE to solve the top piece of the tree first and work its way to the bottom in stages. Selecting bottom-up causes SABLE to solve pieces near the bottom of the tree first and to work its way up to the top in stages. It is difficult to predict which method will work better for a particular problem, but bottom-up is often best for a first attempt to solve a large tree. It is also usually preferred for a tree with many OR gates. Generally, a tree with many AND gates near the top is better suited for the top-down solution method. You must select one or the other, but not both. By default the program will use a top-down solution method.

7 Data Processing Options

You must tell SABLE which files to use and what types of information are to be included in the program output. These are specified on the Data Processing Options screen. From the summary screen, go to the area of the screen marked "Data Processing Options" and press the Modify button. This brings up the Data Processing Options screen, which is shown in Figure 7-1.


SABLE Data Processing Options

Block Printout Options

Before Analysis	After Analysis	
☐	☐	Print Name of Block[s] in Block and Value Block Files
☐	☐	Print Values in all the User Specified Block[s]
☐	☐	Print Values in all the User Specifed Value Block[s]

Final Fault Tree Output Options

☐ Substitute Independent Subtrees [ISTS] back into the Equation[s]

☐ Requantify Equation[s] after Substitution

☐ Print Unquantified Cut Sets

☐ Write Unquantified Cut Sets to a Separate File

Write File: C:\SIPRA\WINDRUN.SCO

☐ Place Final Equation's Cut Sets into its own Block

☐ Print Quantified Cut Sets

☐ Probability: 1E-9

☐ Sum ☐ Maximum ☐ Minimum ☐ Count

SABLE Output Files

Output File: C:\SIPRA\WINDRUN.SMO

Block File: C:\SIPRA\WINDRUN.BLK

Value Block File: C:\SIPRA\WINDRUN.VAL

Figure 7-1: Data Processing Options Screen

7.1 Block Printout Options

These options allow you to take an inventory of the contents of the block file and the value block file at the beginning of a SABLE run, at the end of a run, or both. The "Print Name of Blocks . . ." option on the Data Processing Options screen prints out a directory list containing the *names of the blocks* in the block and value block files. The

"Print Values . . ." options print the *actual fault tree, cut sets, or values* that reside in the block files. The "Print Values . . ." options can produce large output files when big fault trees are being used. You may select to print any of these options before and/or after the analysis has been completed.

7.2 Substitution of Independent Subtrees

This option may only be exercised if the fault tree solution option "Form Stem and Independent Subtrees" was selected. By default, SABLE leaves the cut sets for the "stem" fault trees in terms of IST developed event names (it does not expand these IST developed events to include their constituent primary events and cut sets). To have SABLE generate cut sets in terms of only primary events, check the "Substitute Independent Subtrees . . ." option on the Data Processing Options screen. You may also requantify the equations after substitution by selecting the "Requantify Equation(s) . . ." option. Since ISTS are independent, there is no need to requantify; the results of the initial calculation are already correct. However, if the ISTS were solved without truncation (or using a different truncation value), requantification will produce uniform truncation of all cut sets.

7.3 Print or Write Unquantified Cut Sets

To include the unquantified cut sets in the program output, select either the "Print Unquantified Cut Sets" option or the "Write Unquantified Cut Sets to a Separate File" option on the Data Processing Options screen. The cut sets are printed as a disjunctive normal form (DNF) of the top event equation. DNF is the fully expanded form of a cut set equation (it contains no parenthesis). The events in each individual cut set are separated by AND operators (typically "*" symbols), and the cut sets are separated from one another by OR operators (typically "+" symbols). Thus, it is a "sum of products" representation of the minimal cut sets. Printing the DNF to the standard output file is for the analysts to see. Writing the equations to a separate file is used generally by other programs, such as TEMAC3. In each case, the cut sets are ordered such that cut sets containing only one term are placed first, followed by cut sets containing two terms, and so forth.

7.4 Place Cut Set Results in their Own Block

When SABLE is asked to solve multiple fault trees simultaneously, it usually places all of the cut set expressions for all of the fault trees into a single entry in the block file. For most analysts this is OK, and it allows SABLE to execute a little faster. However, when the SANET and SEQUENCE codes are used to generate cut sets for event sequences, a different procedure must be used. The SEQUENCE software uses the cut set results from SABLE and combines them using the logic from event trees generated by SANET. SEQUENCE assumes that the minimal cut sets for each fault tree in the event sequence are stored in a separate block that is named after the top event in the fault tree whose cut sets it contains. For example, if a SANET event sequence model is based on fault trees with top events named T1, T2 and T3, then SEQUENCE assumes that the fault trees with these top events have been solved by SABLE, and that their minimal cut sets reside in separate blocks within the SABLE block file, and that these blocks are named T1, T2 and T3, respectively.

In order to facilitate the linkage between these codes, an option has been installed in SABLE to allow the cut sets from different fault trees to be placed into their own separate blocks following the SEQUENCE naming convention. This is accomplished by selecting the "Place Final Equation's Cut Sets into its own Block" option on the Data Processing Options screen. When this option is selected, each cut set expression is placed in its own block, and each block name is the same as the name of the top event in the cut set expression.

7.5 Print Quantified Cut Sets

In order for SABLE to show you any quantification information for the cut sets it generates, you must select the "Print Quantified Cut Sets" option on the Data Processing Options screen. This option will place the quantified cut set values in the standard output file. Cut set values can be calculated by any or all of several methods:

Probability. The cut set value is the product of the primary event values. You may also specify a lower limit for values; this number is a truncation value for printing purposes alone.

Sum. The cut set value is the sum of the primary event values.

Maximum. The cut set value is the maximum value among the primary events.

Minimum. The cut set value is the minimum value among the primary events.

Count. The cut set value is the number of primary events in the cut set.

Note that quantification input must be present in order for these quantification requests to be carried out.

7.6 SABLE Output Files

SABLE can write information to four different files: a standard output file for analyst viewing, a cut set file for input to other codes, a block file, and a value block file.

Output File. This file contains program output for user viewing. It is an ASCII-formatted file containing all program output intended for user viewing. The file name may be specified; by default, the name is WINDRUN.SMO. You may either type in a file name or press the button "Output File" on the Data Processing Options screen to bring up a standard Windows dialog box.

Write File. This file contains information SABLE wants to pass on to another code. It is an ASCII file containing cut sets or values for transfer to other software. The file name may be specified; by default, the name is WINDRUN.SCO. You may either type in a file name or press the button "Write File" on the Data Processing Options screen to bring up a standard Windows dialog box.

Block File. This file is like a hard disk for storing fault trees, Boolean equations, and cut sets. It is a binary file containing SABLE blocks (fault trees, cut sets, and equations). The file name may be specified; by default, the name is WINDRUN.BLK. You may either type in a file name or press the button "Block File" on the Data Processing Options screen to bring up a standard Windows dialog box.

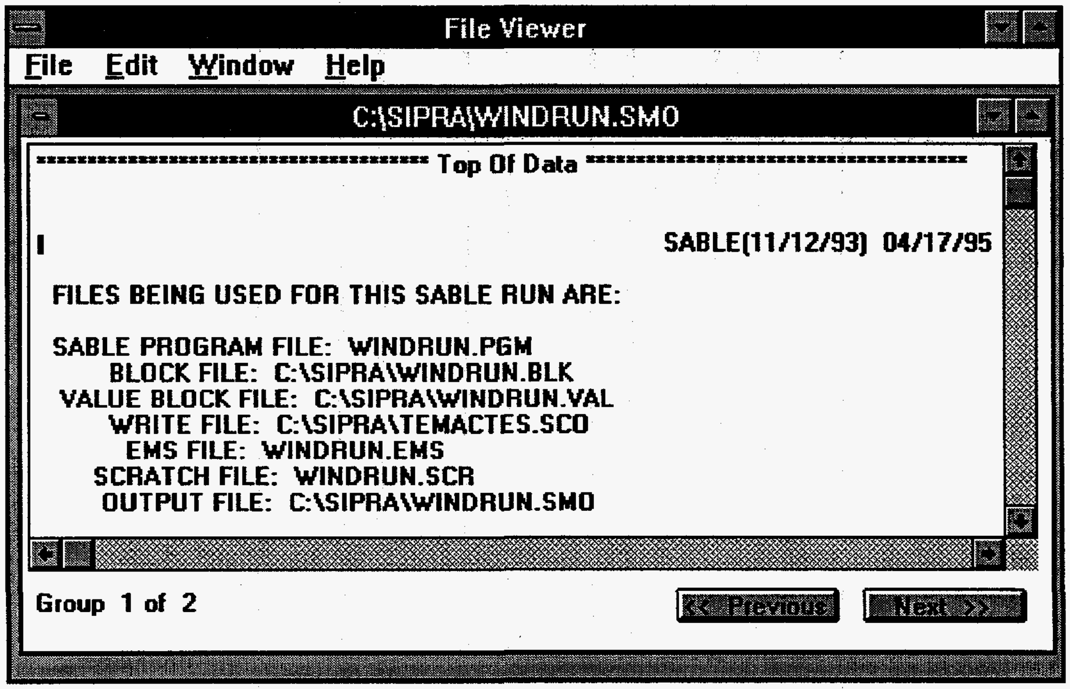
Value Block File. This file is like a hard disk for storing event values. It is a binary file containing SABLE value blocks (quantification values). The file name may be specified; by default, the name is WINDRUN.VAL. You may either type in a file name or press the button "Value Block File" on the Data Processing Options screen to bring up a standard Windows dialog box.

8 Viewing SABLE Output with the View Utility

Once the program has run, you may view your output file by either pressing the "View" button on the bottom of the summary screen or selecting "File | View" from the menu. This activates the Sandia Integrated Probabilistic Risk Assessment (SIPRA) File Viewer utility, which was designed specifically to allow a user to view and move quickly through very large ASCII-formatted files. Using this utility you can select portions of the file to be printed or placed on the Windows clipboard to be pasted into other Windows applications.

