ARH-ST-112

UC-32

UC-41

UC-78

ALLCUTS, A FAST,
COMPREHENSIVE FAULT TREE ANALYSIS CODE

by

W. J. Van Slyke
D. E. Griffing

Advanced Waste Engineering Department
Hanford Waste Engineering

July, 1975

NOTICE
This report was prepared as an account of work sponsored by the United States Government. Neither the United States nor the United States Energy Research and Development Administration, nor any of their employees, nor any of their contractors, subcontractors, or their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness or usefulness of any information, apparatus, product or process disclosed, or represents that its use would not infringe privately owned rights.

ATLANTIC RICHFIELD HANFORD COMPANY
RICHLAND, WASHINGTON 99352

MASTER

FIRST UNRESTRICTED
DISTRIBUTION MADE SEP 2 - 1975

DISTRIBUTION OF THIS DOCUMENT UNLIMITED

DISCLAIMER

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency Thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

DISCLAIMER

Portions of this document may be illegible in electronic image products. Images are produced from the best available original document.

TABLE OF CONTENTS

	<u>Page</u>
ABSTRACT	iii
INTRODUCTION	1
SUMMARY	4
ALLCUTS, FAULT TREE ANALYSIS PROGRAM	5
THE ALGORITHM OF ALLCUTS	5
USER INSTRUCTIONS	8
BRANCH: TABLE OF GATE AND BOTTOM EVENT INTERRELATIONSHIPS .	19
OBJECTIVE OF BRANCH	19
THE ALGORITHM OF BRANCH	19
OUTPUT FROM BRANCH	22
ACKNOWLEDGMENTS	23
REFERENCES	25
BIBLIOGRAPHY	27
APPENDICES	
A FAULT TREE EXAMPLE	29
B ALLCUTS DATA INPUT EXAMPLE	33
C OUTPUT DATA LISTING FROM ALLCUTS FOR THE EXAMPLE DATA .	39
D OUTPUT DATA LISTING FROM BRANCH FOR THE EXAMPLE DATA .	51
E SOURCE LANGUAGE LISTING FOR PROGRAM ALLCUTS	55
F SOURCE LANGUAGE LISTING FOR PROGRAM BRANCH	109
G METHOD FOR PROBABILITY EVALUATION OF CUTSETS	119

ABSTRACT

A description, user instructions, and a source program listing are presented for ALLCUTS, a Fortran computer code for fault tree analysis. The code has been specifically designed to be easy to use as well as being fast, versatile, and powerful. It may be easily modified by a moderately skilled programmer to fit the variable needs of the user and the capabilities of his computer. A code, BRANCH, for determining input data gate and bottom event interrelationships is also presented.

ALLCUTS, A FAST
COMPREHENSIVE FAULT TREE ANALYSIS CODE

INTRODUCTION

The fortran computer code, ALLCUTS, which is discussed and presented in this report, was developed to fill the need for a faster and more complete identification of fault tree cutsets. It was conceived after trial solutions to the author's fault tree with existing Monte Carlo and Deterministic codes either produced considerably less than complete lists of cutsets for the first type of code or implied extremely large computer execution times with the second type of code.

The author's fault tree proved challenging to these prior versions of the code. First, the most probable cutsets resided in the groups of cutsets which contained three and four bottom events. The problem was further complicated by the need to find all the cutsets with probabilities ranging to several factors of ten lower than the most probable ones. This need arose because the cutset seriousness was evaluated as the logarithm of the product of the cutset probability, multiplied by the quantity and type of material released and transported. This definition of cutset seriousness was defined in an internally distributed report as the "risk index," in contrast to the usual definition of cutset seriousness, the system failure probability for the cutset. Thus the most serious cutsets proved to be considerably less probable than

the most probable cutsets depending on the amount and type of material involved.

The basic algorithm for ALLCUTS was first conceived and tested in late 1973. It proved to be fast and effective, and subsequent trial and error development minimized its core storage requirements, further decreased its execution times, and added to its versatility and convenience of use. The more important characteristics of the code may be listed as follows:

- Is easy to use.
- Provides comprehensive error checking of the input data file.
- Finds non-minimal or minimal cutsets.
- Finds cutsets within any size range from one to ten events.*
- Finds cutsets within any specified probability range.
- Internally stores, orders by probability, and lists up to a thousand cutsets.*
- Optionally prints all cutsets found, without storing them internally.
- Probability calculations (and therefore input of bottom event probability data) are optional.
- Probability calculations employ the methodology employed in the Rasmussen Report, WASH-1400.⁽²⁾
- Cutsets listings are printed with bottom event descriptions at the option of the user.
- The input data format is consistent with KILMER computer graphics code,⁽³⁾ and with the BRANCH code for generating a

*This characteristic may be modified easily by a moderately skilled programmer.

- table of fault tree interrelationships, presented in Appendix F.
- An optional table of bottom event importances and importance factors may be printed.
- Looping detection and diagnosis is powerful and automatic.
- Recapitulation of input data is optional.
- Analysis may start at the top event of the tree or at any intermediate gate.
- An optional trace printout may be ordered for all paths found from the top of the tree to any gate.
- All cutsets containing any one bottom event may be listed.
- Cumulative cutset probabilities may be requested by cutset size group and/or for the system as whole.
- Performs optional output of cutset data to a data file for further processing by other codes, as when evaluating cutset probability spectrums. (7)

SUMMARY

A fault tree analysis code and a code to make a table of fault tree gate and bottom event interrelationships were developed to enable a particular fault tree analysis task to be performed. They are designed to be easy to use, flexible, and powerful.

ALLCUTS employs an efficient top down set expansion algorithm to find cutsets. The code is presently capable of finding up to 10-event cutsets from a tree with 175 bottom events and 425 gates. Its algorithm is general, however, and it may be modified relatively easily by the user to handle larger or smaller fault trees, depending on the needs of the user and the capabilities of his computer. ALLCUTS as listed requires 34,700 decimal words of internal memory storage on the Computer Sciences Corporation Richland Cyber 70 computer. Reduction of internal sets storage from 1000 to 100 permits it to be run interactively with less than 24,000 words of memory required.

Program BRANCH was developed to meet the need to identify fault tree transfers for fault tree diagram and input data error checking. Although this function can be, and was performed manually prior to the development of BRANCH, the automatic generation of fault tree interrelationships saves time and money, and improves the efficiency of the fault tree analyst. Two tables are produced. The first details gate to gate relationships and the second identifies the gates to which each bottom event is an input.

ALLCUTS: FAULT TREE ANALYSIS PROGRAM

ABOUT THE ALGORITHM

The algorithm for ALLCUTS was independently conceived in late 1973 after the realization that the then existing and available codes were not sufficiently powerful for the task. Subsequently, reports were found which presented codes using similar algorithm.^(3,4)

Continued use of the algorithm occurred in a succession of computer codes which explored several approaches which either required excessive computer memory size, prohibitive input-output times for external memory access, or a less than useful output format. The performance of the present code avoids these former evolutionary difficulties while enabling a fast, powerful, and versatile approach to a fault tree analysis.

Two types of results are required when performing a fault tree analysis. The first is to identify all the cutsets of interest, and the second is to calculate the likelihood of the cutset occurrence or existence. Furthermore, knowing the cutsets is often useful even when probability calculations are not possible. ALLCUTS first finds each cutset and then, if required calculates the cutset probability.

Conception of the basic algorithm of ALLCUTS occurred as a result of realizing that,

- (1) the form of the answer required was that of a series of

sets of AND-related events, where the sets themselves are mutually OR-related.

- (2) the top event of the fault tree diagram may be considered to be a special case of this type of answer, that is, the top event may be looked upon as a cutset which encompasses all the permutations of the entire fault tree system, and
- (3) it is possible to completely expand the fault tree top event into its individual cutsets without ever deviating from the desired form of the answer.

The expansion is performed by following a relatively small number of rules, as follows. The starting cutset is the starting event for the analysis, usually the top event of the tree. Where any event in a developing set is a bottom event, no further development is possible and the event label is carried with the set until the set is fully expanded. The set is fully developed when it contains only bottom events. Where any event in a developing set is an AND gate, and AND gate label is replaced in the set by the gates and/or bottom events which are input to the AND gate. Similarly, an INHIBIT gate, which by definition can only have a single input event, is replaced in the developing set by its input event label. The convention used here is that INHIBIT gates are not considered to have two inputs and the inhibit condition does therefore not appear as a member of a cutset. Finally, an OR gate with N inputs in a developing set is expanded by creating N cutsets, each having the OR gate of the original set replaced by one of its N inputs. By convention, the original set has the OR gate replaced by its first input. During and after each expansion step, the developing cutsets have any duplicate events deleted and have their set probability value,

initially 1.0, multiplied by the gate event probability, if other than 1.0.

In order to conserve on computer memory space requirements, the last set in a list of developing cutsets is always the one chosen for further expansion. When some "last set" in the list finally becomes either too large for further consideration or becomes fully expanded, containing nothing but bottom events, then it receives whatever further processing is necessary and is deleted from the list of developing cutsets. The number of developing sets requiring internal storage has been considerably less than a hundred for all applications of ALLCUTS to date.

In order to speed the execution of ALLCUTS, the inputs to all gates are initially converted from Alphanumeric labels to non-redundant gate and bottom event integer indices. This permits the rapid, direct retrieval of gate or bottom event information by referring to its index. The bottom event indices used in the expansion are their own original indices plus the maximum observed gate index value. After each cutset is fully expanded, the maximum observed gate index value is subtracted from the bottom indices in the set to obtain the corrected bottom event indices for set probability calculations, and for the corresponding bottom event labels and descriptions. The net result of performing the algorithm with indices rather than with alphanumeric labels is to eliminate label searches and to decrease execution times considerably.

The method of probability calculations used in ALLCUTS is that used in the Rasmussen report⁽¹⁾ as elaborated to cover the range of failure and unavailability possibilities.⁽⁷⁾ Maximum likelihood set duration times are calculated by the formulae of Roberts.⁽⁵⁾

Additional information about the inner workings of the code may be obtained from the comment statements in the Fortran source listing, Appendix E.

USER INSTRUCTIONS

The following text describes the input data expected by the program on a card-by-card and section-by-section basis including some brief information about how the program uses the input data. Appendix B is a short input data file listing which is typical of the input data expected by the program. Appendix A presents the fault tree diagram from which the input data was taken.⁽⁶⁾

The input data is set up on an 80 column format for maximum usefulness by a variety of data input devices including 80 column punch cards.

The first card image read is the case title. All 80 columns may be used. The title is reproduced at several places in the output data. Additional cards may be inserted (after the case title card) before each of the several data section identifier cards which have an asterisk in column 1. These additional cards will be reproduced in the output data in their entirety exactly as read. They are ignored by the

program and must not have an asterisk in column 1.

The first section of the data contains nine integers, two floating point numbers, and two alphanumeric labels.

The first card of this section of the data has the characters *DATA in column one through five. The second card of the data contains the nine integers in 815 format. They control the operation of the program as follows:

MAX The first integer is MAX and specifies the maximum set size that the program will attempt to find. If MAX is specified greater than 10, the program will reduce it to 10. The program may be easily modified to search for larger than ten-event cutsets, however, since the algorithm and program source language is perfectly general. If MAX is input as 0, the program will read through the input data doing error checking as usual but will come to an orderly stop prior to doing the actual analysis.

The program actually will carry one more event in each developing set than is specified by MAX. When the set has been fully expanded, it will be accepted for further processing and printout if its number of events is less than, or equal to MAX. This procedure was adopted because it is possible to have multiple paths to one bottom event, leading to expanding sets which appear to be too large. When the

set is fully expanded and the duplications of events eliminated, the true set size may be acceptable. It would be coincidental if duplicated gate or bottom events appeared simultaneously in a developing set before it is fully expanded and were therefore recognized and eliminated. Carrying an extra event in the developing set therefore helps to prevent losing a set which temporarily appears to be too large to save.

- MIN The second integer, MIN specifies the minimum set size to be saved for processing after expansion. MIN can have values from zero through the value of MAX. If its value lies outside this range, its value will be set to one.
- MSS The third variable is MSS, the maximum number of sets stored for eventual printout. This variable can have values up to 1000 in the listed version of the program, but this limitation is primarily a function of the available computer memory size. Since some running options of the program do not require sets storage, the program may also be run on a relatively small computer.
- IDEX1 The fourth variable is IDEX1, a specification on the extent and type of final analysis to be performed. Its effect on program behavior is as follows:
Less than one is not permitted.