8.1 Purpose of the View Utility

The View Utility is a multiple-document interface (MDI) that allows the user to view large ASCII-formatted text files. In this program the user may print, change the font, search for text, and copy to the Windows clipboard. No editing is allowed. The View Utility screen with an open file is shown in Figure 8-1.



```
************************************** Top Of Data **************************************

1                                                          SABLE(11/12/93) 04/17/95

 FILES BEING USED FOR THIS SABLE RUN ARE:

 SABLE PROGRAM FILE:  WINDRUN.PGM
        BLOCK FILE:  C:\SIPRA\WINDRUN.BLK
  VALUE BLOCK FILE:  C:\SIPRA\WINDRUN.VAL
        WRITE FILE:  C:\SIPRA\TEMACTES.SCO
          EMS FILE:  WINDRUN.EMS
      SCRATCH FILE:  WINDRUN.SCR
       OUTPUT FILE:  C:\SIPRA\WINDRUN.SMO
```

Figure 8-1: The View Utility

Depending on the size of the file, it may be broken into groups. The groups are approximately 32 Kbytes in size. When the document is first brought up, the first group is shown. To proceed to the next group, push the "Next" button below the text screen. Likewise, to return to a previous group, select the "Previous" button.

8.2 Maneuvering in a Multiple-Document Interface

Viewing more than one file at a time is possible with a multiple-document interface. When multiple files are being viewed, each file populates its own window. The title of each window (the top bar) is the path and name of the file.

A window is active (the current window) after you have selected that window with the mouse.

Minimization. To reduce the window to an icon, press the button in the upper right hand corner of the window with the down arrow.

Maximization. To expand the window to take up the full screen, press the button in the upper right hand corner of the window with the up arrow. To make the window occupy less than the entire screen once again, press the button in the upper right hand corner of the window with both the up and down arrows.

Resizing. To resize a window, position the mouse at any corner or side. When the pointer becomes a double-headed arrow, move the arrow to the desired size with the mouse button depressed. The expandable border will be visible as a dotted gray line. Release the mouse button when you have finished resizing.

Window Manipulation. You can expand any window or cause it to revert to an icon by selecting the appropriate item from the Window menu. From the keyboard, press Alt-W to activate the Window menu, then use the appropriate hot key (Alt plus an underlined letter) to activate or deactivate the desired window. You may arrange multiple documents from the "Window" menu item. The document windows can be cascaded or tiled horizontally or vertically. When the windows are reduced to icons, you may line them up evenly with the "Arrange Icons" option.

8.3 Opening Files

To open a file for viewing in the View Utility, select the "File" menu item and then the "Open" option. A dialog box will appear allowing you to select the path and name of the file you wish to view. After selecting the file name, a window with the contents of the file is displayed.

To open multiple files, follow the same procedure for each file to be viewed. A separate window is created for each file.

8.4 Closing Files

To close a file, select the window containing the file you wish to close and choose the "Close" option from the "File" menu. You may also double click on the control button in the upper left-hand corner of the window. The View Utility will ask you for confirmation before it closes the file and deletes its window.

8.5 Printing

To print the file in the active window, select the "Print" option from the "File" menu. A dialog box will appear with options for printer setup, number of copies, and printing quality.

To print a selected portion of the file, highlight the desired text before you select the Print option from the menu. When the dialog box is displayed, be sure that "Selection" is chosen before you press "OK".

You may also print the current page of text by choosing the "Print Current Page" option from the "File" menu. On completion of this function, the page that was printed remains highlighted in the current window.

If you have changed the font of a document, the new font is used in printing. Otherwise, the default printer font is used.

NOTE: A selected page is based on form feed characters or FORTRAN carriage controls, not on the length of a standard page. If you try to print a selected page and you have Landscape as the printing orientation instead of Portrait, the selected page will probably not fit on only one printed page. The same is true if a large font has been selected for the current window. You may want to adjust the font size and/or the printer page orientation to compensate.

A known deficiency in Visual Basic exists. The sequence (to change the printer setup) of "File | Print | Printer Setup" does not work unless a change has been made and saved using the sequence "File | Printer Setup." Thus, to change the settings for the printer, select "File | Printer Setup" the first time.

8.6 The Edit Menu

The edit menu allows for copying, finding text, and font selections. No alterations to the file can be made; you may only view the file.

Copy. To copy a section of the current file to the Windows Clipboard, highlight the desired portion with the mouse and select "Copy" from the "Edit" menu. You may then paste this text into another Windows application for further processing.

Selecting the Current Page. To select the current page, make sure that the cursor is placed somewhere in the page you wish to select and choose "Edit | Select Current Page". This page will be highlighted.

Selecting the File Type. When you open a document, the program automatically identifies the file type by the presence of either form feed characters or FORTRAN carriage controls. If you wish to see what type of file you have opened or you would like to change the type of file, choose the "Select File Type" option from the "Edit" menu. A dialog box explaining which file type with an option to change the type will appear.

Finding Text. To find a selection of text, select the "Find" option from the "Edit" menu. A dialog box will appear; type the text string that you wish to find. This routine is case-insensitive. The next occurrence (after the last current mouse position in the document) will be highlighted.

The "Find Next" routine will find the next occurrence of the last text string searched for. This option can also be accessed with the F3 button.

If you searching in a document with multiple groups and no occurrence is found in the current group, you will be asked if you wish to continue the search in the next group. If an occurrence is not found before the end of the document, you will be asked if you wish to continue the search from the beginning of the document.

Selecting a Font. To change the font type and/or size, select the "Select Font" from the "Edit" menu. You may select any font available on your computer and the size. Any change you select will apply to the entire document. You may, however, select different fonts for different documents if desired.

If you select a font for a document, any printing from that document will also be performed in this font.

9 Additional Topics

9.1 Legal SABLE Names

A SABLE name must meet the following requirements: it may not exceed 16 characters (12 characters for a "prefix"), it may not contain embedded blank spaces, and it may contain only *upper case* letters, numbers, and the special characters dash, underscore, colon, or backward slash (-, _, :, \). SABLE file names may be any valid DOS file name. SABLE will check all names for validity.

9.2 Function of OK and Cancel

Pressing the OK button on any screen will save the information on the current screen and summarize it on the main screen. The Cancel button will revert to the "old" information before alterations were made.

9.3 Keyboard and Mouse Operations

For those who prefer using a keyboard to a mouse, several options are built into this program to allow flexibility between the two input devices.

If a caption on an option (such as a button or a menu option) has a letter that is underlined, pressing that letter and the ALT key simultaneously is equivalent to clicking the mouse on that option. For example, pressing "ALT - S" is the same as clicking Solve on the summary screen with the mouse.

The Tab key will move you forward around the screen to all valid options. The combination of the Shift and Tab keys will move you backward around the screen. Pressing the Enter Key when positioned on a button will press the button. Pressing the space bar when positioned on a check box will check the box.

9.4 Getting Help

Five help options are available in SABLE, four of which are available through the "Help" menu on the Analysis Setup Screen.

To learn how to use help files in a Windows environment, select "Help | How to Use Help" from the menu on the Analysis Setup screen.

To get a listing of all help topics for SABLE, select "Help | Contents" from the menu on the Analysis Setup screen.

To get help by searching for a specific keyword relative to SABLE, select "Help | Search for Help On . . ." from the menu on the Analysis Setup screen.

To obtain a quick overview of the SABLE interface, select "Help | SABLE Quick Start" from the menu on the Analysis Setup screen.

If you need help on a current screen or control, pressing "F1" will activate the help files and display the portion related to the current screen or control.

9.5 Leaving Suggestions

If you have suggestions for this program, select "Help | Suggestion Box" from the menu. This will activate a screen , Figure 9-1, allowing you to enter your comments. Pressing "Place in Suggestion Box" will save your comments in a text file.

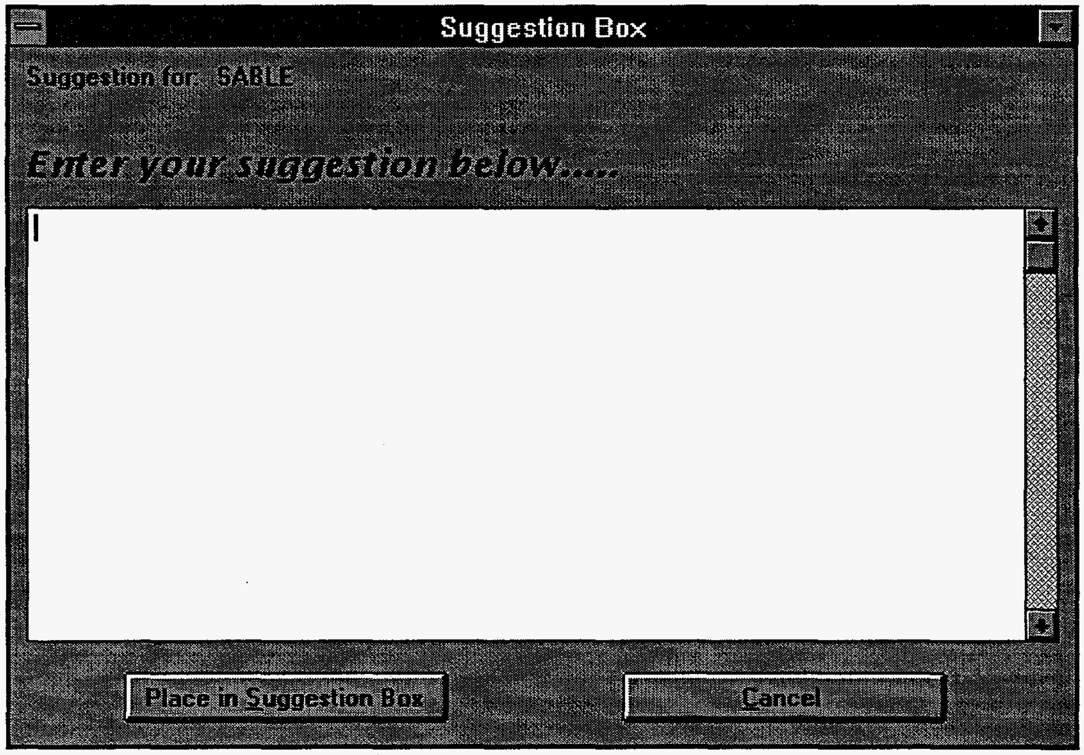


Figure 9-1: The Suggestion Screen

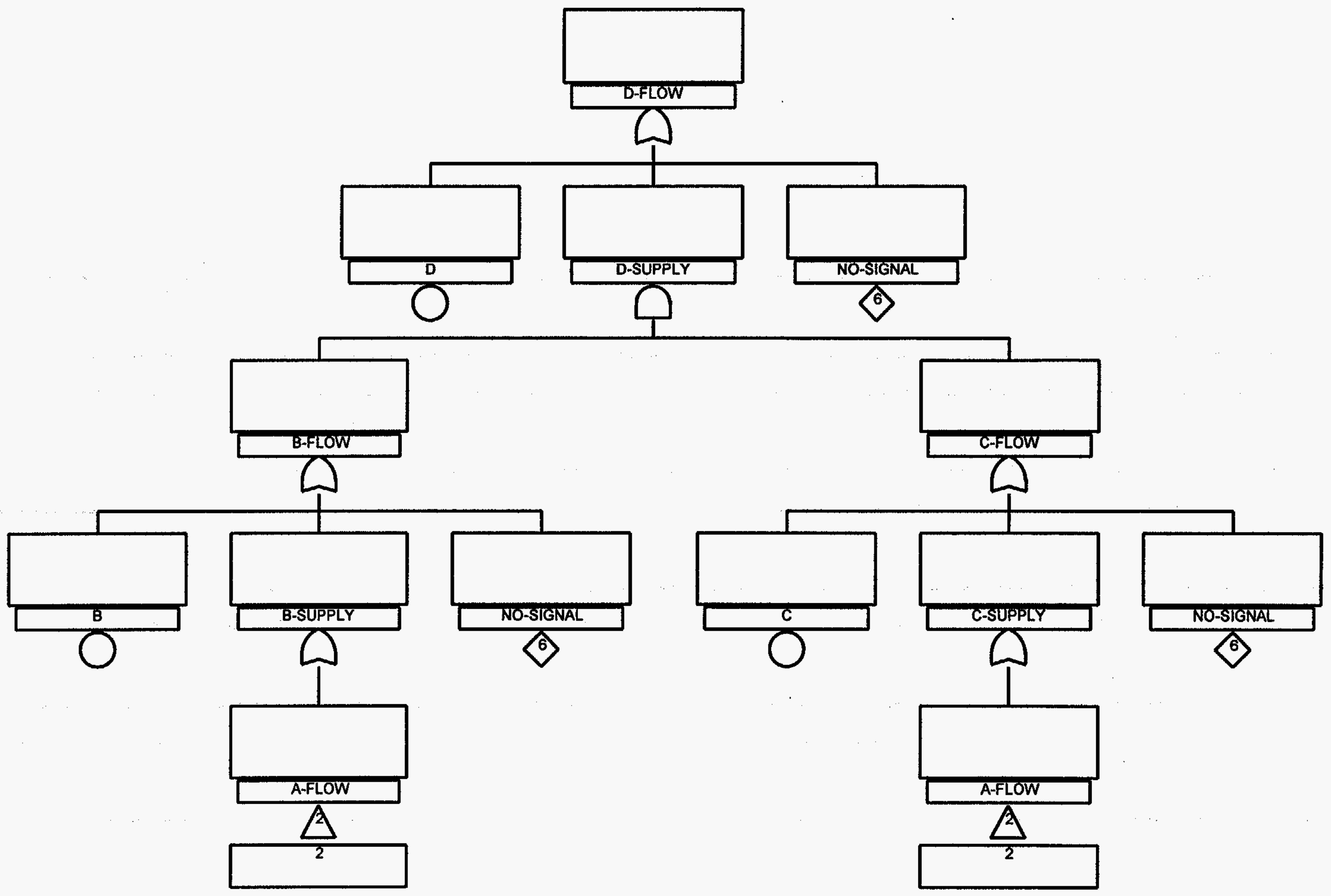
Appendix A: A Sample Problem

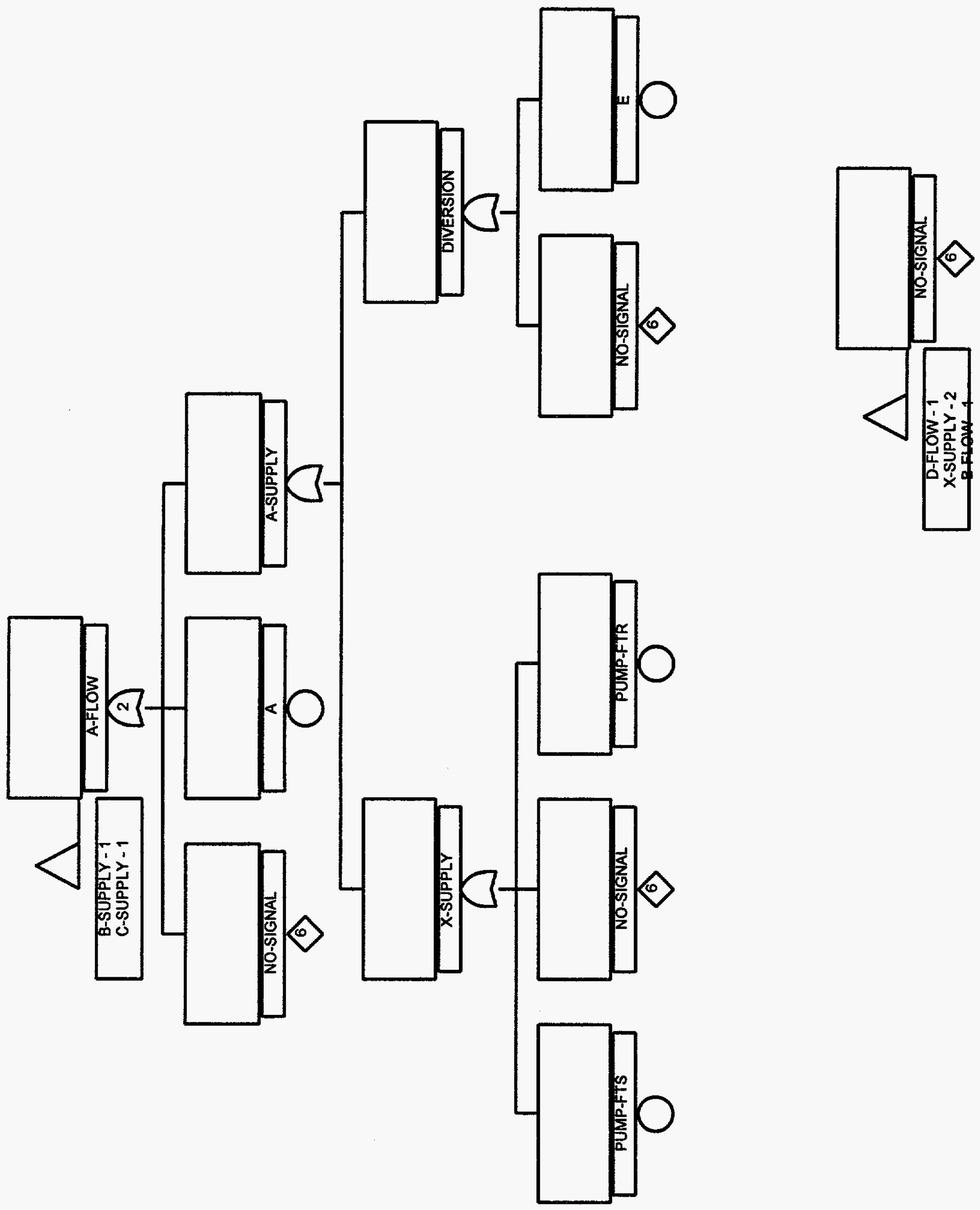
This appendix will take you through a sample problem by showing all stages of solving a fault tree.

A.1 Graphing the Fault Tree

Using SEATREE 2.6, the fault tree (titled PLT1-EX) was graphed and saved in the file PLT1-EX.SET (for input to SABLE) and PLT1-EX.TRE (as a native SEATREE file). Figure A-1 shows the list view of the fault tree, Figure A-2 shows the galley view of the fault tree, and Figure A-3 shows the ASCII PLT1-EX.SET file.

```
1 │△ D-FLOW { }
1 │ └⌓ D-FLOW.
1 │     ├○ D.
1 │     ├⌓ D-SUPPLY.
1 │     │   ├⌓ B-FLOW.
1 │     │   │   ├○ B.
1 │     │   │   ├⌓ B-SUPPLY.
2 │     │   │   │   └△ ⌓ A-FLOW.
2 │     │   │   └△ ◇ NO-SIGNAL.
1 │     │   └⌓ C-FLOW.
1 │     │       ├○ C.
1 │     │       ├⌓ C-SUPPLY.
2 │     │       │   └△ ⌓ A-FLOW.
2 │     │       └△ ◇ NO-SIGNAL.
2 │     └△ ◇ NO-SIGNAL.
2 │△ A-FLOW { B-SUPPLY, C-SUPPLY }
2 │ └⌓ A-FLOW.
2 │     ├△ ◇ NO-SIGNAL.
2 │     ├○ A.
2 │     └⌓ A-SUPPLY.
2 │         ├⌓ X-SUPPLY.
2 │         │   ├○ PUMP-FTS.
2 │         │   ├△ ◇ NO-SIGNAL.
2 │         │   └○ PUMP-FTR.
2 │         └⌓ DIVERSION.
2 │             ├△ ◇ NO-SIGNAL.
2 │             └○ E.
2 │△ NO-SIGNAL { D-FLOW, X-SUPPLY, B-FLOW, C-FLOW, A-FLOW, DIVERSION }
2 │ └◇ NO-SIGNAL.
```