- 1 Sets are printed with no probability calculation results, ordered by set size.
- 2 Sets are printed with probability calculations, ordered by set size and by probability within each set size group.
- 3 As in 2 with cumulative probability and maximum likelihood duration times calculated for each set size group.
- 4 As in 3 with cumulative probability calculated for the entire system as well.
- 5 Sets are printed with probability calculations, ordered by probability only, not by set size.
- 6 As in 4 with cumulative probability calculated and printed for the entire system.
- 7 Sets are printed out in the order found, without their probabilities being calculated or printed.
- 8 Sets are printed out in the order found along with their probabilities and maximum likelihood duration times. They are therefore not ordered by probability.

IDEX1 values greater than 8 are not permitted.

IDEX2 The fifth integer on this card image is IDEX2 and controls whether bottom event descriptions accompany the cutset events listings and also controls the calculation and printout of a table of bottom event importances and importance factors. Table I lists the program behavior for IDEX2 values of one through four. The printout of the table of importances and importance factors also requires an IDEX1 value that will result in cutset probabilities being calculated and printed.

TABLE I

IDEX2 RESULTS

<u>IDEX2 Value</u>	<u>Bottom Event Descriptions</u>	<u>Table of Importances</u>
1	No	No
2	No	Yes
3	Yes	No
4	Yes	Yes

NABS The sixth variable is NABS, a variable which controls absorptions checking. If it is input as one, no absorptions checking will occur. If NABS is input as two, absorptions checking will occur only between sets with the same number of events. If NABS is input as three, absorptions checking will occur between sets with both the same and different numbers of events, resulting in only minimal cutsets being retained and printed.

NDPR The seventh variable is NDPR. It controls the printout of the gate and bottom event data read in the input data. If it is greater than zero, gate and bottom event data will be printed out as read with a few spaces inserted for readability. If NDPR is less than one, this printout will be omitted.

NTRANS The eighth variable is NTRANS. It controls the output of

sets data to a data transfer file, TOUT, logical unit 7. If it is zero, no data transfer occurs. If it is 1, the internally stored sets will be transferred to the data transfer file, if it is set equal to 2, all the sets generated will be written to the file as they are generated. In either case, each line or card image of TOUT will contain the number of events in the set, the set probability, and the event indices in the same order read by ALLCUTS. The third card of this section of data contains two floating point and two optional alphanumeric variables. The input format is 2E10.3, 2(1X, A7).

PMIN The first floating point number PMIN, is a number between zero and one which is the cutoff probability level for sets. If the set probability is less than this number, the set will be discarded with no further processing.

TIMEN The second floating point number, TIMEN, is the mission time in hours used in calculating cutset probabilities.

GTRACE The first alphanumeric input, GTRACE, (optional and may be left blank) beginning in column 22, is interpreted as the name of gate in the tree. Whenever this gate name is encountered in expanding a set, the gate path from the top event of the analysis down to that gate is printed.

CTTRACE The second alphanumeric input, CTRACE, (optional and may be left blank) begins in card column 29. It is the name of a bottom event which, if present in a fully expanded cutset will cause the cutset to be written out immediately. Probability of the cutset is output depending on whether the value of IDEX1 requires probability calculations or not, and irrespective of the input value of PMIN.

The final card of this section has END punched in card columns one through three.

The next section of data (which may be preceded by hollerith information cards as explained above) is signaled by a card with *TREE in columns one through five with the rest of the card blank. The gate name, type, probability, and input type quantities and names and gate hollerith descriptions follow in sets of two cards each.

The gate names of each gate card may be one to seven characters beginning in column one. Column eight is blank. Columns nine through eleven contain the gate type, either AND, OR, or INH. Column twelve is blank.

A gate probability multiplier may appear in columns 13 through 20, read under a F8.5 format specification. If omitted for any type of gate, it is assumed to be 1.0.

Column 2 must be blank. Two integers appear in columns 22 and 23,

signifying the number of gates and bottom events respectively, input to this gate. Columns 24 and 25 are blank.

The names of the gate inputs begin in columns 26, 34, 53, 50, 58, 66, and 74, each having a maximum of seven characters, and agreeing in type and number with the two integers appearing in columns 22 and 23. Two blank columns (columns 24, and 25) precede the first input name and a blank must follow each name. If a character appears in any of these blank spots on this card, an error message will be printed, the gate will be ignored, and reading gate data will continue.

The second card of each gate data set is alphanumeric information, the hollerith description of the data which is printed out as part of the input data iteration and is also used by a plotting program, to place this information in the gate rectangles. It is entered on the second card in four twenty-character blocks and has a space inserted between blocks when reprinted in the output data.

In addition, the program accepts and ignores a second type of gate card set used only by the companion plotting program. Its first gate card is blank except for the characters MOD which appear in columns 9 through 11. A second, alphanumeric card after this second type of gate card is read and printed out precisely as described just above. These cards are printed by the program and are otherwise ignored. In either case the second, alphanumeric card may contain any characters or may be blank.

The format specification for the pairs of gate card images read by the program is,

(A7, A1, A3, A1, F8.5, A1, I2, A1, 7 (A1, A7)/8(A10))

The first of the last two cards of the section must be an END card.

The second card may also be an END card or any other card including a blank one.

The third section of the data, is optional, and may be omitted. It is used to define the starting gate for the analysis to be a particular gate, usually not the top event of the tree which the program has already found. The first card of this section contains the characters, *TOPS, which appears in columns 1 through 5. The second card contains the starting gate name beginning in column 1. This gate name may be any gate in the tree including the top event. The third card of this section is an END card as defined above.

The final section of the data presents data for the bottom events of the tree. The first card of this section has *RATES in card columns 1 through 6. The bottom event data follows in two-card sets. The first card of each set includes the bottom event name, type, probability, detection time, repair time, and unavailability data. The second card has hollerith information as for the gate card data.

The format specification for these two successive card images is

(A7, A1, A3, A1, 12X, A1, 5(F10.6, A1)/8A10)

The bottom event name begins in column 1 of the first card and occupies a maximum of seven characters. The bottom event type designator begins in column nine, either CRC for a circle, DMD for a diamond, or HSE for a house.

The remaining five variables are floating point variables beginning in card columns 26, 37, 48, 59, and 70. The first of these is $\lambda * 10^6$ per hour and the second variable is the detection time, t_d hours.

If the detection time, t_d is zero, then the first variable is taken by the program to be a straight probability. The third variable is the bottom event repair time in hours, and if it is set to zero, the component is assumed to be nonrepairable and its (zero) repair time is not used in the calculation of set effective duration time. The program will print a set effective duration time of 10^{50} if all the bottom events in set are nonrepairable. The set effective duration time is calculated according to

$$T = \frac{1}{\sum_{i=1}^n \frac{1}{t_i}}$$

where $t_i = t_d + t_r$.

The fourth variable maximum allowable downtime, T_{max} , which if greater than zero, is used in the exponential repair factor calculation as presented in the Rasmussen study.⁽¹⁾ The fifth and final variable is the unavailability probability. If it is input as zero when

λ is greater than zero, it will be calculated by the program as follows: For a λ equal to a straight probability ($t_d = 0.$), Q is set equal to λ . For a standby or emergency component (signaled by λ input as 0.), there is no valid way to calculate a Q value and it is set equal to 0. With both λ and t_d greater than zero, Q is calculated as $1.-e^{-(t_d=t_r)}$. The second card of each bottom event data card set carries alphanumeric information as described for the second card of the date data. Contrary to the gate hollerith information, however, this hollerith information is stored in computer memory for printout with the bottom event name if control variable IDEX2 is three or four.

The end of the bottom event data section is signaled to the computer program by an END card and an additional card is required which may carry alphanumeric information or may also be an END card.

The program is set up to read the above data from an attached perm-file with a local file name DT, logical unit 5. It writes to a file with a local file name BOP, logical unit 6.

Set probability and component indices are written to an attached perm-file with a local file name of TOUT, logical unit 7.

BRANCH: TABLE OF GATE AND BOTTOM EVENT INTERRELATIONSHIPS

OBJECTIVE OF BRANCH

Most computer fault trees are written using, as data, gates and the bottom events and gates input to those gates. Frequently, the fault tree analyst needs to know all of the gates that have a particular gate or bottom event as an input. Such a fault tree data cross reference can be used for:

- Checking the validity of the fault tree by determining that each gate or bottom event is input only to the correct gates.
- Debugging the fault tree with regard to loops and unnecessary repetition of gate inputs.
- Providing an easily understandable fault tree diagram by showing the analyst where to place transfer symbols and where to expand the fault tree diagram in a most convenient and clear manner.
- Showing the fault tree analyst the parts of the tree that have the most interdependence because of shared input gates and bottom events.

The computer program BRANCH, written in FORTRAN IV for the CDC (Cyber 70 Computer), provides such a set of cross references to the fault tree data file.

THE ALGORITHM OF BRANCH

The algorithm of BRANCH can be separated into two parts. The first part finds all of the gates that have each gate (excluding the top gate) as an input. The second part of the algorithm finds all of the gates that have each bottom event as an input.

The part of the algorithm for finding all of the outputs from each gate compares each gate name with the names of all of the other gates' input gates. Similarly, the part that finds all of the outputs from each bottom event compares each bottom event name with the names of the bottom events listed as inputs to all of the gates.

The input file for BRANCH (referred to in the program as DATREE and TAPE5) can have the identical format to files used and described previously for fault tree analysis. The file lists each of the gate names as well as the names of all gates and bottom events that are input to each gate. A separate part of the file lists each bottom event name.

In the algorithm each gate name is read into core as the array element $GNAME(N)$, where N is the N th gate read. The gates' inputs to this gate are each read into core as the array element $IGATE(I,N)$, where I and N mean the input is the I th gate input to the N th gate. The bottom events' input to this gate are read into core as $IBEV(I,N)$, which is the I th input bottom event to the N th gate.

Bottom event names are also read into computer core as the elements of the array $BNAME(N)$, or the N th bottom event read.

The total number of gates is NG while the total number of bottom events is NB . The N th gate has IG input gates and IB input bottom events.

The following steps show how the algorithm finds all of the outputs to each and every gate:

- 1 Initialize I, U, and N to 1.
- 2 If GNAME(N) is the same as IGATE(I,U) then do step 3; otherwise go to step 4.
- 3 Store GNAME(J) in the array: GMATCH, then go to step 5.
- 4 If $I < I_G$, then increment I by 1 and go to step 2; otherwise go to step 5.
- 5 Re-initialize I to 1. If $J < N_G$, then increment J by 1 and go to step 2; otherwise go to step 6.
- 6 Output GNAME(N) and the array GMATCH. Re-initialize J to 1. If $N < N_G$, then increment N by 1 and go to step 2; otherwise all of the outputs from each gate have been found and output.

The next six steps show how BRANCH finds the outputs to each and every bottom event. The method is identical to the one for finding the gates' outputs:

- 1 Initialize I, J, and N to 1.
- 2 If BNAME(N) is the same as IBEV(I,J), go to step 3; otherwise go to step 4.
- 3 Store GNAME(J) in the array GMATCH, then go to step 5.
- 4 If $I < I_B$, then increment I by 1 and go to step 2. Otherwise go to step 5.
- 5 Re-initialize I to 1. If $J < N_G$, then increment J by 1 and go to step 2. Otherwise go to step 6.
- 6 Output BNAME(N) and the array GMATCH. Re-initialize J to 1. If $N < N_B$, then increment N by 1 and go to step 2. Otherwise all the gates having BNAME(N) as an input have been found, for all N between 1 and N_B .

A copy of the FORTRAN IV code for the program BRANCH can be found in Appendix F.

OUTPUT FROM BRANCH

The computer program BRANCH writes to a file named REFILE, which is formatted for output on the line printer. The output is arranged in a tabular form of pages with headings on the top of each page. The ratio of length to width for each page is approximately such that it allows the tables generated by the line printer to be photographically reduced and used directly in reports.

There are two kinds of output, one for the gates and one for the bottom events. The tables containing the gate output are arranged into three vertical fields for gate name, inputs to that gate, and gates that have that gate as an input. The bottom events in the second field are preceded by asterisks. The headings and underscores at the top of each field indicate, respectively, the contents and lengths of each field. A sample table of the gate output from BRANCH is shown in Appendix D.