D-FLOW
D
D-SUPPLY
NO-SIGNAL
6
B-FLOW
C-FLOW
B
B-SUPPLY
NO-SIGNAL
6
C
C-SUPPLY
NO-SIGNAL
6
A-FLOW
2
2
A-FLOW
2
2


A-FLOW
2
B-SUPPLY - 1
C-SUPPLY - 1
NO-SIGNAL
6
A
A-SUPPLY
X-SUPPLY
DIVERSION
PUMP-FTS
NO-SIGNAL
6
PUMP-FTR
NO-SIGNAL
6
E
NO-SIGNAL
6
D-FLOW - 1
X-SUPPLY - 2
B-FLOW - 1

```
FAULT TREE$PLT1-EX.
OG$ D-FLOW.
IN$D,NO-SIGNAL,D-SUPPLY.
OG$ X-SUPPLY.
IN$PUMP-FTS,PUMP-FTR,NO-SIGNAL.
OUT$A-SUPPLY.
UE$ NO-SIGNAL.
OUT$D-FLOW,X-SUPPLY,B-FLOW,C-FLOW,A-FLOW,DIVERSION.
OG$ A-FLOW.
IN$NO-SIGNAL,A-SUPPLY,A.
OUT$B-SUPPLY,C-SUPPLY.
BE$ D.
OUT$D-FLOW.
AG$ D-SUPPLY.
IN$B-FLOW,C-FLOW.
OUT$D-FLOW.
OG$ B-FLOW.
IN$B,NO-SIGNAL,B-SUPPLY.
OUT$D-SUPPLY.
BE$ B.
OUT$B-FLOW.
OG$ C-FLOW.
IN$C,NO-SIGNAL,C-SUPPLY.
OUT$D-SUPPLY.
BE$ C.
OUT$C-FLOW.
BE$ PUMP-FTS.
OUT$X-SUPPLY.
OG$ A-SUPPLY.
IN$X-SUPPLY,DIVERSION.
OUT$A-FLOW.
OG$ B-SUPPLY.
IN$A-FLOW.
OUT$B-FLOW.
OG$ DIVERSION.
IN$NO-SIGNAL,E.
OUT$A-SUPPLY.
OG$ C-SUPPLY.
IN$A-FLOW.
OUT$C-FLOW.
BE$ E.
OUT$DIVERSION.
BE$ PUMP-FTR.
OUT$X-SUPPLY.
BE$ A.
OUT$A-FLOW.
EOF
```

Figure A-3: “PLT1-EX.SET”, Fault Tree Input to SABLE

A.2 Developing Quantification Information

Using SEATree 2.6, a value block named "FAIL-VAL" containing values for each event was constructed. An ASCII view of this data is shown below in Figure A-4.

```
VALUE BLOCK$ FAIL-VAL.
1.0E-3$ A$
1.0E-5$ B$
1.0E-2$ C$
1.0E-2$ D$
1.0E-2$ E$
1.0E-3$ PUMP-FTS$
1.0E-3$ PUMP-FTR$
1.0E-3$ NO-SIGNAL$
EOF
```

Figure A-4: "PLT1-EX.EDT", Value Block Input to SABLE

A.3 Creating an Analysis inside SABLE

To solve this fault tree, the following options were selected.

Fault Tree Selection. The fault tree "PLT1-EX" was imported from the file "PLT1-EX.SET". Figure A-5 shows the Fault Tree Selection screen after the fault tree was imported.

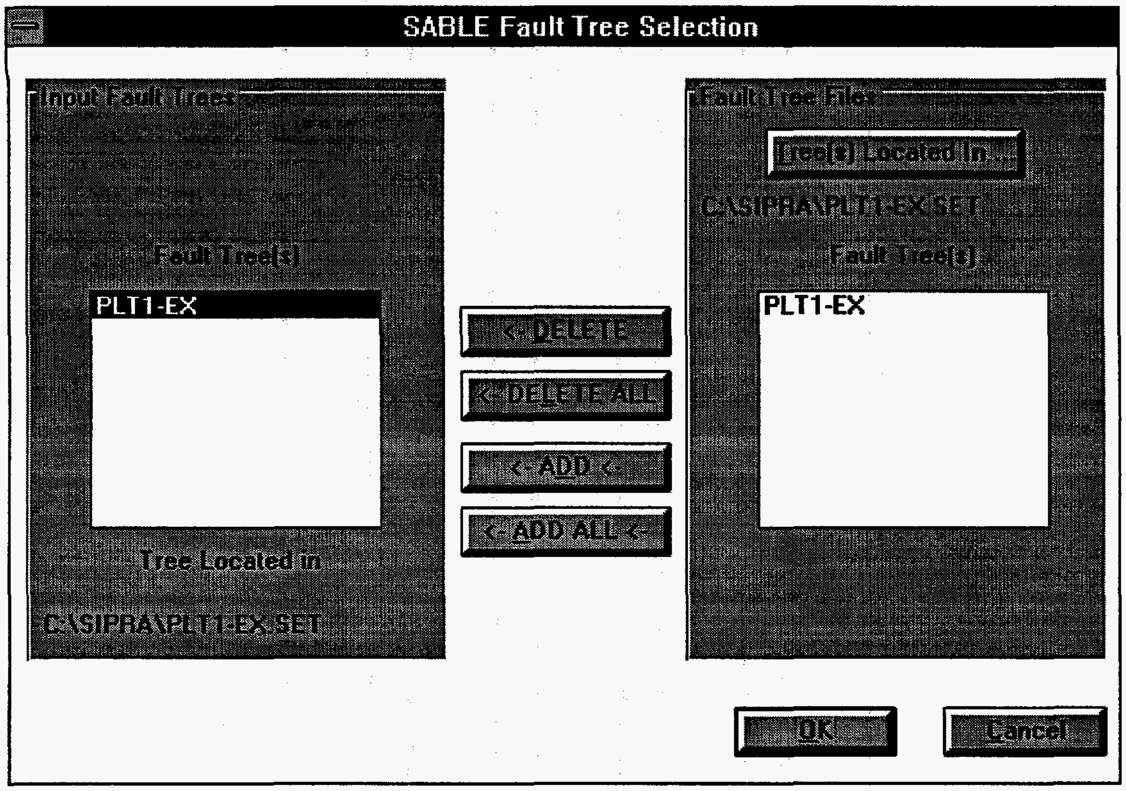


Figure A-5: Fault Tree Selection Screen after importing "PLT1-EX"

Value Block Selection. The value block "FAIL-VAL" was imported from the file "PLT1-EX.EDT" for quantification. Figure A-6 shows the Value Block Selection Screen after selecting the value block.

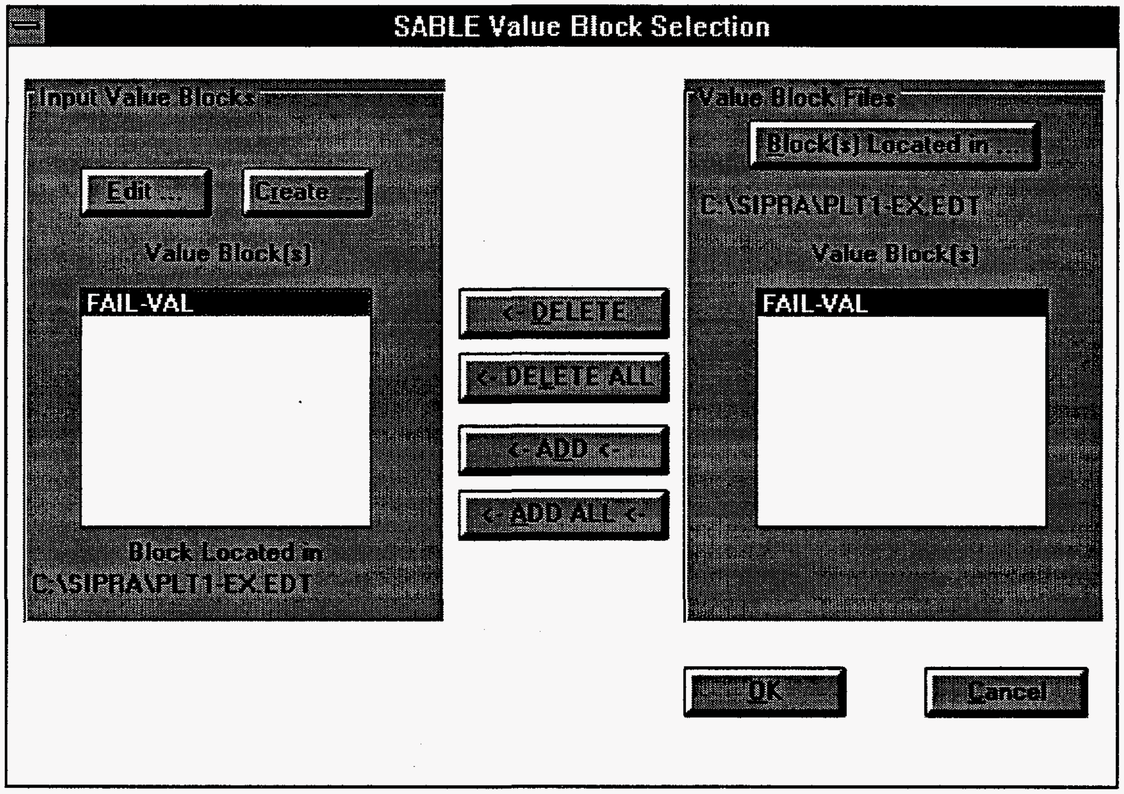
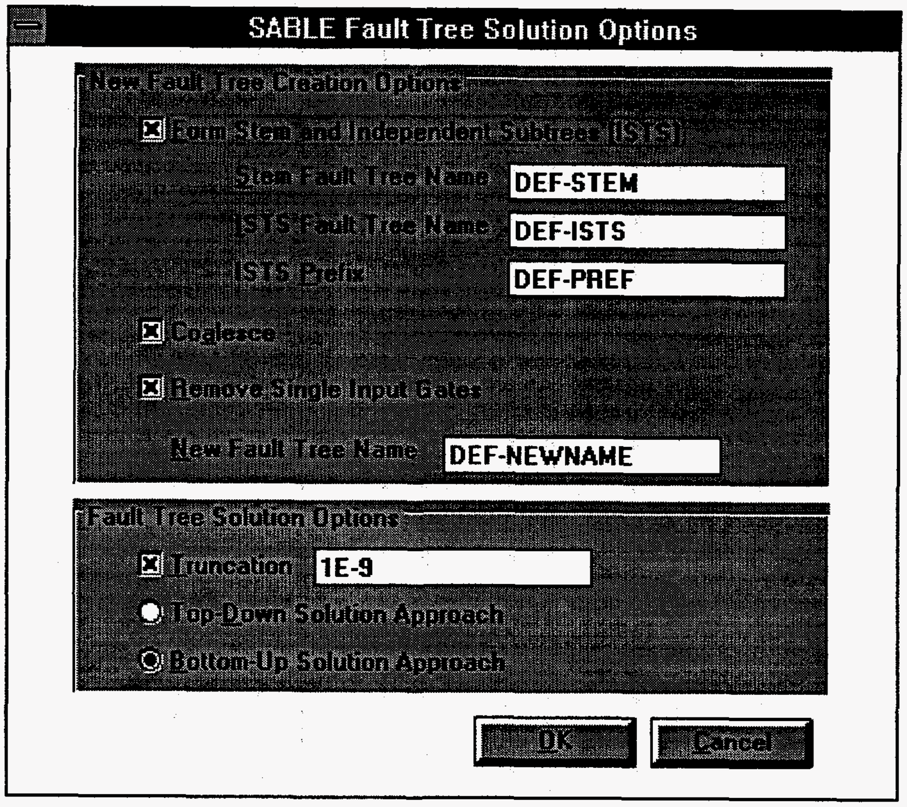