The tables containing the bottom event data are divided into two vertical fields -- one for the bottom event name and one for a list of the names of the gates that have each bottom event as an input. Each page of bottom event data has a heading and underscore at the top indicating the content and length of the tabular fields. Sample bottom event output from BRANCH is shown in Appendix D.

ACKNOWLEDGMENTS

The fault tree example presented here was adapted from a case example in the printed class notes from a fault tree analysis course taught by David F. Haasl at the Joint Center for Graduate Study, Richland, Washington, on February 19 through March 1, 1974. Permission to use this material is gratefully acknowledged.

Appendix G is an edited version of material provided by Dr. T. H. Smith and P. J. Pelto of Battelle Northwest, who graciously provided invaluable assistance and advice about quantification of fault tree analysis cutsets and desired program capabilities during the final stages in the development of ALLCUTS. Their contributions have been greatly appreciated.

(This page has been intentionally left blank)

REFERENCES

- ¹WASH-1400 (Draft), August 1974, USAEC, Washington D. C., "Reactor Safety Study, An Assessment of Accident Risks in U. S. Commercial Nuclear Power Plants."
- ²CSC-RL-SCI-122, July 1975, J. N. Diven, D. E. Griffing, A. Thorpe, and W. J. Van Slyke, "Fault Tree Analysis System."
- ³Jerry B. Fussel and W. E. Vesely, "A New Methodology for Obtaining Cut Sets for Fault Trees," American Nuclear Transactions, Volume 15, No. 1, June 1972, pp. 262-263.
- ⁴Jerry B. Fussel, E. B. Henry, and N. H. Marshall, "Mocus-A Computer Program to Obtain Minimal Sets from Fault Trees," Aerojet Nuclear Company, Idaho Falls, Idaho, March 1974.
- ⁵Class handouts for the course, "System Safety and Reliability Analysis," Lamba-Tau Method section, authored and instructed by N. H. Roberts, June 18-29, University of Washington, Seattle, Washington 98105. Available from Professor N. H. Roberts or Professor W. Kielsing, Mechanical Engineering Dept., University of Washington.
- ⁶Fault Tree analysis Instructional Example, "Pressure Tank System," from course presented by D. F. Haasl, Joint Center for Graduate Study, Richland, Washington, February 19 - March 1, 1974. Available from David F. Haasl, 3085 North Bay Drive, North Bend, Oregon 97459.
- ⁷T. H. Smith, W. K. Winegardner, G. Jansen, Jr., L. D. Williams, and T. I. McSweeney, "A Methodology for Risk Analysis of Nuclear Waste Management Systems," BNWL-SA-4899, Battelle Pacific Northwest Laboratories, June 1974.

(This page has been intentionally left blank)

BIBLIOGRAPHY

Fault Tree Analysis: Min Cut Set Algorithms, P. Chatterjee, ORC 74-2, Operations Research Center, College of Engineering, University of California, Berkeley, California, January 1974.

Advanced Concepts in Fault Tree Analysis, David F. Haasl, presented at System Safety Symposium, sponsored by University of Washington and the Boeing Company, Olympic Hotel, Seattle, Wash., June 8-9, 1965.

Computerized Fault Tree Analysis: TREEL and MICSUP, P. K. Pande, M. E. Spector, and P. Chatterjee, ORC 75-3, Operations Research Center, College of Engineering, University of California, Berkeley, California, April 1975.

(This page has been intentionally left blank)

APPENDIX A

FAULT TREE EXAMPLE

The following material presents a hypothetical "pressure tank" system.*

A diagrammatic representation of the system is shown in Figure A-1.

The system includes a pressure tank, a pressurizing pump, a start push-button switch S1, a pressure switch, and two relays, K1 and K2.

The normal operating sequence for the system begins when an operator momentarily pushes switch S1, causing relay K1 to latch in. K2 relay then actuates, enabling electrical current from the power supply to run the pump. When the pressure tank reaches the desired pressure, the pressure switch opens, interrupting power to the pump. The timer relay then resets to zero time without its contacts opening. Whenever the tank pressure drops to below the pressure switch lower limit, its contacts close K2 relay actuates, and the pump operates to build up pressure in the tank.

In the event that the pressure switch ever fails to open within 60 seconds, the pump will operate for 60 seconds and then the timer relay contacts will open. This unlatches relay K1, deactivates relay K2 and the pump, and restores the system to its prestart status.

We wish to discover what conditions might operate to cause the pressure

*Please see the Acknowledgments.

tank to rupture. Using fault tree analysis methodology, the first step is to complete a fault tree diagram for the system. A valid diagram for the pressure tank system is presented in Figure A-2, as plotted by a computerized fault tree diagramming code, KILMER.⁽²⁾ The diagram was synthesized using the same input data as used by ALLCUTS,

Appendix B.

Figure A-1

Fault Tree Example, "Pressure Tank System"

Figure A-2
Fault Tree for the "Pressure Tank System"

APPENDIX B

ALLCUTS INPUT DATA EXAMPLE

The following four pages are a listing of the input data which was used by program ALLCUTS to perform the analysis results presented in Appendix C. This data was also used by the Fault Tree Diagram plotting program to produce the diagram of Figure A-2. The data file listing has been double spaced for improved readability.

FAULT TREE ANALYSIS PROGRAM DEMONSTRATION EXAMPLE.

BASED ON A FAULT TREE ANALYSIS INSTRUCTIONAL EXAMPLE.

"PRESSURE TANK SYSTEM", FROM A FAULT TREE ANALYSIS COURSE
 TAUGHT BY DAVID F. HAASL

AT THE JOINT CENTER FOR GRADUATE STUDY,

RICHLAND, WASHINGTON, ON FEBRUARY 19 THROUGH
 MARCH 1, 1974.

PERMISSION TO USE THIS EXAMPLE MATERIAL IS
 GRATEFULLY ACKNOWLEDGED.

*DATA

10 1 100 1 3 3 1 0

1.-99 1.+04 PR65 PR61

END

HOLLERITH (OR COMMENT) LINES MAY BE INSERTED JUST PRECEDING

ANY DATA SECTION INITIAL CARD WHICH HAS AN ASTERISK

IN COLUMN 1. THESE INITIAL CARDS ARE, "**DATA", "**TREE",

TOPS", AND "RATES".

*TREE

PR OR 12 PR11 PR12 PR13

PRESSURE TANK RUPTURE

PR11 OR 11 PR21 PR22

SECONDARY FAILURE OF PRESSURE TANK - RUPTURED.

PR21 INH 10 PR31

TANK RUPTURE DUE TO INTERNAL OVERPRESSURE FROM PUMP OPERATION, $T > 60$ SEC.

MOD

OVER PRESSURE FROM PUMPING FOR $T > 60$ SEC. IS SUFFICIENT TO RUPTURE TANK.

PR31 OR 12 PR43 PR41 PR42

K2 RELAY CONTACTS REMAIN CLOSED FOR T GREATER THAN 60 SECONDS.

MOD

PUMP OPERATING CONTINUOUSLY FOR T GREATER THAN 60 SECONDS.

PR43 AND 20 PR51 PR52

EMF TO K2 RELAY COIL FOR T GREATER THAN 60 SECONDS.

PR51 OR 03 PR61 PR62 PR63

PRESSURE SWITCH CONTACTS CLOSED FOR T GREATER THAN 60 SECONDS.

PR52 OR 20 PR64 PR65

EMF REMAINS ON PRESSURE SWITCH CONTACTS WHEN P/S CLOSED, $T > 60$ SEC.

PR64 OR 12 PR71 PR72 PR73

EMF THRU K1 RELAY CONTACTS WHEN P/S CONTACTS CLOSED FOR $T > 60$ SECONDS.

PR71 OR 03 PR81 PR82 PR83

TIMER RELAY CONTACTS FAIL TO OPEN WHEN P/S CONTACTS CLOSED, $T > 60$ SEC.

MOD

EMF NOT REMOVED FROM K1 RELAY COIL WHEN P/S CONTACTS CLOSED FOR $T > 60$ SECONDS.

PR65 OR 03 PR74 PR75 PR76

EMF THROUGH S1 SWITCH WHEN P/S CONTACTS CLOSED FOR $T > 60$ SECONDS.

END

END

THIS IS THE POSITION FOR INSERTING THE OPTIONAL THREE CARD SET,

WHICH HAS "TOFS" IN COLUMNS 1 THROUGH 5 OF THE FIRST CARD,

HAS THE GATE NAME OF THE ALTERNATIVE GATE FOR THE TOP EVENT

OF THE ANALYSIS ON THE SECOND CARD BEGINNING IN CARD COLUMN

ONE, AND HAS "END" IN COLUMNS 1 THROUGH 3 OF THE THIRD CARD.

THE DATA IN THE FOLLOWING, "RATES" SECTION HAS BEEN USED SOLELY TO

ILLUSTRATE THE FORMAT FOR THE PROBABILITY AND REPAIR TIME

PARAMETERS. A VALUE OF ONE WAS USED FOR ALL DATA. THE DATA

CAN BE OMITTED IF NO PROBABILITY CALCULATIONS WILL BE ATTEMPTED

BY PROGRAM "ALLOUTS".

RATES

PR12 CRC 1. 1. 1. 1. 1.

PRESSURE TANK RUPTURES UNDER LOAD.

PR13 DMD 1. 1. 1. 1. 1.

TANK RUPTURED DUE TO IMPROPER SELECTION OR INSTALLATION.

PR22 DMD 1. 1. 1. 1. 1.

SECONDARY FAILURE OF TANK FROM OTHER OUT-OF-TOLERANCE CONDITIONS (E.G. MECH.)

PR41 DMD 1. 1. 1. 1. 1.

K2 RELAY SECONDARY FAILURE.

PR42 CRC 1. 1. 1. 1. 1.

K2 RELAY CONTACTS FAIL TO OPEN.

PR61 DMD 1. 1. 1. 1. 1.

EXCESS PRESSURE NOT SENSED BY PRESSURE ACTIVATED SWITCH.

PR62 CRC 1. 1. 1. 1. 1.

PRESSURE SWITCH - CONTACTS FAIL TO OPEN.

PR63 DMD 1. 1. 1. 1. 1.

PRESSURE SWITCH SECONDARY FAILURE.

PR72 CRC 1. 1. 1. 1. 1.

K1 RELAY CONTACTS FAIL TO OPEN.

PR73	DMD	1.	1.	1.	1.	1.
K1 RELAY - SECONDARY FAILURE.						
PR74	DMD	1.	1.	1.	1.	1.
EXTERNAL RESET	ACTUATION FORCE			REMAINS ON		SWITCH S1.
PR75	CRC	1.	1.	1.	1.	1.
S1 SWITCH CONTACTS FAIL TO OPEN.						
PR76	DMD	1.	1.	1.	1.	1.
S1 SWITCH SECONDARY FAILURE.						
PR81	DMD	1.	1.	1.	1.	1.
TIMER DOES NOT TIME OUT DUE TO IMPROPER SETTING OR					INSTALLATION.	
PR82	CRC	1.	1.	1.	1.	1.
TIMER RELAY -	CONTACTS FAIL TO			OPEN.		
PR83	DMD	1.	1.	1.	1.	1.
TIMER RELAY -	SECONDARY FAILURE.					
END						
END						

(This page has been intentionally left blank)

APPENDIX C

OUTPUT DATA LISTING FROM ALLCUTS FOR THE EXAMPLE DATA

The following pages of this appendix are reproductions of the ALLCUTS output data which was generated by the input data listed in Appendix B. Only one set of output options is presented here from the multiple sets of options which the user will want to use. This particular set of options lists all the input data, lists gate paths through gate PR65, and lists all cutsets containing the bottom event PR61 in the order in which they were found. The cutsets listing follows.

The table of bottom event importances and importance factors was not requested, but it could not have been printed even if requested because the value of IDEX1 did not allow probability calculations to be made. The input data resulted in 5 single event and 24 double event cutsets being found and listed without calculated probabilities. No higher order cutsets exist for this tree, but ALLCUTS generated additional pages, blank except for page headings similar to those at the tops of the single event and double event cutsets listing. These pages were not included in the listing below.