Figure A-6: Value Block Selection Screen after importing "FAIL-VAL"

Fault Tree Solution Options. The following solution options were selected to solve this fault tree:

- Form stem and independent subtrees with a stem fault tree name of "DEF-STEM", an ISTS fault tree name of "DEF-ISTS", and an ISTS prefix of "DEF-PREF".
- Coalesce gates.
- Remove single input gates.
- Truncate cut sets below a value of 1E-9.
- Solve the fault tree with the bottom - up solution method.

Figure A-7 shows the Fault Tree Solution Options screen after the above selections were made.


SABLE Fault Tree Solution Options

New Fault Tree Creation Options

[✓] Form Stem and Independent Subtrees (ISTS)

Stem Fault Tree Name: DEF-STEM

ISTS Fault Tree Name: DEF-ISTS

ISTS Prefix: DEF-PREF

[✓] Coalesce

[✓] Remove Single Input Gates

New Fault Tree Name: DEF-NEWNAME

Fault Tree Solution Options

[✓] Truncation: 1E-9

○ Top-Down Solution Approach

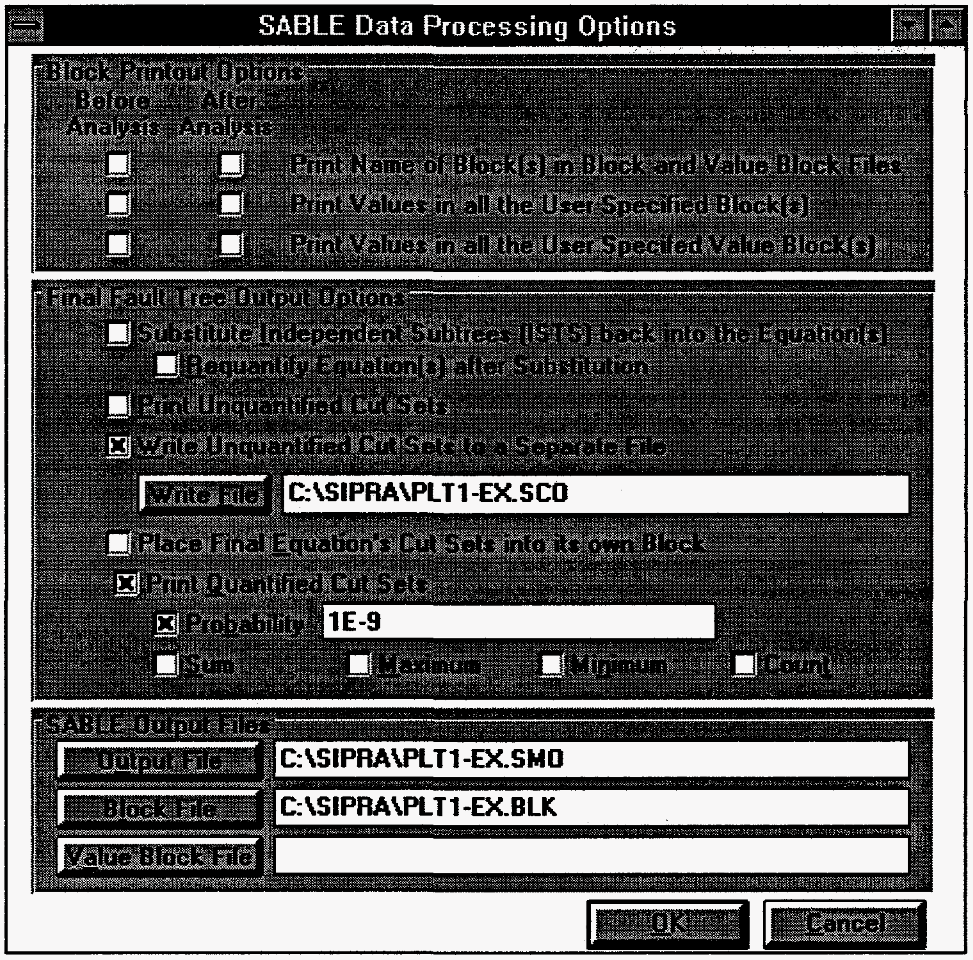
● Bottom-Up Solution Approach

Figure A-7: Fault Tree Solution Options Screen.

Data Processing Options. The following data processing options were selected for this analysis:

- Write the unquantified cut sets to the file "PLT1-EX.SCO".
- Print the quantified cut sets in the output file, truncating at 1E-9.
- Write the SABLE general output to "PLT1-EX.SMO".
- Write the SABLE block file to "PLT1-EX.BLK".

Figure A-8 shows the Data Processing Options screen after selecting the above options.



SABLE Data Processing Options

Block Printout Options

Before Analysis	After Analysis	
☐	☐	Print Name of Block(s) in Block and Value Block Files
☐	☐	Print Values in all the User Specified Block(s)
☐	☐	Print Values in all the User Specifed Value Block(s)

Final Fault Tree Output Options

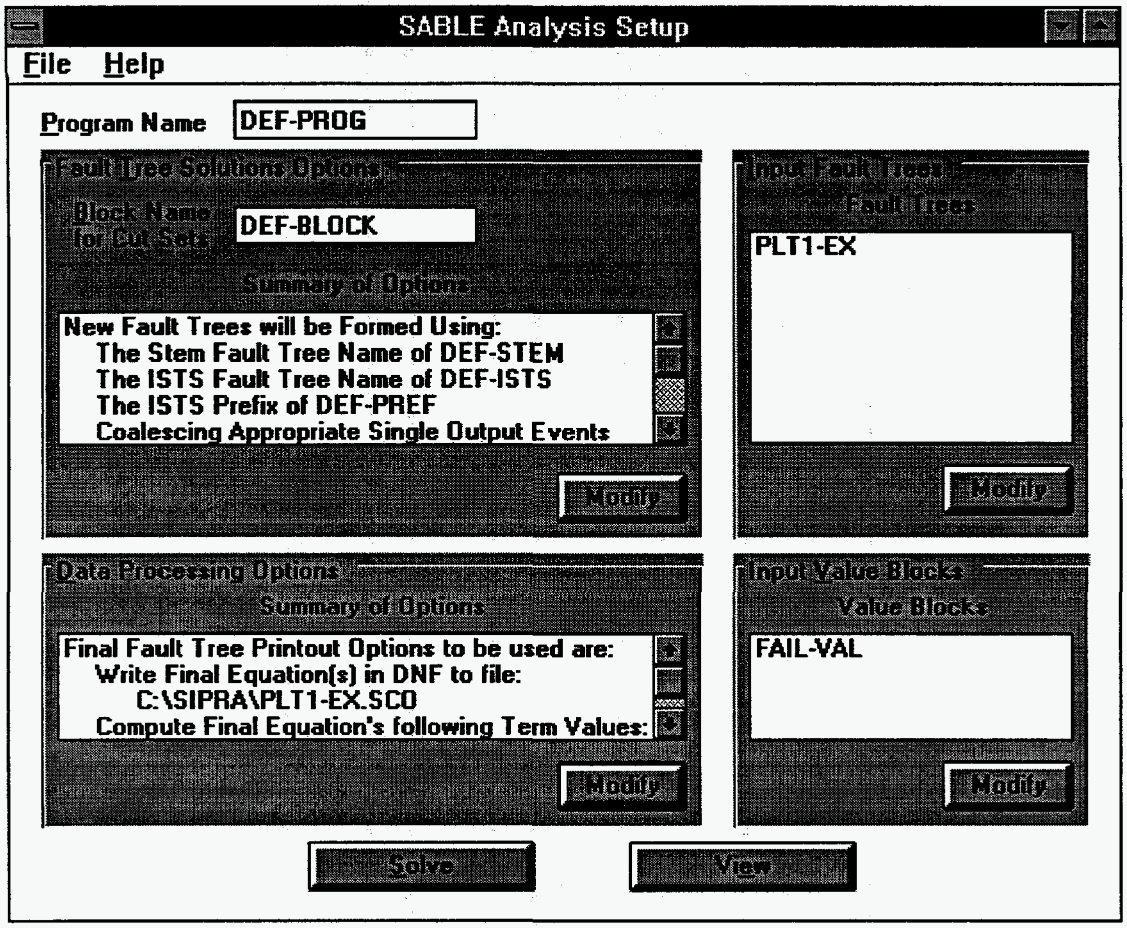
☐ Substitute Independent Subtrees (ISTS) back into the Equation(s)
☐ Requantify Equation(s) after Substitution
☐ Print Unquantified Cut Sets
☒ Write Unquantified Cut Sets to a Separate File
Write File: C:\SIPRA\PLT1-EX.SCO
☐ Place Final Equation's Cut Sets into its own Block
☒ Print Quantified Cut Sets
☒ Probability: 1E-9
☐ Sum ☐ Maximum ☐ Minimum ☐ Count

SABLE Output Files

Output File: C:\SIPRA\PLT1-EX.SMO
Block File: C:\SIPRA\PLT1-EX.BLK
Value Block File:

Figure A-8: Data Processing Options screen after selecting options

Analysis Setup Screen. After pressing "OK" on the above screens, the corresponding summary boxes on the Analysis Screen were populated accordingly. Figure A-9 shows the Analysis Setup screen after the above-mentioned options were selected.