* * * * *
* ALLCUTS, FAULT TREE ANALYSIS PROGRAM
* * * * *

FAULT TREE ANALYSIS PROGRAM DEMONSTRATION EXAMPLE:

BASED ON A FAULT TREE ANALYSIS INSTRUCTIONAL EXAMPLE,
"PRESSURE TANK SYSTEM", FROM A FAULT TREE ANALYSIS COURSE
TAUGHT BY DAVID F. HAASL
AT THE JOINT CENTER FOR GRADUATE STUDY,
RICHLAND, WASHINGTON, ON FEBRUARY 19 THROUGH
MARCH 1, 1974.

PERMISSION TO USE THIS EXAMPLE MATERIAL IS
GRATEFULLY ACKNOWLEDGED.

*DATA

MAXIMUM SET SIZE,	MAX-----	10
MINIMUM SET SIZE,	MIN-----	1
MAXIMUM NO. OF SETS SAVED,	MSS-----	100
CALCULATION OPTION SWITCH,	IDEX1---	1
EVENT WRITE SWITCH,	IDEX2---	3
ABSORPTION CHECK SWITCH,	NABS----	3
DATA PRINT SWITCH,	NDPR----	1
DATA TRANSFER SWITCH,	NTRANS--	0
MINIMUM PROBABILITY,	PNIN----	1.000000E-99
MISSION TIME,	TIMEM---	1.000000E+04
TRACE GATE LABEL,	GTRACE--	PR65
SET IDENT. EVENT,	CTRACE--	PR61

HOLLERITH (OR COMMENT) LINES MAY BE INSERTED JUST PRECEDING
ANY DATA SECTION INITIAL CARD WHICH HAS AN ASTERISK
IN COLUMN 1. THESE INITIAL CARDS ARE, "**DATA", "**TREE",
"**TOPS", AND "**RATES".

*TREE

* * * * *

* ALLCUTS, FAULT TREE ANALYSIS PROGRAM

* * * * *

FAULT TREE ANALYSIS PROGRAM DEMONSTRATION EXAMPLE.

INDEX GATE NAME TYPE INPUTS

1	PR	OR	1.00000	12	PR11	PR12	PR13	
		PRESSURE TANK			RUPTURE			
2	PR11	OR	1.00000	11	PR21	PR22		
		SECONDARY FAILURE			OF PRESSURE TANK -		RUPTURED.	
3	PR21	INH	1.00000	10	FR31			
		TANK RUPTURE DUE TO			INTERNAL OVERPRES-		SURE FROM PUMP OPER-	ATION, T > 60 SEC.
4		MOD	1.00000	-0				
		OVER PRESSURE FROM			PUMPING FOR T > 60		SEC. IS SUFFICIENT	TO RUPTURE TANK.
4	PR31	OR	1.00000	12	PR43	PR41	PR42	
		K2 RELAY CONTACTS			REMAIN CLOSED FOR		T GREATER THAN	60 SECONDS.
5		MOD	1.00000	-0				
		PUMP OPERATING			CONTINUOUSLY FOR		T GREATER THAN	60 SECONDS.
5	PR43	AND	1.00000	20	FR51	PR52		
		EMF TO K2 RELAY			COIL FOR T GREATER		THAN 60 SECONDS.	
6	PR51	OR	1.00000	3	PR61	PR62	PR63	
		PRESSURE SWITCH			CONTACTS CLOSED FOR		T GREATER THAN	60 SECONDS.
7	PR52	OR	1.00000	20	PR64	PR65		
		EMF REMAINS ON			PRESSURE SWITCH		CONTACTS WHEN P/S	CLOSED, T > 60 SEC.
8	PR64	OR	1.00000	12	PR71	PR72	PR73	
		EMF THRU K1 RELAY			CONTACTS WHEN P/S		CONTACTS CLOSED FOR	T > 60 SECONDS.
9	PR71	OR	1.00000	3	PR81	PR82	PR83	
		TIMER RELAY CON-			TACTS FAIL TO OPEN		WHEN P/S CONTACTS	CLOSED, T > 60 SEC.
10		MOD	1.00000	-0				
		EMF NOT REMOVED FROM			K1 RELAY COIL WHEN		P/S CONTACTS CLOSED	FOR T > 60 SECONDS.
10	PR65	OR	1.00000	3	PR74	PR75	PR76	
		EMF THROUGH S1			SWITCH WHEN P/S		CONTACTS CLOSED FOR	T > 60 SECONDS.
11	END		1.00000	-0				
		END						

* * * * *
* ALLCUTS, FAULT TREE ANALYSIS PROGRAM
* * * * *

FAULT TREE ANALYSIS PROGRAM DEMONSTRATION EXAMPLE.

THE TOP GATE OF THIS TREE IS PR

THIS IS THE POSITION FOR INSERTING THE OPTIONAL THREE CARD SET,

WHICH HAS "*TOPS" IN COLUMNS 1 THROUGH 5 OF THE FIRST CARD,

HAS THE GATE NAME OF THE ALTERNATIVE GATE FOR THE TOP EVENT

OF THE ANALYSIS ON THE SECOND CARD BEGINNING IN CARD COLUMN

ONE, AND HAS "ENC" IN COLUMNS 1 THROUGH 3 OF THE THIRD CARD.

THE DATA IN THE FOLLOWING, "*RATES" SECTION HAS BEEN USED SOLELY TO

ILLUSTRATE THE FORMAT FOR THE PROBABILITY AND REPAIR TIME

PARAMETERS. A VALUE OF ONE WAS USED FOR ALL DATA. THE DATA

CAN BE OMITTED IF NO PROBABILITY CALCULATIONS WILL BE ATTEMPTED

BY PROGRAM "ALLCUTS".

*RATES

 * ALLCUTS, FAULT TREE ANALYSIS PROGRAM

FAULT TREE ANALYSIS PROGRAM DEMONSTRATION EXAMPLE.

COMPONENT NAMES, TYPES, AND FAILURE/REPAIR DATA-

INDEX	EVENT NAME	TYPE	LAMDA-10E6	DETECT. TIME	REPAIR TIME	MAX. TIME FAILED	UNAVAILABILITY
1	PR12 PRESSURE TANK	GRG	1.00000E+00	1.0000 RUPTURES UNDER LOAD.	1.0000	1.00000E+00	1.0000
2	PR13 TANK RUPTURED DUE TO IMPROPER	DMD	1.00000E+00	1.0000 SELECTION OR	1.0000	1.00000E+00 INSTALLATION.	1.0000
3	PR22 SECONDARY FAILURE OF TANK FROM OTHER OUT-	DMD	1.00000E+00	1.0000 OF-TOLERANCE CONDI-	1.0000	1.00000E+00 TIONS (E.G. MECH.)	1.0000
4	PR41 K2 RELAY SECONDARY FAILURE.	DMD	1.00000E+00	1.0000	1.0000	1.00000E+00	1.0000
5	PR42 K2 RELAY CONTACTS FAIL TO OPEN.	GRG	1.00000E+00	1.0000	1.0000	1.00000E+00	1.0000
6	PR61 EXCESS PRESSURE NOT SENSED BY PRESSURE	DMD	1.00000E+00	1.0000 ACTIVATED SWITCH.	1.0000	1.00000E+00	1.0000
7	PR62 PRESSURE SWITCH - CONTACTS FAIL TO	GRG	1.00000E+00	1.0000 OPEN.	1.0000	1.00000E+00	1.0000
8	PR63 PRESSURE SWITCH SECONDARY FAILURE.	DMD	1.00000E+00	1.0000	1.0000	1.00000E+00	1.0000
9	PR72 K1 RELAY CONTACTS FAIL TO OPEN.	GRG	1.00000E+00	1.0000	1.0000	1.00000E+00	1.0000
10	PR73 K1 RELAY - SECONDARY FAILURE.	DMD	1.00000E+00	1.0000	1.0000	1.00000E+00	1.0000
11	PR74 EXTERNAL RESET ACTUATION FORCE	DMD	1.00000E+00	1.0000 REMAINS ON	1.0000	1.00000E+00 SWITCH S1.	1.0000
12	PR75 S1 SWITCH CONTACTS FAIL TO OPEN.	GRG	1.00000E+00	1.0000	1.0000	1.00000E+00	1.0000
13	PR76 S1 SWITCH SECONDARY FAILURE.	DMD	1.00000E+00	1.0000	1.0000	1.00000E+00	1.0000
14	PR81 TIMER DOES NOT TIME OUT DUE TO IMPROPER	DMD	1.00000E+00	1.0000 SETTING OR	1.0000	1.00000E+00 INSTALLATION.	1.0000

15	PR62	CRC	1.00000E+00	1.0000	1.0000	1.00000E+00	1.0000
	TIMER RELAY -		CONTACTS FAIL TO	OPEN.			
16	PR63	DMD	1.00000E+00	1.0000	1.0000	1.00000E+00	1.0000
	TIMER RELAY -		SECONDARY FAILURE.				
17	END		-0.	-0.0000	-0.0000	-0.	-0.0000
	END						

* * * * * THE INPUT DATA LISTED 10 GATES AND 16 BOTTOM EVENTS.

SET CONTAINING THE BOTTOM EVENT PR61

1	PR61	EXCESS PRESSURE NOT	SENSED BY PRESSURE	ACTIVATED SWITCH.	
2	PR61	TIMER DOES NOT TIME	OUT DUE TO IMPROPER	SETTING OR	INSTALLATION.

SET CONTAINING THE BOTTOM EVENT PR61

1	PR61	EXCESS PRESSURE NOT	SENSED BY PRESSURE	ACTIVATED SWITCH.
2	PR63	TIMER RELAY -	SECONDARY FAILURE.	

SET CONTAINING THE BOTTOM EVENT PR61

1	PR61	EXCESS PRESSURE NOT	SENSED BY PRESSURE	ACTIVATED SWITCH.
2	PR62	TIMER RELAY -	CONTACTS FAIL TO	OPEN.

SET CONTAINING THE BOTTOM EVENT PR61

1	PR61	EXCESS PRESSURE NOT	SENSED BY PRESSURE	ACTIVATED SWITCH.
2	PR73	K1 RELAY -	SECONDARY FAILURE.	

SET CONTAINING THE BOTTOM EVENT PR61

1	PR61	EXCESS PRESSURE NOT	SENSED BY PRESSURE	ACTIVATED SWITCH.
2	PR72	K1 RELAY CONTACTS	FAIL TO OPEN.	

PATH TRACE FOR GATE PR65

PR PR11 PR21 PR31 PR43 PR52 PR65

SET CONTAINING THE BOTTOM EVENT PR61

1	PR61	EXCESS PRESSURE NOT	SENSED BY PRESSURE	ACTIVATED SWITCH.	
2	PR74	EXTERNAL RESET	ACTUATION FORCE	REMAINS ON	SWITCH S1.

SET CONTAINING THE BOTTOM EVENT PR61

1	PR61	EXCESS PRESSURE NOT	SENSED BY PRESSURE	ACTIVATED SWITCH.
2	PR76	S1 SWITCH	SECONDARY FAILURE.	

SET CONTAINING THE BOTTOM EVENT PR61

1	PR61	EXCESS PRESSURE NOT	SENSED BY PRESSURE	ACTIVATED SWITCH.
2	PR75	S1 SWITCH CONTACTS	FAIL TO OPEN.	

PATH TRACE FOR GATE PR65

PR PR11 PR21 PR31 PR43 PR52 PR65

PATH TRACE FOR GATE PR65

PR PR11 PR21 PR31 PR43 PR52 PR65

THE MAX NUMBER OF SETS STORED INTERNALLY WAS 13

THIS TREE HAS 29 CUTSETS CONTAINING
FROM 1 TO 10 EVENTS

THE TOTAL NUMBER OF SETS GENERATED WAS 29

THE TOP GATE OF THIS TREE IS PR

THERE ARE 29 SETS REMAINING AFTER ABSORPTION CHECKING.