Program Name DEF-PROG

Block Name for Cut Sets DEF-BLOCK

New Fault Trees will be Formed Using:
The Stem Fault Tree Name of DEF-STEM
The ISTS Fault Tree Name of DEF-ISTS
The ISTS Prefix of DEF-PREF
Coalescing Appropriate Single Output Events

PLT1-EX

Final Fault Tree Printout Options to be used are:
Write Final Equation(s) in DNF to file:
C:\SIPRA\PLT1-EX.SCO
Compute Final Equation's following Term Values:

FAIL-VAL

Figure A-9: Analysis Setup screen after selecting all options

A.4 SABLE Output

After the above analysis was run, two user-readable output files were produced: "PLT1-EX.SMO", the standard SABLE output file, and "PLT1-EX.SCO", the file containing the unquantified cut sets to be used as input to other PRA codes such as TEMAC3. Figure A-10 shows the contents of "PLT1-EX.SMO".

```
  FILES BEING USED FOR THIS SABLE RUN ARE:

  SABLE PROGRAM FILE:  WINDRUN.PGM
         BLOCK FILE:  C:\SIPRA\PLT1-EX.BLK
   VALUE BLOCK FILE:  WINDRUN.VAL
         WRITE FILE:  C:\SIPRA\PLT1-EX.SCO
           EMS FILE:  WINDRUN.EMS
       SCRATCH FILE:  WINDRUN.SCR
        OUTPUT FILE:  C:\SIPRA\PLT1-EX.SMO
BEGIN EXECUTION OF SABLE.


      READ SABLE USER PROGRAM.

                  PROGRAM$ DEF-PROG.
                  COMMENT$ BEGINNING OF WINDOWS RUN INFORMATION

                           NUMBER OF FAULT TREES TO READ:    1
                           FAULT TREE: PLT1-EX
                              LOCATED: C:\SIPRA\PLT1-EX.SET

                           NUMBER OF VALUE BLOCKS TO READ:   1
                           VALUE BLOCK: FAIL-VAL
                               LOCATED: C:\SIPRA\PLT1-EX.EDT

                           WRITE FILE: C:\SIPRA\PLT1-EX.SCO

                           OUTPUT FILE: C:\SIPRA\PLT1-EX.SMO

                           BLOCK FILE: C:\SIPRA\PLT1-EX.BLK

                           VALUE BLOCK FILE: C:\SIPRA\WINDRUN.VAL

                           PROGRAM: DEF-PROG

                           USER BLOCK: DEF-BLOCK

                           DATA PROC OPTIONS:
                             BEFORE ANALYSIS:
                             AFTER ANALYSIS:
                             WRTBLKEQNDNF
                             WRITE FILE: C:\SIPRA\PLT1-EX.SCO
                             COMPUTEBLOCK
                             COMPUTE OPTIONS: PROB /10E-10/

                           FAULT TREE OPTIONS:
                             FORM3
                             NEW TREE: DEF-NEWNAME
                             STEM TREE: DEF-STEM
                             IST TREE: DEF-ISTS
                             IST PREFIX: DEF-PREF
                             TRUNCATION
                             TRUNC VALUE: 1E-9
```

Figure A-10: Output file of SABLE, "PLT1-EX.SMO", Page 1 of 12

```
SABLE(11/12/93)  08/03/95 10:02:04.49  PAGE NO.    2
                             TOP DOWN
                             BOTTOM UP

                           END OF WINDOWS RUN INFORMATION $
                  DLTBLK (PLT1-EX).
                  RDFT (PLT1-EX).
                  DLTVALBLK (FAIL-VAL).
                  RDVALBLK (FAIL-VAL).
                  DLTBLK (DEF-BLOCK,
                          S-DEF-STEM,
                          I-DEF-ISTS,
                          DEF-STEM,
                          DEF-ISTS).
                  DLTEQN.
                  FRMNEWFT (FORM3$ (DEF-PREF)
                                   PLT1-EX/ DEF-STEM, DEF-ISTS).
                  DLTEQN.
                  GENFTEQN (METHOD1$ DEF-ISTS/ I-DEF-ISTS* SAVE$
                            PROBABILITY/1E-9, FAIL-VAL).
                  DLTEQN.
                  GENFTEQN (METHOD1$ DEF-STEM/ S-DEF-STEM*
                            PROBABILITY/1E-9, FAIL-VAL).
                  LDBLK (S-DEF-STEM).
                  COMPUTEBLOCK(PROBABILITY/10E-10,FAIL-VAL).
                  WRTBLKEQNDNF.
```

Figure A-10: Output file of SABLE, "PLT1-EX.SMO", Page 2 of 12

```
SABLE(11/12/93)  08/03/95 10:02:04.49  PAGE NO.    3

      EXECUTE SABLE USER PROGRAM DEF-PROG



       DLTBLK (PLT1-EX).

            STATEMENT EXECUTION REQUIRED     0.441 SECONDS FOR DLTBLK



       RDFT (PLT1-EX).

                   FAULT TREE$PLT1-EX.
                   OG$ D-FLOW.
                   IN$D,NO-SIGNAL,D-SUPPLY.
                   OG$ X-SUPPLY.
                   IN$PUMP-FTS,PUMP-FTR,NO-SIGNAL.
                   OUT$A-SUPPLY.
                   UE$ NO-SIGNAL.
                   OUT$D-FLOW,X-SUPPLY,B-FLOW,C-FLOW,A-FLOW,DIVERSION.
                   OG$ A-FLOW.
                   IN$NO-SIGNAL,A-SUPPLY,A.
                   OUT$B-SUPPLY,C-SUPPLY.
                   BE$ D.
                   OUT$D-FLOW.
                   AG$ D-SUPPLY.
                   IN$B-FLOW,C-FLOW.
                   OUT$D-FLOW.
                   OG$ B-FLOW.
                   IN$B,NO-SIGNAL,B-SUPPLY.
                   OUT$D-SUPPLY.
                   BE$ B.
                   OUT$B-FLOW.
                   OG$ C-FLOW.
                   IN$C,NO-SIGNAL,C-SUPPLY.
                   OUT$D-SUPPLY.
                   BE$ C.
                   OUT$C-FLOW.
                   BE$ PUMP-FTS.
                   OUT$X-SUPPLY.
                   OG$ A-SUPPLY.
                   IN$X-SUPPLY,DIVERSION.
                   OUT$A-FLOW.
                   OG$ B-SUPPLY.
                   IN$A-FLOW.
                   OUT$B-FLOW.
                   OG$ DIVERSION.
                   IN$NO-SIGNAL,E.
                   OUT$A-SUPPLY.
                   OG$ C-SUPPLY.
                   IN$A-FLOW.
                   OUT$C-FLOW.
                   BE$ E.
                   OUT$DIVERSION.
                   BE$ PUMP-FTR.
```

Figure A-10: Output file of SABLE, "PLT1-EX.SMO", Page 3 of 12

```
SABLE(11/12/93)  08/03/95 10:02:04.49  PAGE NO.    4
                  OUT$X-SUPPLY.
                  BE$ A.
                  OUT$A-FLOW.

                  THE FOLLOWING EVENTS ARE EQUIVALENT

                       1.  B-SUPPLY
                       2.  C-SUPPLY

                  THERE ARE NO CYCLES IN PLT1-EX

            FAULT TREE BLOCK PLT1-EX
            HAS BEEN ADDED TO THE BLOCK FILE

            STATEMENT EXECUTION REQUIRED     1.320 SECONDS FOR RDFT



       DLTVALBLK (FAIL-VAL).

            STATEMENT EXECUTION REQUIRED     0.219 SECONDS FOR DLTVALBLK



       RDVALBLK (FAIL-VAL).

                  VALUE BLOCK$ FAIL-VAL.
                  CREATE VALUE BLOCK FAIL-VAL
                  1.0E-3$ A$
                  1.0E-5$ B$
                  1.0E-2$ C$
                  1.0E-2$ D$
                  1.0E-2$ E$
                  1.0E-3$ PUMP-FTS$
                  1.0E-3$ PUMP-FTR$
                  1.0E-3$ NO-SIGNAL$

            STATEMENT EXECUTION REQUIRED     0.172 SECONDS FOR RDVALBLK



       DLTBLK (DEF-BLOCK, S-DEF-STEM, I-DEF-ISTS, DEF-STEM, DEF-ISTS).

            STATEMENT EXECUTION REQUIRED     0.270 SECONDS FOR DLTBLK



       DLTEQN.

            STATEMENT EXECUTION REQUIRED     0.000 SECONDS FOR DLTEQN



       FRMNEWFT (FORM3$(DEF-PREF) PLT1-EX/ DEF-STEM, DEF-ISTS).
```