SETS WITH 1 COMPONENTS---

CUT SET NO.	1	NUMBER	1	OF THIS GROUP.		
1 PR42		K2 RELAY CONTACTS		FAIL TO OPEN.		
CUT SET NO.	2	NUMBER	2	OF THIS GROUP.		
1 PR41		K2 RELAY SECONDARY		FAILURE.		
CUT SET NO.	3	NUMBER	3	OF THIS GROUP.		
1 PR22		SECONDARY FAILURE OF TANK FROM OTHER		OUT- OF-TOLERANCE CONDI- TIONS (E.G. MECH.)		
CUT SET NO.	4	NUMBER	4	OF THIS GROUP.		
1 PR13		TANK RUPTURED DUE		TO IMPROPER	SELECTION OF	INSTALLATION.
CUT SET NO.	5	NUMBER	5	OF THIS GROUP.		
1 PR12		PRESSURE TANK		RUPTURES UNDER LOAD.		

SETS WITH 2 COMPONENTS---

CUT SET NO.	6	NUMBER	1	OF THIS GROUP.		
1 PR61		EXCESS PRESSURE NOT	SENSED BY PRESSURE	ACTIVATED SWITCH.		
2 PR81		TIMER DOES NOT TIME	OUT DUE TO IMPROPER	SETTING OR		INSTALLATION.
CUT SET NO.	7	NUMBER	2	OF THIS GROUP.		
1 PR61		EXCESS PRESSURE NOT	SENSED BY PRESSURE	ACTIVATED SWITCH.		
2 PR83		TIMER RELAY -	SECONDARY FAILURE.			
CUT SET NO.	8	NUMBER	3	OF THIS GROUP.		
1 PR61		EXCESS PRESSURE NOT	SENSED BY PRESSURE	ACTIVATED SWITCH.		
2 PR82		TIMER RELAY -	CONTACTS FAIL TO	OPEN.		
CUT SET NO.	9	NUMBER	4	OF THIS GROUP.		
1 PR61		EXCESS PRESSURE NOT	SENSED BY PRESSURE	ACTIVATED SWITCH.		
2 PR73		K1 RELAY - SECONDARY	FAILURE.			
CUT SET NO.	10	NUMBER	5	OF THIS GROUP.		
1 PR61		EXCESS PRESSURE NOT	SENSED BY PRESSURE	ACTIVATED SWITCH.		
2 PR72		K1 RELAY CONTACTS	FAIL TO OPEN.			
CUT SET NO.	11	NUMBER	6	OF THIS GROUP.		
1 PR61		EXCESS PRESSURE NOT	SENSED BY PRESSURE	ACTIVATED SWITCH.		
2 PR74		EXTERNAL RESET	ACTUATION FORCE	REMAINS ON		SWITCH S1.
CUT SET NO.	12	NUMBER	7	OF THIS GROUP.		
1 PR61		EXCESS PRESSURE NOT	SENSED BY PRESSURE	ACTIVATED SWITCH.		
2 PR76		S1 SWITCH SECONDARY	FAILURE.			
CUT SET NO.	13	NUMBER	8	OF THIS GROUP.		
1 PR61		EXCESS PRESSURE NOT	SENSED BY PRESSURE	ACTIVATED SWITCH.		
2 PR75		S1 SWITCH CONTACTS	FAIL TO OPEN.			
CUT SET NO.	14	NUMBER	9	OF THIS GROUP.		
1 PR63		PRESSURE SWITCH	SECONDARY FAILURE.			
2 PR81		TIMER DOES NOT TIME	OUT DUE TO IMPROPER	SETTING OR		INSTALLATION.
CUT SET NO.	15	NUMBER	10	OF THIS GROUP.		
1 PR63		PRESSURE SWITCH	SECONDARY FAILURE.			
2 PR83		TIMER RELAY -	SECONDARY FAILURE.			
CUT SET NO.	16	NUMBER	11	OF THIS GROUP.		
1 PR63		PRESSURE SWITCH	SECONDARY FAILURE.			
2 PR82		TIMER RELAY -	CONTACTS FAIL TO	OPEN.		

CUT SET NO.	17	NUMBER 12 OF THIS GROUP.		
1 PR63		PRESSURE SWITCH	SECONDARY FAILURE.	
2 PR73		K1 RELAY - SECONDARY	FAILURE.	
CUT SET NO.	18	NUMBER 13 OF THIS GROUP.		
1 PR63		PRESSURE SWITCH	SECONDARY FAILURE.	
2 PR72		K1 RELAY CONTACTS	FAIL TO OPEN.	
CUT SET NO.	19	NUMBER 14 OF THIS GROUP.		
1 PR63		PRESSURE SWITCH	SECONDARY FAILURE.	
2 PR74		EXTERNAL RESET	ACTUATION FORCE	REMAINS ON SWITCH S1.
CUT SET NO.	20	NUMBER 15 OF THIS GROUP.		
1 PR63		PRESSURE SWITCH	SECONDARY FAILURE.	
2 PR76		S1 SWITCH SECONDARY	FAILURE.	
CUT SET NO.	21	NUMBER 16 OF THIS GROUP.		
1 PR63		PRESSURE SWITCH	SECONDARY FAILURE.	
2 PR75		S1 SWITCH CONTACTS	FAIL TO OPEN.	
CUT SET NO.	22	NUMBER 17 OF THIS GROUP.		
1 PR62		PRESSURE SWITCH -	CONTACTS FAIL TO	OPEN.
2 PR81		TIMER DOES NOT TIME	OUT DUE TO IMPROPER	SETTING OR INSTALLATION.
CUT SET NO.	23	NUMBER 18 OF THIS GROUP.		
1 PR62		PRESSURE SWITCH -	CONTACTS FAIL TO	OPEN.
2 PR83		TIMER RELAY -	SECONDARY FAILURE.	
CUT SET NO.	24	NUMBER 19 OF THIS GROUP.		
1 PR62		PRESSURE SWITCH -	CONTACTS FAIL TO	OPEN.
2 PR82		TIMER RELAY -	CONTACTS FAIL TO	OPEN.
CUT SET NO.	25	NUMBER 20 OF THIS GROUP.		
1 PR62		PRESSURE SWITCH -	CONTACTS FAIL TO	OPEN.
2 PR73		K1 RELAY - SECONDARY	FAILURE.	
CUT SET NO.	26	NUMBER 21 OF THIS GROUP.		
1 PR62		PRESSURE SWITCH -	CONTACTS FAIL TO	OPEN.
2 PR72		K1 RELAY CONTACTS	FAIL TO OPEN.	
CUT SET NO.	27	NUMBER 22 OF THIS GROUP.		
1 PR62		PRESSURE SWITCH -	CONTACTS FAIL TO	OPEN.
2 PR74		EXTERNAL RESET	ACTUATION FORCE	REMAINS ON SWITCH S1.
CUT SET NO.	28	NUMBER 23 OF THIS GROUP.		
1 PR62		PRESSURE SWITCH -	CONTACTS FAIL TO	OPEN.
2 PR76		S1 SWITCH SECONDARY	FAILURE.	
CUT SET NO.	29	NUMBER 24 OF THIS GROUP.		
1 PR62		PRESSURE SWITCH -	CONTACTS FAIL TO	OPEN.
2 PR75		S1 SWITCH CONTACTS	FAIL TO OPEN.	

(This page has been intentionally left blank)

APPENDIX D

OUTPUT DATA LISTING FROM BRANCH FOR THE EXAMPLE DATA

The following two pages reproduce the output from program BRANCH for the example data. The output always appears in two sections. The first of these details gate-to-gate interrelationships and the second covers bottom events and the gates to which they are input. This type of output listing becomes extremely useful for large trees which have large numbers of transfers and/or multiple bottom events each of which are input to more than one gate.

GATE NAME	GATE TYPE	INPUTS TO THIS GATE			THIS GATE IS INPUT TO GATES:
PR	OR	FR11	*PR12	*PR13	PR
PR11	OR	FR21	*PR22		PR11
PR21	INHIBIT	PR31			PR21
PR31	OR	FR43	*PR41	*PR42	PR31
FR43	AND	FR51	PR52		PR43
FR51	OR	*FR61	*PR62	*PR63	PR43
FR52	OR	FR64	PR65		PR52
PR64	OR	FR71	*PR72	*PR73	PR64
PR71	OR	*FR81	*PR82	*PR83	PR52
PR65	OR	*FR74	*PR75	*PR76	

BOTTOM EVENT NAME -----	BOTTOM EVENT TYPE -----	THIS BOTTOM EVENT IS INPUT TO GATES: -----
PR12	CIRCLE	PR
PR13	DIAMOND	PR
PR22	DIAMOND	PR11
PR41	DIAMOND	PR31
PR42	CIRCLE	PR31
PR61	DIAMOND	PR51
PR62	CIRCLE	PR51
PR63	DIAMOND	PR51
PR72	CIRCLE	PR64
PR73	DIAMOND	PR64
PR74	DIAMOND	PR65
PR75	CIRCLE	PR65
PR76	DIAMOND	PR65
PR81	DIAMOND	PR71
PR82	CIRCLE	PR71
PR83	DIAMOND	PR71

2110-100

APPENDIX E

SOURCE LANGUAGE LISTING FOR PROGRAM ALLCUTS

The following pages are the source language listing for ALLCUTS. It is written in Fortran IV language as implemented on CDC Cyber 70 Computer operated by Computer Sciences Corporation at Richland, Washington. The computer memory space required by the code is strongly dependent on the size of the fault tree to be analyzed and on the number of cutsets stored internally for probability and/or cutset size ordering. The dimensions of the program may be readily adjusted by changing the appropriate array dimensions. The number of gates stored is presently controlled by those program and subroutine arrays which have a dimension equal to 425, and by the variable MGATES, card sequence number 650.

The number of bottom events stored is specified by these program and subroutine arrays presently dimensioned 175, and by the variable MCOMPS, card sequence number 660.

The number of cutsets stored internally is presently controlled by these main program variables which are dimensioned 1001, and by variable MSSP1, card sequence number 630.

Liberal use of comment statements was made to provide aid to the analyst who wishes to modify the code. A tape, source language, or card deck copy of these codes will be available from Argonne Code Center in the near future.⁽³⁾

```

PROGRAM ALLCUTS (OUTPUT,80P,DT,TOUT,TAPE5=DT,TAPE6=80P,TAPE7=TOUT) 000100
C *** 000110
C *** CONCEIVED AND PROGRAMMED BY 000120
C *** WILLIAM J. VAN SLYKE 000130
C *** ATLANTIC RICHFIELD HANFORD COMPANY 000140
C *** P. O. BOX 250 000150
C *** RICHLAND, WASHINGTON 99352 000160
C *** 000170
C *** 000180
C *** REFERENCE TO THE SOURCE AND USE OF THIS PROGRAM AS FOLLOWS 000190
C *** WILL BE APPRECIATED: 000200
C *** 000210
C *** W. J. VAN SLYKE AND D. E. GRIFFING, 000220
C *** "ALLCUTS, A FAST, COMPREHENSIVE FAULT TREE ANALYSIS CODE.", 000230
C *** ARH-ST-112, ATLANTIC RICHFIELD HANFORD COMPANY, JULY, 1975. 000240
C *** 000250
REAL TITLE(10),BEV(8,175),TO(175),TR(175), 000260
1 Q(175),LAMBDA,LAMDA(175),SENSIT(175), 000270
2 BUFR(11),GP(425),SETP(100),BLURB(8),SETPS(1001) 000280
INTEGER NGC(425),GATE(9,425),SET(10,100),CARD(11),TOPWUN 000290
2 ,ERROR,NVNS(100),AND,OR,INH,MOD,DATA,TREE,RATES,TOPS,GTRACE 000300
3 ,NAME,NAM(175),END,BETYP(175),CRC,OKD,HSE,COMP,ASK,CTRACE,BLNK7 000310
5 ,NLEVL5(10,50),NGP(50),SSET(10,1001),NVNSS(1001) 000320
LOGICAL GTRACK,CTACK 000330
COMMON GATE,SET,GP,SETP,NGC,NGP,NLEVL5,TO,TR,Q,LAMDA, 000340
2 NVNS,MAX,NMAX,MIN,MCOMPS,MGATES,NG,NC,NSETS,LEVL5 000350
C***** 000360

```

C *** INITIALIZE HOLLORITH DATA	000370
C*****	000380
DATA AND/3HAND/,BLNK1/1H /,BLNK7/7H /,	000390
1 OR/2HOR/,TTOP/3HTOP/,END/3HEND/	000400
DATA ASK/1H*//,DATA/7HDATA /,TREE/7HTREE /	000410
DATA RATES/7HRATES /,INH/3HINH/,MOD/3HMOD/	000420
DATA DMD/3HDMD/,CRC/3HCRC/,HSE/3HHSE/,TOPS/4HTOPS/	000430
C*****	000440
C *** INITIALIZATION SECTION	000450
C*****	000460
REWIND 5	000470
REWIND 6	000480
REWIND 7	000490
C*****	000500
C *** TAPES IS INPUT FILE, TAPE6 IS OUTPUT FILE.	000510
C *** TAPE7 IS THE FILE TO TRANSFER DATA FOR SUBSEQUENT PROCESSING.	000520
C*****	000530
NSETS=0	000540
NG = 0	000550
GTRACK= .F.	000560
CTrack= .F.	000570
INDEX = 0	000580
NABITS=0	000590
MATCH = 0	000600
MLEVELS=50	000610
MSETS=100	000620
MSSP1=1001	000630

MVNS=10	000640
MGATES=425	000650
MCOMPS=175	000660
SLSUM=0.	000670
SPSUM=0.	000680
CRITL=1.E-6	000690
CRITU=500.	000700
DO 3 I=1,MSETS	000710
SETP(I)=1.0	000720
DO 2 J=1,MVNS	000730
SET(J,I)=0	000740
2 CONTINUE	000750
3 CONTINUE	000760
DO 4 J=1,100	000770
4 NGP(J)=0	000780
DO 6 J=1,MCOMPS	000790
GP(J)=0.	000800
NGC(J)=0	000810
DO 5 I=1,9	000820
5 GATE(I,J) = 0	000830
6 CONTINUE	000840
C*****	000850
C *** READ TITLE CARD	000860
C*****	000870
READ(5,8) TITLE	000880
8 FORMAT(10A8)	000890
C*****	000900

C *** READ COMMENTS AND CONTROL CARD FOR TREE DATA	000910
C*****	000920
WRITE(6, 15)	000930
15 FORMAT(1H1,7(2H *)/1H ,1H*,* ALLCUTS, FAULT TREE ANALYSIS PROGRAM*	000940
1/1H ,7(2H *)/1H-)	000950
WRITE(6, 20) TITLE	000960
20 FORMAT(1X,10A8)	000970
25 READ(5, 40) CARD	000980
IF (EOF(5)) 270,30,270	000990
30 CONTINUE	001000
35 FORMAT(1H0,1X,A1,A7,9A8)	001010
40 FORMAT(A1,A7,9A8)	001020
WRITE (6,35) CARD	001030
IF(CARD(1) .NE. ASK) GO TO 25	001040
C*****	001050
C *** HOLLERITH DATA MAY BE INSERTED BEFORE EACH DATA TYPE CARD	001060
C *** FOR PRINT OUT AS SUCH BY THE PROGRAM.	001070
C*****	001080
IF (CARD(2) .NE. DATA) GO TO 320	001090
C*****	001100
C *** READ CONTROL PARAMETERS	001110
C*****	001120
READ(5,50) MAX,MIN,NSS,IOEX1,IOEX2,NABS,NOPR,NTRANS,PMIN,	001130
2 TIMEM,GTRACE,CTRACE	001140
IF (IOEX1 .LT. 1 .OR. IOEX1 .GT. 8) MAX=0	001150
IF (MAX .GT. MVNS) MAX=MVNS	001160
IF (MIN .GT. MAX .OR. MIN .LT. 0) MIN=1	001170

```

      NMAX=MAX                                001180
      IF (MAX .LT. MVNS) NMAX=MAX+1           001190
      PRM=PMIN                                001200
      IF (IDEX1 .GT. 1 .AND. IDEX1 .LT. 7) PRM=1. 001210
      IF (NABS .LT. 1 .OR. NABS .GT. 3) NABS=1 001220
      IF (MSS .GE. MSSP1 .OR. MSS .LE. 0) MSS=MSSP1 - 1 001230
      IF (GTRACE .NE. BLNK7) GTRACK = .T.    001240
      IF (CTRACE .NE. BLNK7) CTRACK = .T.    001250
      C*****                                001260
      C *** "MAX" IS THE MAXIMUM NUMBER OF COMBINED BOTTOM EVENTS TO 001270
      C *** CAUSE SYSTEM FAILURE. MAX LESS THAN 1 DIRECTS THE PROGRAM TO 001280
      C *** PERFORM ERROR CHECKING ON THE DATA FILE ONLY AND THEN STOP. 001290
      C ***                                001300
      C *** "MIN" IS THE MINIMUM NUMBER OF COMBINED BOTTOM EVENTS TO 001310
      C *** CAUSE SYSTEM FAILURE. MIN .GT. MAX WILL RESULT IN MIN BEING 001320
      C *** SET EQUAL TO ONE. 001330
      C ***                                001340
      C *** "MSS" IS THE NUMBER OF SETS TO BE SAVED FOR PRINTOUT. 001350
      C ***                                001360
      C *** "IDEX1" CONTROLS PROB. CALC. AND PRINTOUT 001370
      C *** =0: NOT PERMITTED. 001380
      C *** =1: SET COMPONENT NAMES ONLY ORDERED BY SET SIZE. 001390
      C *** =2: SET COMPONENT NAMES, SET PROB., DURATION, BY SET SIZE. 001400
      C *** =3: AS ABOVE (2) WITH RESULTS FOR SET SIZE GROUPS. 001410
      C *** =4: AS ABOVE (3) WITH RESULTS FOR SYSTEM AS A WHOLE. 001420
      C *** =5: SETS WITH COMPONENT NAMES ARE OUTPUT WITH PROB., DURATIONS 001430
      C *** BY DECREASING ORDER OF PROBABILITY OF OCCURRENCE. 001440

```


C *** =6: AS ABOVE (5) WITH RESULTS FOR SYSTEM AS A WHOLE. 001450

C *** =7: ALL SETS FOUND ARE PRINTED WITHOUT CALCULATED 001460

C *** PROBABILITIES. 001470

C *** =8: ALL SETS FOUND ARE PRINTED WITH CALCULATED PROBABILITIES 001480

C *** AND MAXIMUM LIKELIHOOD DURATION TIMES. 001490

C *** IDEX1 .GT. 8 IS NOT PERMITTED. 001500

C *** 001510

C *** "IDEX2" CONTROLS WRITEOUT OF BOTTOM EVENT DESCRIPTIONS AND 001520

C *** BOTTOM EVENT IMPORTANCES. 001530

C *** ----- 001540

IDEX2 VALUE	DESCRIPTIONS	IMPORTANCES	
1	NO	NO	001560
2	NO	YES	001570
3	YES	NO	001580
4	YES	YES	001590

C *** 001600

C *** "NABS" IS A SWITCH FOR ABSORPTIONS CHECKING, ONE FOR NO 001610

C *** ABSORPTION CHECKS, TWO FOR ABSORPTIONS ONLY AMONG CUTSETS 001620

C *** OF THE SAME SIZE, THREE FOR ABSORPTION CHECKS BETWEEN SETS OF 001630

C *** DIFFERENT SIZES. 001640

C *** 001650

C *** "NDPR" .GT. ZERO DIRECTS PRINTOUT OF GATE AND BOTTOM EVENT DATA. 001660

C *** 001670

C *** "NTRANS" CONTROLS PROBABILITY AND COMPONENT INDEX DATA TRANSFER 001680

C *** TO OUTPUT FILE "TOUT", LOGICAL UNIT 7, FOR ALL ACCEPTABLE 001690

C *** VALUES OF IDEX1 EXCEPT 1 AND 7. 001700

C *** =0: NO DATA TRANSFER TAKES PLACE. 001710

C ***	=1: SET DATA IS WRITTEN ONLY WHEN THE SET PROBAB. .GE. PMIN.	001720
C ***	=2: SET DATA IS WRITTEN FOR ALL FULLY EXPANDED SETS.	001730
C ***		001740
C ***	"PMIN" IS THE MINIMUM SET PROBABILITY FOR RETAINING A SET,	001750
C ***	ONCE IT HAS BEEN FULLY EXPANDED.	001760
C ***		001770
C ***	"TIMEN" IS THE MISSION TIME FOR THE ANALYSIS.	001780
C ***		001790
C ***	"GTRACE" IS THE GATE TRACE LABEL. WHEN ENCOUNTERED DURING	001800
C ***	SETS EXPANSION, THE GATE PATH FROM TREE TOP TO GTRACE GATE	001810
C ***	LABEL IS PRINTED.	001820
C ***		001830
C ***	"CTRACE" CAUSES ONLY SETS WHICH CONTAIN COMPONENT WITH THE	001840
C ***	SAME LABEL AS CTRACE TO BE PRINTED.	001850
C ***		001860
C *****		001870
50	FORMAT(8I5/2E10.3,2(1X,A7))	001880
	WRITE(6,55) MAX,MIN,MSS,INDEX1,INDEX2,NABS,NOPR,NTRANS,PMIN,	001890
2	TIMEN	001900
	IF (GTRACK) WRITE (6,56) GTRACE	001910
	IF (CTRACK) WRITE (6,57) CTRACE	001920
55	FORMAT(1H0,*MAXIMUM SET SIZE, MAX-----*,I5/	001930
1	1H0,*MINIMUM SET SIZE, MIN-----*,I5/	001940
2	1H0,*MAXIMUM NO. OF SETS SAVED, MSS-----*,I5/	001950
3	1H0,*CALCULATION OPTION SWITCH, INDEX1---*,I5/	001960
4	1H0,*EVENT WRITE SWITCH, INDEX2---*,I5/	001970
5	1H0,*ABSORPTION CHECK SWITCH, NABS-----*,I5/	001980

```

6      1H0,*DATA PRINT SWITCH,      NOPR----*,I5/      001990
7      1H0,*DATA TRANSFER SWITCH,    NTRANS--*,I5/      002000
8      1H0,*MINIMUM PROBABILITY,      PHIN----*,IPE12.6/  002010
9      1H0,*MISSION TIME,             TIMEM---*,IPE12.6)  002020
56 FORMAT(1H0,*TRACE GATE LABEL,      GTRACE--*,A7)      002030
57 FORMAT(1H0,*SET IDENT. EVENT,      CTRACE--*,A7)      002040
      READ(5, 8) (CARD(I),I=1,10)      002050
      IF(CARD(1) .NE. END) GO TO 320      002060
C*****      002070
C *** READ GATES AND THEIR INPUTS INTO THE GATE TABLE      002080
      J=0      002090
62 CONTINUE      002100
      READ (5,40) CARD      002110
      IF (EOF(5)) 270,64,270      002120
64 CONTINUE      002130
      WRITE (6,35) CARD      002140
      IF (CARD(1) .NE. ASK) GO TO 62      002150
      IF (CARD(2) .NE. TREE) GO TO 320      002160
      J=0      002170
      IF (NOPR .LT. 1) GO TO 70      002180
      WRITE(6,15)      002190
      WRITE(6,20) TITLE      002200
      WRITE (6,67)      002210
      GO TO 70      002220
67 FORMAT(1H0,*INDEX GATE NAME TYPE*,5X,*INPUTS*,//)      002230
69 CONTINUE      002240
      MOOTST=1      002250

```

```

      J=J-1                                     002260
70 J=J+1                                     002270
      IF (J .GT. MGATES) GO TO 452             002280
      READ(5,80) GATE(1,J),BUFR(1),GATE(2,J),BUFR(2),GP(J),BUFR(3), 002290
1      NGC(J),BUFR(4),(BUFR(I+2),GATE(I,J),I=3,9), 002300
2      (BLURB(I),I=1,8)                       002310
      IF(EOF(5)) 270,75,270                   002320
75 CONTINUE                                  002330
C*****                                       002340
C *** GATE(1,J) IS THE NAME LABEL FOR THIS GATE. 002350
C *** GATE(2,J) IS THE TYPE LABEL FOR THIS GATE, EITHER "AND", 002360
C *** "OR", OR "INH".                       002370
C *** GP(J) IS THE PROBABILITY MULTIPLIER FOR THIS GATE. 002380
C *** NGC(J) IS THE NUMBER OF LOWER GATES INPUT TO THIS GATE (FIRST 002390
C *** DIGIT) AND THE NUMBER OF BOTTOM EVENTS (SECOND DIGIT) 002400
C *** INPUT TO THIS GATE.                   002410
C *** GATE(J),J=3,9 ARE GATE NAMES AND COMPONENT NAMES IN THE 002420
C *** RESPECTIVE QUANTITIES SPECIFIED BY NGC(J) INPUT TO THIS GATE. 002430
C*****                                       002440
80 FORMAT(A7,A1,A3,A1,F8.5,A1,I2,A1,7(A1,A7)/8(A10)) 002450
      IF (GP(J) .LE. 0. .OR. GP(J) .GT. 1.) GP(J)=1. 002460
      IF (NDPR .LT. 1) GO TO 902              002470
      WRITE(6,90) J,GATE(1,J),BUFR(1),GATE(2,J),BUFR(2),GP(J),BUFR(3), 002480
1      NGC(J),BUFR(4),(BUFR(I+2),GATE(I,J),I=3,9),(BLURB(K),K=1,8) 002490
90 FORMAT(1H0,I4,1X,A7,A1,A8,A1,F8.5,A1,I2,A1,7(A1,A7)/11X, 002500
2 4(2A10,1X))                                002510
902 CONTINUE                                002520