Figure A-10: Output file of SABLE, "PLT1-EX.SMO", Page 4 of 12

```
SABLE(11/12/93)  08/03/95 10:02:04.49  PAGE NO.    5

                  FAULT TREES MERGED

                       1.  PLT1-EX

            ITERATION  1

                  SINGLE-INPUT EVENTS REMOVED

                       1.  B-SUPPLY
                       2.  C-SUPPLY

                  EVENTS REMOVED BY COALESCING

                       1.  X-SUPPLY
                       2.  A-SUPPLY
                       3.  DIVERSION

                  NO EVENTS WERE COMBINED

            ITERATIONS COMPLETED

                  TOP EVENTS AND THEIR INPUTS
                  FOR CREATED LSIP SUBTREES

                        DEF-PREF1
                          PUMP-FTS
                          PUMP-FTR
                          A
                          E

                  THERE WERE    4 EVENTS REMOVED
                  TO FORM THE STEM FAULT TREE DEF-STEM

                  THERE ARE NO CYCLES IN DEF-STEM

                  TOP EVENTS OF DEF-STEM

                       1.  D-FLOW

            FAULT TREE BLOCK DEF-STEM
            HAS BEEN ADDED TO THE BLOCK FILE

                  THERE WERE    9 EVENTS REMOVED
                  TO FORM THE COLLECTION OF LSIP
                  SUBTREES FAULT TREE DEF-ISTS

                  THERE ARE NO CYCLES IN DEF-ISTS

                  TOP EVENTS OF DEF-ISTS

                       1.  DEF-PREF1
```

Figure A-10: Output file of SABLE, “PLT1-EX.SMO”, Page 5 of 12

```
SABLE(11/12/93)  08/03/95 10:02:04.49  PAGE NO.    6

          FAULT TREE BLOCK DEF-ISTS
          HAS BEEN ADDED TO THE BLOCK FILE

          STATEMENT EXECUTION REQUIRED     0.660 SECONDS FOR FRMNEWFT


     DLTEQN.

          STATEMENT EXECUTION REQUIRED     0.000 SECONDS FOR DLTEQN


     GENFTEQN (METHOD1$ DEF-ISTS/ I-DEF-ISTS* SAVE$ PROBABILITY/ 1E-9,
               FAIL-VAL).

                TOP EVENTS OF DEF-ISTS

                     1.  DEF-PREF1

                SABLE USER PROGRAM SEGMENT PRODUCED BY GENFTEQN

                       DLTEQN.
                       LDBLK (DEF-ISTS).
                       SUBINEQN (DEF-PREF1, DEF-PREF1).
                       TRNTRMVAL (PROBABILITY/ 1E-9, FAIL-VAL* DEF-PREF1,
                                  DEF-PREF1).
                       FRMBLK (I-DEF-ISTS* ONLY$ DEF-PREF1).
                       LDBLK (I-DEF-ISTS).
                       COMTRMVAL (PROBABILITY/ 1E-9, FAIL-VAL* DEF-PREF1).
                       SVISTVL (FAIL-VAL).

          STATEMENT EXECUTION REQUIRED     0.219 SECONDS FOR GENFTEQN


     DLTEQN.

          STATEMENT EXECUTION REQUIRED     0.000 SECONDS FOR DLTEQN


     LDBLK (DEF-ISTS).

          STATEMENT EXECUTION REQUIRED     0.059 SECONDS FOR LDBLK


     SUBINEQN (DEF-PREF1, DEF-PREF1).

          STATEMENT EXECUTION REQUIRED     0.113 SECONDS FOR SUBINEQN


     TRNTRMVAL (PROBABILITY/ 1E-9, FAIL-VAL* DEF-PREF1, DEF-PREF1).
```

Figure A-10: Output file of SABLE, “PLT1-EX.SMO”, Page 6 of 12

```
SABLE(11/12/93)  08/03/95 10:02:04.49  PAGE NO.    7
                        THE MAXIMUM NUMBER OF TERMS THAT CAN BE
                        GENERATED BY EXPANSION IS          4.

                        TERMS GENERATED BY EXPANSION
                              4 TERMS CONTAIN  1 VARIABLES
                        TOTAL TERMS GENERATED        4.

                        THE MAXIMUM TERM VALUE FOR COMPUTATION 1 IS   9.999999780000E-03
                             (THE SUM OF THE TERM VALUES IS   1.300000030000E-02)

                        *** WARNING ***  THE SUM OF TERM VALUES WAS DERIVED BEFORE
SIMPLIFICATION.
                                         IF TERM(S) ARE REMOVED IN THE SIMPLIFICATION STEP
BELOW, THEN THIS CUT SET
                                         EXPRESSION SHOULD BE REQUANTIFIED TO OBTAIN THE
CORRECT SUM OF TERM VALUES.

                  EXPANSION TOOK     0.000 SECONDS.

                        TERMS RETAINED BY SIMPLIFICATION
                              4 TERMS CONTAIN  1 VARIABLES
                        TOTAL TERMS RETAINED        4.
                  SIMPLIFICATION TOOK     0.000 SECONDS.

                  FACTORIZATION TOOK     0.000 SECONDS.

            STATEMENT EXECUTION REQUIRED     0.098 SECONDS FOR TRNTRMVAL


       FRMBLK (I-DEF-ISTS* ONLY$ DEF-PREF1).

            EQUATION BLOCK I-DEF-ISTS
            HAS BEEN ADDED TO THE BLOCK FILE

            STATEMENT EXECUTION REQUIRED     0.281 SECONDS FOR FRMBLK


       LDBLK (I-DEF-ISTS).

            STATEMENT EXECUTION REQUIRED     0.109 SECONDS FOR LDBLK


       COMTRMVAL (PROBABILITY/ 1E-9, FAIL-VAL* DEF-PREF1).

                        THE MAXIMUM NUMBER OF TERMS THAT CAN BE
                        GENERATED BY EXPANSION IS          4.

                        TERMS GENERATED BY EXPANSION
                              4 TERMS CONTAIN  1 VARIABLES
                        TOTAL TERMS GENERATED        4.
                  EXPANSION TOOK     0.109 SECONDS.
```

Figure A-10: Output file of SABLE, "PLT1-EX.SMO", Page 7 of 12

```
SABLE(11/12/93)  08/03/95 10:02:04.49  PAGE NO.    8
                                                        * * * * VARIABLE OCCURRENCE TABLE * * * *


                              NONCOMPLEMENT        NUMBER OF          COMPLEMENT
NUMBER OF
                                VARIABLE          OCCURRENCES          VARIABLE
OCCURRENCES


                                PUMP-FTS                1
                                PUMP-FTR                1
                                A                       1
                                E                       1

                          THERE ARE   4 DIFFERENT VARIABLES IN THE
                          EQUATION FOR DEF-PREF1



                  TERM     PROB.
                 NUMBER   OF TERM

                                    DEF-PREF1 =

                     1    1.0000E-02    E +

                     2    1.0000E-03    A +

                     3    1.0000E-03    PUMP-FTR +

                     4    1.0000E-03    PUMP-FTS

                          THE MAXIMUM TERM VALUE FOR COMPUTATION 1 IS   9.999999780000E-03
                               (THE SUM OF THE TERM VALUES IS   1.300000030000E-02)

            STATEMENT EXECUTION REQUIRED     0.160 SECONDS FOR COMOTRMVAL


       SVISTVL  (FAIL-VAL).

                 SAVE IST VALUES TO VALUE BLOCK FAIL-VAL

            STATEMENT EXECUTION REQUIRED     0.281 SECONDS FOR SVISTVL


       DLTEQN.

            STATEMENT EXECUTION REQUIRED     0.000 SECONDS FOR DLTEQN


       GENFTEQN (METHOD1$ DEF-STEM/ S-DEF-STEM* PROBABILITY/ 1E-9, FAIL-VAL).
```

Figure A-10: Output file of SABLE, "PLT1-EX.SMO", Page 8 of 12

```
SABLE(11/12/93)  08/03/95 10:02:04.49  PAGE NO.    9
                  TOP EVENTS OF DEF-STEM

                       1.  D-FLOW

                  SABLE USER PROGRAM SEGMENT PRODUCED BY GENFTEQN

                         DLTEQN.
                         LDBLK (DEF-STEM).
                         SUBINEQN (D-FLOW, D-FLOW* STOP$ A-FLOW).
                         TRNTRMVAL (PROBABILITY/ 1E-9, FAIL-VAL* D-FLOW,
                                    D-FLOW* EXCEPTNONCMP$ A-FLOW).
                         SUBINEQN (D-FLOW, D-FLOW).
                         TRNTRMVAL (PROBABILITY/ 1E-9, FAIL-VAL* D-FLOW,
                                    D-FLOW).
                         FRMBLK (S-DEF-STEM* ONLY$ D-FLOW).

            STATEMENT EXECUTION REQUIRED     0.160 SECONDS FOR GENFTEQN



       DLTEQN.

            STATEMENT EXECUTION REQUIRED     0.000 SECONDS FOR DLTEQN



       LDBLK (DEF-STEM).

            STATEMENT EXECUTION REQUIRED     0.168 SECONDS FOR LDBLK



       SUBINEQN (D-FLOW, D-FLOW* STOP$ A-FLOW).

            STATEMENT EXECUTION REQUIRED     0.051 SECONDS FOR SUBINEQN



       TRNTRMVAL (PROBABILITY/ 1E-9, FAIL-VAL* D-FLOW, D-FLOW* EXCEPTNONCMP$
                  A-FLOW).

                        THE MAXIMUM NUMBER OF TERMS THAT CAN BE
                        GENERATED BY EXPANSION IS         11.

                        TERMS GENERATED BY EXPANSION
                              4 TERMS CONTAIN  1 VARIABLES
                              7 TERMS CONTAIN  2 VARIABLES
                        TOTAL TERMS GENERATED       11.

                        THE MAXIMUM TERM VALUE FOR COMPUTATION 1 IS    1.000000000000E+00
                             (THE SUM OF THE TERM VALUES IS    1.024020200000E+00)

                        *** WARNING ***  THE SUM OF TERM VALUES WAS DERIVED BEFORE
SIMPLIFICATION.
                                         IF TERM(S) ARE REMOVED IN THE SIMPLIFICATION STEP
BELOW, THEN THIS CUT SET
                                         EXPRESSION SHOULD BE REQUANTIFIED TO OBTAIN THE
CORRECT SUM OF TERM VALUES.
```