```

```

00000 DO 905 K=1,11                                002530
00001 IF (BUFR(K) .NE. BLNK1) GO TO 340              002540
00002 905 CONTINUE                                  002550
00003 IF (GATE(1,J) .EQ. END) GO TO 95              002560
00004 IF (GATE(2,J) .EQ. MOD .AND. MODTST .EQ. 1) GO TO 362 002570
00005 IF (GATE(2,J) .EQ. MOD) GO TO 69              002580
00006 MODTST=0                                       002590
00007 IF (GATE(2,J) .NE. AND .AND. GATE(2,J) .NE. OR .AND. 002600
00008 1 GATE(2,J) .NE. INH) GO TO 340                002610
00009 NG=J                                           002620
00010 NG IS THE NUMBER OF GATES COUNTED ON INPUT CARDS READ. 002630
00011 NP1=NGC(J)                                     002640
00012 CALL GATCOM(NP1,NGG,NCC)                      002650
00013 NII=NGG+NCC                                    002660
00014 IF (NII .LT. 1 .OR. NII .GT. 7) GO TO 340     002670
00015 IF (GATE(2,J) .EQ. INH .AND. NII .NE. 1) GO TO 340 002680
00016 GO TO 70                                       002690
00017 95 CONTINUE                                   002700
00018 IF (GTRACK) 96,976                             002710
00019 96 CONTINUE                                   002720
00020 GTRACK= .F.                                     002730
00021 DO 97 I=1,NG                                   002740
00022 KK=GATE(1,I)                                   002750
00023 IF (KK .NE. GTRACE) GO TO 97                  002760
00024 GTRACE=I                                       002770
00025 GTRACK = .T.                                   002780
00026 GO TO 976                                     002790

```

```

97 CONTINUE                                002800
    WRITE (6,974) GTRACE                    002810
974 FORMAT(1H0,7(1H*),* NO MATCH WAS FOUND IN THE GATE DATA FOR THE GA002820
    2TE LABEL, *,A7)                        002830
976 CONTINUE                                002840
    ERROR=0                                 002850
C*****                                     002860
C *** RUN LOOP TO CHECK THE TREE, AND GET THE TREE TOP NAME      002870
C *** IF PRESENT IN THE DATA FILE.                                002880
C*****                                     002890
    NX=0                                     002900
    DO 105 K=1,NG                           002910
    DO 100 I=1,NG                             002920
        IF (I .EQ. K .OR. NGC(I) .LT. 10) GO TO 100              002930
C *** NG IS THE TOTAL NUMBER OF GATES IN THIS TREE.              002940
    NP1=NGC(I)                                002950
    CALL GATCOM(NP1,NGG,NCC)                   002960
    N = NGG+2                                  002970
C *** N CAN HAVE VALUES FROM 2 TO 9.                               002980
    DO 98 L=3,N                               002990
C *** GATE IS DIMENSIONED: GATE(9,425).                          003000
    IF(GATE(1,K) .EQ. GATE(L,I)) GO TO 105    003010
C *** "GATE(1,K) IS CHECKED FOR A MATCH TO INPUTS OF "GATE(L,I)". 003020
C *** THE MATCH IS SATISFIED IF "GATE(1,K)" IS INPUT TO AT LEAST ONE 003030
C *** OF THE GATES, "GATE(L,I)".                                003040
    98 CONTINUE                                003050
    100 CONTINUE                               003060

```

```
      NX=NX+1                                003070
      TOPWUN=GATE(1,K)                        003080
      ERROR=ERROR+1                          003090
      NGP(NX)=GATE(1,K)                      003100
      IF (ERROR .GE. 20) GO TO 380            003110
195  CONTINUE                                003120
      IF(ERROR .GT. 1) GO TO 380              003130
      ERROR = 0                              003140
107  CONTINUE                                003150
      NGM1 = NG -1                           003160
      DO 110 I=1,NGM1                        003170
      IP1 = I+1                              003180
      KK=I                                    003190
      DO 110 J=IP1,NG                        003200
      IF(GATE(1,I) .EQ. GATE(1,J)) GO TO 420 003210
      C *** STATEMENT 420 STATES ERROR MESSAGE, GATE NAME USED MORE THAN ONCE.003220
110  CONTINUE                                003230
      WRITE (6,15)                           003240
      WRITE (6,20) TITLE                     003250
      WRITE (6,247) TOPWUN                   003260
1101 CONTINUE                                003270
      READ (5,40) CARD                       003280
      IF (EOF(5)) 270,1102,270               003290
1102 CONTINUE                                003300
      WRITE (6,35) CARD                       003310
      IF (CARD(1) .NE. ASK) GO TO 1101        003320
      IF (CARD(2) .NE. TOPS) GO TO 118        003330
```

C*****	003340
*** READ THE STARTING GATE EVENT FOR THE ANALYSIS, ASSUMING THAT	003350
C*** IT IS DIFFERENT THAN THE TOP EVENT OF THE TREE DATA.	003360
C*****	003370
READ (5,80) TOPWUN	003380
IF (EOF(5)) 270,1106,270	003390
1106 CONTINUE	003400
READ (5,80) NAME	003410
IF (EOF(5)) 270,1108,270	003420
1108 CONTINUE	003430
WRITE (6,1109) TOPWUN	003440
IF (NAME .NE. END) GO TO 320	003450
1109 FORMAT (1H0,7(2H *),* THE TOP GATE FOR THIS ANALYSIS IS REDEFINED	003460
2TO BE *,A7)	003470
C*****	003480
***GET COMPONENT DATA	003490
C*****	003500
112 CONTINUE	003510
READ (5,40) CARD	003520
IF (EOF(5)) 270,114,270	003530
114 CONTINUE	003540
WRITE (6,35) CARD	003550
IF (CARD(1) .NE. ASK) GO TO 112	003560
118 IF (CARD(2) .NE. RATES) GO TO 320	003570
NC=0	003580
IF (NOPR .LT. 1) GO TO 125	003590
WRITE(6,15)	003600


```

WRITE(6,20) TITLE                                003610
WRITE(6,180)                                       003620
WRITE(6,170)                                       003630
125 CONTINUE                                       003640
NP=NP+1                                           003650
READ(5,130) NAME,BUFR(1),COMP,BUFR(2),BUFR(3),LAMBDA,BUFR(4), 003660
1      T1,BUFR(5),T2,BUFR(6),TEEMAX,BUFR(7),CUE,    003670
2      BUFR(8),(BEV(I,NP),I=1,8)                   003680
IF (EOF(5)) 270,127,270                           003690
127 CONTINUE                                       003700
DO 128 JJ=1,6                                     003710
IF (BUFR(JJ) .NE. BLNK1) GO TO 460                 003720
128 CONTINUE                                       003730
C*****                                           003740
C *** "NAME" IS THE COMPONENT OR BOTTOM EVENT LABEL. 003750
C *** "COMP" IS THE COMPONENT OR BOTTOM EVENT TYPE.  003760
C *** "LAMBDA" IS COMPONENT FAILURE RATE PER HOUR (*10E6). 003770
C *** "T1" IS THE COMPONENT FAILURE DETECTION TIME.  003780
C *** "T2" IS THE COMPONENT REPAIR TIME.             003790
C *** "TEEMAX" IS THE MAX TIME THAT THIS COMPONENT MAY BE FAILED 003800
C *** BEFORE CAUSING THE RELATED SYSTEM FAILURE.    003810
C *** "CUE" IS THE COMPONENT UNAVAILABILITY, FRACTION OR PROBABILITY. 003820
C*****                                           003830
130 FORMAT(A7,A1,A3,A1,12X,A1,5(F10.6,A1)/8(A10)) 003840
IF (NDPR .LT. 1) GO TO 1305                        003850
WRITE(6,190) NP,NAME,COMP,LAMBDA,T1,T2,TEEMAX,CUE, 003860
2      (BEV(I,NP),I=1,8)                           003870

```

1305 CONTINUE	003880
IF(NAME .EQ. END) GO TO 140	003890
IF (COMP .EQ. HSE) GO TO 131	003900
IF (COMP .NE. DMD .AND. COMP .NE. CRC) GO TO 460	003910
131 CONTINUE	003920
IF (NC .LT. 1) GO TO 137	003930
DO 132 J=1,NC	003940
C *** CHECK FOR DUPLICATE COMPONENTS. IF FOUND, WRITE ERROR MESSAGE.	003950
IF (NAME .EQ. NAM(J)) GO TO 312	003960
132 CONTINUE	003970
DO 136 J=1,NC	003980
NP1=NGC(J)	003990
CALL GATCOM(NP1,NGG,NCC)	004000
IF (NCC .LT. 1) GO TO 136	004010
J1=NGG+3	004020
J2=NGG+NCC+2	004030
DO 135 K=J1,J2	004040
IF (NAME .EQ. GATE(K,J)) GO TO 137	004050
135 CONTINUE	004060
136 CONTINUE	004070
ERROR=1	004080
WRITE(6,310)	004090
GO TO 125	004100
137 CONTINUE	004110
DO 138 I=1,NC	004120
IF (NAME .EQ. GATE(1,I)) GO TO 316	004130
138 CONTINUE	004140

NC=NC+1	004150
IF (NC .GT. MCOMPS) GO TO 440	004160
C *** BEGIN INTERPRETATION OF COMPONENT CONSISTENT WITH THE	004170
C *** USAGE BY SUBROUTINE PROCAL.	004180
TS=T1 + T2	004190
FF=1.	004200
IF (TS .LE. 0.) GO TO 1380	004210
D=TEEMAX/TS	004220
IF (D .GT. 0. .AND. D .LT. CRITU) FF=EXP(-D)	004230
1380 CONTINUE	004240
NAM(NC)=NAME	004250
BETYP(NC)=COMP	004260
IF (COMP .EQ. HSE) GO TO 1385	004270
LAMDA(NC)=LAMBDA*FF	004280
TD(NC)=T1	004290
TR(NC)=T2	004300
Q(NC)=CUE*FF	004310
IF (CUE .LE. 0.) Q(NC)=0.	004320
IF (T1 .LE. 0. .OR. LAMBDA .LE. 0.) GO TO 1382	004330
LAMBDA=LAMBDA*1.E-6	004340
ARG=LAMBDA*TIMEM	004350
LAMDA(NC)=ARG*FF	004360
IF (ARG .GT. CRITU) GO TO 1381	004370
IF (ARG .GT. CRITL) LAMDA(NC)=(1.-EXP(-ARG))*FF	004380
GO TO 1382	004390
1381 CONTINUE	004400
LAMDA(NC)=FF	004410

```

1382 CONTINUE                                004420
      IF (CUE .GT. 0.) GO TO 125              004430
      Q(NC)=LAMDA(NC)                        004440
      IF (T2 .LE. 0. .OR. LAMDA .LE. 0.) GO TO 125 004450
      ARG=LAMDA*TS                           004460
      IF (ARG .LE. 0.) GO TO 125              004470
      Q(NC)=ARG                              004480
      IF (ARG .GT. CRITU) GO TO 1384          004490
      IF (ARG .GT. CRITL) Q(NC)=(1.-EXP(-ARG))*FF 004500
      GO TO 125                              004510
1384 CONTINUE                                004520
      Q(NC)=FF                               004530
      GO TO 125                              004540
1385 CONTINUE                                004550
      TD(NC)=0.                              004560
      TR(NC)=0.                              004570
      LAMDA(NC)=0.                           004580
      IF (LAMDA .GT. 0.) LAMDA(NC)=1.         004590
      Q(NC)=LAMDA(NC)                        004600
      GO TO 125                              004610
C *** COMPONENT DATA PROCESSED, GO TO 125 TO READ NEXT COMPONENT DATA. 004620
140 CONTINUE                                004630
141 FORMAT (1H0,7(2H *),* THE INPUT DATA LISTED *,IS, 004640
      2,* GATES AND *,IS,* BOTTOM EVENTS.*/) 004650
      IF (CTACK) 1414,1418                   004660
1414 CONTINUE                                004670
      CTACK= .F.                             004680