Figure A-10: Output file of SABLE, "PLT1-EX.SMO", Page 9 of 12

```
SABLE(11/12/93)  08/03/95 10:02:04.49  PAGE NO.   10

                  EXPANSION TOOK     0.168 SECONDS.

                        TERMS RETAINED BY SIMPLIFICATION
                              3 TERMS CONTAIN  1 VARIABLES
                              1 TERMS CONTAIN  2 VARIABLES
                        TOTAL TERMS RETAINED        4.
                  SIMPLIFICATION TOOK     0.051 SECONDS.

                  FACTORIZATION TOOK     0.000 SECONDS.

            STATEMENT EXECUTION REQUIRED     0.391 SECONDS FOR TRNTRMVAL



       SUBINEQN (D-FLOW, D-FLOW).

            STATEMENT EXECUTION REQUIRED     0.000 SECONDS FOR SUBINEQN



       TRNTRMVAL (PROBABILITY/ 1E-9, FAIL-VAL* D-FLOW, D-FLOW).

                        THE MAXIMUM NUMBER OF TERMS THAT CAN BE
                        GENERATED BY EXPANSION IS          5.

                        TERMS GENERATED BY EXPANSION
                              4 TERMS CONTAIN  1 VARIABLES
                              1 TERMS CONTAIN  2 VARIABLES
                        TOTAL TERMS GENERATED        5.

                        THE MAXIMUM TERM VALUE FOR COMPUTATION 1 IS   1.300000030000E-02
                             (THE SUM OF THE TERM VALUES IS   2.500010100000E-02)

                        *** WARNING ***  THE SUM OF TERM VALUES WAS DERIVED BEFORE
SIMPLIFICATION.
                                         IF TERM(S) ARE REMOVED IN THE SIMPLIFICATION STEP
BELOW, THEN THIS CUT SET
                                         EXPRESSION SHOULD BE REQUANTIFIED TO OBTAIN THE
CORRECT SUM OF TERM VALUES.

                  EXPANSION TOOK     0.109 SECONDS.

                        TERMS RETAINED BY SIMPLIFICATION
                              3 TERMS CONTAIN  1 VARIABLES
                              1 TERMS CONTAIN  2 VARIABLES
                        TOTAL TERMS RETAINED        4.
                  SIMPLIFICATION TOOK     0.000 SECONDS.

                  FACTORIZATION TOOK     0.000 SECONDS.

            STATEMENT EXECUTION REQUIRED     0.160 SECONDS FOR TRNTRMVAL



       FRMBLK (S-DEF-STEM* ONLY$ D-FLOW).

            EQUATION BLOCK S-DEF-STEM
            HAS BEEN ADDED TO THE BLOCK FILE
```

Figure A-10: Output file of SABLE, “PLT1-EX.SMO”, Page 10 of 12

```
SABLE(11/12/93)  08/03/95 10:02:04.49  PAGE NO.   11

            STATEMENT EXECUTION REQUIRED     0.219 SECONDS FOR FRMBLK


       LDBLK (S-DEF-STEM).

            STATEMENT EXECUTION REQUIRED     0.109 SECONDS FOR LDBLK


       COMPUTEBLOCK (PROBABILITY/ 10E-10, FAIL-VAL).

                  SABLE USER PROGRAM SEGMENT PRODUCED BY COMPUTEBLOCK

                         COMTRMVAL (PROBABILITY/ 10E-10, FAIL-VAL* D-FLOW).

            STATEMENT EXECUTION REQUIRED     0.000 SECONDS FOR COMPUTEBLOCK


       COMTRMVAL (PROBABILITY/ 10E-10, FAIL-VAL* D-FLOW).

                        THE MAXIMUM NUMBER OF TERMS THAT CAN BE
                        GENERATED BY EXPANSION IS          4.

                        TERMS GENERATED BY EXPANSION
                              3 TERMS CONTAIN  1 VARIABLES
                              1 TERMS CONTAIN  2 VARIABLES
                        TOTAL TERMS GENERATED        4.
                  EXPANSION TOOK     0.109 SECONDS.


                                   * * * * VARIABLE OCCURRENCE TABLE * * * *


                             NONCOMPLEMENT      NUMBER OF         COMPLEMENT
NUMBER OF
                               VARIABLE        OCCURRENCES         VARIABLE
OCCURRENCES


                               D                     1
                               NO-SIGNAL             1
                               B                     1
                               C                     1
                               DEF-PREF1             1

                        THERE ARE   5 DIFFERENT VARIABLES IN THE
                        EQUATION FOR D-FLOW
```

Figure A-10: Output file of SABLE, "PLT1-EX.SMO", Page 11 of 12

```
SABLE(11/12/93)  08/03/95 10:02:04.49  PAGE NO.   12

                  TERM     PROB.
                 NUMBER   OF TERM

                                    D-FLOW =

                     1   1.3000E-02    DEF-PREF1 +

                     2   1.0000E-02    D +

                     3   1.0000E-03    NO-SIGNAL +

                     4   1.0000E-07    B * C

                      THE MAXIMUM TERM VALUE FOR COMPUTATION 1 IS   1.300000030000E-02
                           (THE SUM OF THE TERM VALUES IS   2.400010080000E-02)

            STATEMENT EXECUTION REQUIRED     0.172 SECONDS FOR COMTRMVAL


       WRTBLKEQNDNF.

                      THE MAXIMUM NUMBER OF TERMS THAT CAN BE
                      GENERATED BY EXPANSION IS          4.

                      TERMS GENERATED BY EXPANSION
                            3 TERMS CONTAIN  1 VARIABLES
                            1 TERMS CONTAIN  2 VARIABLES
                      TOTAL TERMS GENERATED        4.
                EXPANSION TOOK     0.059 SECONDS.


                THERE IS NO EQUATION FOR DEF-PREF1

            STATEMENT EXECUTION REQUIRED     0.160 SECONDS FOR WRTBLKEQNDNF

END EXECUTION OF SABLE.
```

Figure A-10: Output file of SABLE, "PLT1-EX.SMO", Page 12 of 12

Figure A-11 contains the output of the file "PLT1-EX.SCO", which contains the unquantified cut sets.

```
D-FLOW =
D +
NO-SIGNAL +
DEF-PREF1 +
B * C .
```

Figure A-11: Unquantified Cut Set File, "PLT1-EX.SCO".

Appendix B: SABLE 2.0 Problem Reporting Form

Name: ______________________________ Date: ____________

Organization: ______________________________ Phone: ____________

Address: ______________________________

____

____

Code Version: ______________________________

(Found under "Help | About SABLE . . .")

Problem Information

Type of Problem (check appropriate items):

__ Unclear Documentation
__ Inadequate User Protection
__ Obvious Coding Error
__ Missing Feature
__ Suggested Improvement
__ Other/Unknown

Problem Description: ______________________________

____

____

____

____

____

____

____

(Use additional pages if necessary. Also, when possible, attach copies of results or screen dumps that demonstrate the error or send a disk containing the program file and all input fault tree and value block files with the error).

Severity of problem (check appropriate items):

__ Inconvenience
__ Program Died
__ Lost Data and Results

Urgency of Fix (check appropriate items):

__ Can Work Around It for Now
__ Can't Use Program Until it's Fixed
__ Need Immediate Fix and Willing to Supply Funding

Computer System Information

CPU:	__ 80386
	__ 80486
	__ Pentium
	__ P6
	Other: ____________________ (Please specify)
Hardware Manufacturer:	______________________________
Memory:	Conventional: ______________ kB
	Expanded: ______________ MB
	Extended: ______________ MB
Printer:	__ HP Series II
	__ HP Series III
	__ HP Series IV
	__ Other: ______________ (Please specify)
Other Peripherals:	______________________________
	______________________________

For Department Use Only

Date Logged in: ______________________________

Priority Assigned: ______________________________

Receipt Acknowledged to Sender: ______________________________

Date Resolved: ______________________________

Resolution: ______________________________

____

____

____

____

____

Resolution Acknowledged by Sender: ______________________________

Appendix C: Index

Distribution

Mike Baird
PO Box 4946
Glen Allen, VA 23058-4956

Joseph M. Butner
PO Box 1663
TAS-11, MS K557
Los Alamos National Laboratory
Los Alamos, NM 87545

Mary Hall
PO Box 1663
TAS-11, MS K557
Los Alamos National Laboratory
Los Alamos, NM 87545

Eric Haskin
Chemical and Nuclear Engineering
FEC, Rm. 217
University of New Mexico
Albuquerque, NM 87131

Sharif Heger
Chemical and Nuclear Engineering
FEC, Rm. 217
University of New Mexico
Albuquerque, NM 87131

Desmond Stack
PO Box 1663
TAS-11, MS K557
Los Alamos National Laboratory
Los Alamos, NM 87545

Dale Talbott
PO Box 1663
TAS-11, MS K557
Los Alamos National Laboratory
Los Alamos, NM 87545

Copies	Org.	Name	MS
1	5415	Arthur Payne	0425
1	5417	Dave Kunsman	0423
1	6412	Allen Camp	0747
1	6412	Vince Dandini	0747
1	6412	Sharon Daniel	0747
1	6412	Susan Dingman	0747
1	6412	John Forester	0747
20	6412	Kelly Hays	0747
1	6412	Don Michell	0747
1	6412	Steve Nowlan	0747
1	6412	Heather Shriner	0747
1	6412	Bevan Staple	0747
1	6412	Tina Tanaka	0747
1	6412	Ellen Walroth	0747
1	6412	Donnie Whitehead	0747
1	6412	Greg Wyss	0747
1	6413	Tom Brown	0748
1	6413	Julie Gregory	0748
1	6613	Chris Atcitty	0746
1	6613	Maria Armendariz	0746
1	6613	Jim Campbell	0746
1	6613	Jim Martinez	0746
1	6613	Laura Painton	0746
1	6613	Dave Robinson	0746
1	6613	Hugh Whitehurst	0746
1	7316	Gwen Pirtle	1036
1	12333	Dave Carlson	0405
1	12333	Paul Demmie	0405
1	12333	Nancy Dhooge	0405
1	12333	Michael Dvorack	0492
1	12333	Martin Fuentes	0405
1	12333	Todd Jones	0405
5	12333	Kevin Maloney	0405
1	12333	Keri Sobolik	0405
1	12333	Robert Winchester	0491
1	8523-2	Central Technical Files	9018
5	4414	Technical Library	0899
2	7613-2	Document Processing	0100
1	12615	Print Media	0619