```

```
DO 1416 I=1,NC                                004690
IF (NAM(I) .NE. CTRACE) GO TO 1416              004700
CTRACE=I                                         004710
CTRAK= .T.                                       004720
GO TO 1418                                       004730
1416 CONTINUE                                   004740
WRITE (6,1417) CTRACE                           004750
1417 FORMAT(1H0,7(1H*),* NO MATCH WAS FOUND IN THE COMPONENT NAMES FOR 004760
2THE SET TRACE EVENT *,A7)                     004770
1418 CONTINUE                                   004780
C *** CHECK WHETHER ALL GATE INPUTS TO EACH GATE ARE PRESENT. 004790
DO 160 I=1,NG                                    004800
NP1=NGC(I)                                       004810
CALL GATCOM(NP1,NGG,NCC)                       004820
IF (NGG .LT. 1) GO TO 150                       004830
J1=NGG +2                                       004840
DO 145 J=3,J1                                   004850
DO 142 K=1,NG                                    004860
NK=K                                             004870
IF (K .EQ. I) GO TO 142                         004880
IF (GATE(1,K) .EQ. GATE(J,I)) GO TO 143        004890
142 CONTINUE                                   004900
ERROR=1                                         004910
WRITE(6,370) GATE(1,I),GATE(J,I)              004920
143 CONTINUE                                   004930
C *** SUBSTITUTE GATE INDEX FOR GATE NAME AS AN INPUT TO GATE(J,I). 004940
GATE(J,I)=NK                                    004950
```

145 CONTINUE	004960
150 CONTINUE	004970
C *** CHECK WHETHER ALL COMPONENT INPUTS TO EACH GATE ARE PRESENT.	004980
IF (NCC .LT. 1) GO TO 157	004990
J1=NGG +3	005000
J2=NGG +NCC +2	005010
DO 155 J=J1,J2	005020
DO 152 K=1,NC	005030
NK=K	005040
IF (NAM(K) .EQ. GATE(J,I)) GO TO 154	005050
152 CONTINUE	005060
ERROR=1	005070
WRITE(6,370) GATE(1,I),GATE(J,I)	005080
GO TO 155	005090
154 CONTINUE	005100
C *** SUBSTITUTE COMPONENT INDEX FOR COMPONENT NAME AS AN INPUT TO	005110
C *** GATE(J, I).	005120
GATE(J,I)=NK+MGATES	005130
155 CONTINUE	005140
157 CONTINUE	005150
NGC(I)=NGG+NCC	005160
160 CONTINUE	005170
170 FORMAT(1H0,*INDEX EVENT NAME TYPE LAMDA-10E6 DETECT. TIME	005180
2 REPAIR TIME MAX TIME FAILED UNAVAILABILITY*)	005190
180 FORMAT(1H0,*COMPONENT NAMES, TYPES, AND FAILURE/REPAIR DATA-*)	005200
190 FORMAT(1H0,I4,2X,A7,3X,A3,4X,1PE12.5,3X,0PF12.4,3X,F12.4,3X,	005210
2 1PE12.5,3X,0PF12.4/11X,4(2A10,1X))	005220

ITOP=0	005230
DO 203 I=1,NG	005240
IF (GATE(1,I) .NE. TOPWUN) GO TO 203	005250
ITOP=I	005260
203 CONTINUE	005270
IF (ITOP .EQ. 0) ERROR=1	005280
WRITE (6,141) NG,NC	005290
IF (ERROR .GT. 0 .OR. MAX .LT. 1) GO TO 500	005300
C ***	005310
C BEGIN SETS EXPANSION.	005320
C ***	005330
204 CONTINUE	005340
SET(1,1)=ITOP	005350
NSSTD=0	005360
MAXETS=0	005370
NSETS=1	005380
NSTSF=0	005390
PRM=1.	005400
N0SETS=0	005410
NLEVL(1,1)=1	005420
NVNS(1)=1	005430
DO 205 I=1,NC	005440
SENSIT(I)=0.	005450
205 CONTINUE	005460
206 CONTINUE	005470
I=1	005480
IF (NSETS .LT. 1 .AND. NSTSF .LT. 1) GO TO 400	005490

```
127800 IF (NSETS .LT. 1) GO TO 240 005500
127810 NI=NSETS 005510
127820 IF (NVNS(I) .LT. 1 .OR. SETP(I) .LT. PHIN) GO TO 227 005520
127830 C *** 227 IS THE ADDRESS TO DELETE THIS SET AND CONTINUE WITH NEXT ONE. 005530
127840 NP1=I 005540
127850 CALL DUPCEK(NP1) 005550
127860 KK=NVNS(I) 005560
127870 DO 2074 NJ=1, KK 005570
127880 JN=NJ 005580
127890 K=SET(JN, I) 005590
127900 IF (K .LE. NG) GO TO 2075 005600
127910 2074 CONTINUE 005610
127920 GO TO 229 005620
127930 C *** 229 IS THE ADDRESS TO PROCESS A FULLY EXPANDED SET. 005630
127940 2075 CONTINUE 005640
127950 C *** SET(MVNS, MSETS) WHERE SUBSCRIPTS REPRESENT (EVENTS, SETS). 005650
127960 C *** MODIFIES THIS SET TO SUBSTITUTE THE FIRST GATE INPUT 005660
127970 C *** NAME FOR THE ORIGINAL GATE NAME. 005670
127980 NADD=NGC(K)-1 005680
127990 NJ=3 005690
128000 SET(JN, I)=GATE(NJ, K) 005700
128010 LEVLS=NLEVLS(JN, I) 005710
128020 NJ=1 005720
128030 NGP(LEVLS)=GATE(NJ, K) 005730
128040 NLEVLS(JN, I)=NLEVLS(JN, I)+1 005740
128050 C *** CHECK FOR LOOPING IN THIS GATE PATH. 005750
128060 NL1=LEVLS - 1 005760
```


NL3=LEVL5	005770
DO 2077 NJ=1,NL1	005780
NL2=NJ + 1	005790
DO 2076 NK=NL2,NL3	005800
IF (NGP(NJ) .EQ. NGP(NK)) GO TO 470	005810
C *** 470 IS THE ERROR ADDRESS FOR DETECTED LOOPING.	005820
2076 CONTINUE	005830
2077 CONTINUE	005840
C *** LOOPING CHECK COMPLETED, NONE DETECTED.	005850
IF (GTRACK .AND. K .EQ. GTRACE) 2078,2080	005860
2078 CONTINUE	005870
WRITE (6,2079) GATE(1,K), (NGP(M),M=1,LEVL5)	005880
2079 FORMAT(1H0,* PATH TRACE FOR GATE *,A7/10(1X,10A7/))	005890
2080 CONTINUE	005900
KK=GATE(3,K)	005910
IF (KK .LE. NG) SETP(I) =SETP(I)*GP(KK)	005920
IF (SETP(I) .LT. PHIN) GO TO 227	005930
IF (NLEVL5(JN,I) .GE. MLEVL5 .AND. KK .LE. NG) GO TO 470	005940
IF (GATE(2,K) .EQ. AND) GO TO 220	005950
IF (GATE(2,K) .EQ. INH) GO TO 206	005960
C *** PROCESS OR GATE.	005970
2083 CONTINUE	005980
C *** CREATE ADDITIONAL SETS FROM THIS OR GATE COMPONENT.	005990
IF ((NSETS+NAOD) .LE. MSETS) GO TO 210	006000
2086 CONTINUE	006010
MAX=MAX-1	006020
IF (MIN .GT. MAX) MIN=MAX	006030

NMAX=MAX+1	006040
WRITE (6,209) MAX	006050
209 FORMAT(1H0,*MAX REDUCED TO *,I2,* TO SATISFY MSETS.*)	006060
DO 2090 IT=1,MSETS	006070
2090 SETP(IT)=1.0	006080
GO TO 204	006090
210 CONTINUE	006100
IF (NADD .LT. 1) GO TO 206	006110
NI1=NADD	006120
DO 215 I1=1,NI1	006130
C *** NADD IS THE NUMBER OF NEW SETS TO BE CREATED IN THIS LOOP.	006140
NSI=NSETS+I1	006150
NVNS(NSI)=NVNS(I)	006160
NJ1=NVNS(I)	006170
SETP(NSI)=SETP(I)	006180
DO 214 J1=1,NJ1	006190
NLEVL(J1,NSI)=NLEVL(J1,I)	006200
IF (J1 .EQ. JN) GO TO 212	006210
SET(J1,NSI)=SET(J1,I)	006220
GO TO 214	006230
212 CONTINUE	006240
IT=I1+3	006250
KK=GATE(IT,K)	006260
SET(J1,NSI)=KK	006270
IF (KK .LE. NG .AND. KK .GT. 0) SETP(NSI)=SETP(I)*GP(KK)	006280
214 CONTINUE	006290
215 CONTINUE	006300

NSETS=NSETS+NAOD	006310
GO TO 206	006320
220 CONTINUE	006330
C *** BEGIN PROCESSING AND GATE.	006340
C *** SET INDEX IS I.	006350
NP1=I	006360
NP2=K	006370
LEVL=LEVL + 1	006380
CALL SHRINK(NP1,NP2)	006390
IF (NVNS(I) .GT. 0 .AND. NVNS(I) .LE. NMAX) GO TO 206	006400
227 CONTINUE	006410
NSTSF=NSTSF + 1	006420
228 CONTINUE	006430
C *** DELETE THIS SET AND BEGIN PROCESSING SET(NSETS)	006440
NP1=1	006450
IF (NSETS .GT. MAXETS) MAXETS=NSETS	006460
CALL SOLITR(NP1)	006470
GO TO 206	006480
229 CONTINUE	006490
C *** BEGIN PROCESSING A FULLY EXPANDED SET.	006500
I=1	006510
NV=NVNS(I)	006520
IF (NV .LT. MIN .OR. NV .GT. MAX) GO TO 228	006530
C *** CALCULATE SET PROBABILITY FOR THIS SET IN PREPARATION TO.	006540
C *** STORING IT IN ARRAY OF RESULTS.	006550
DO 230 J=1,NV	006560
K=SET(J,I)-MGATES	006570

SET(J,I)=K	006580
230 CONTINUE	006590
IF (CTACK) 2302,232	006600
2302 CONTINUE	006610
DO 2304 J=1,NV	006620
K=SET(J,I)	006630
IF (K .EQ. CTRACE) GO TO 2306	006640
2304 CONTINUE	006650
GO TO 232	006660
2306 CONTINUE	006670
WRITE (6,2307) NAM(K)	006680
2307 FORMAT(1H0,* SET CONTAINING THE BOTTOM EVENT *,A7)	006690
IF (IDEX1 .EQ. 1 .OR. IDEX1 .EQ. 7) GO TO 231	006700
NP1=1	006710
CALL PROCAL(NP1,PROB)	006720
WRITE(6,2308) PROB	006730
2308 FORMAT(1H+,43X,*SET PROBABILITY IS *,1PE16.6)	006740
231 CONTINUE	006750
IF (IDEX2 .LT. 3) 2328,2325	006760
232 CONTINUE	006770
IF (IDEX1 .LT. 2) GO TO 2338	006780
IF (IDEX1 .NE. 7) GO TO 2330	006790
WRITE(6,274)	006800
IF (IDEX2 .LT. 3) GO TO 2328	006810
2325 CONTINUE	006820
DO 2327 J=1,NV	006830
K=SET(J,I)	006840

```
WRITE(6,253) J,NAM(K),(BEV(JJ,K),JJ=1,8) 006850
2327 CONTINUE 006860
GO TO 232 006870
2328 CONTINUE 006880
DO 2329 J=1,NV 006890
K=SET(J,I) 006900
WRITE(6,253) J,NAM(K) 006910
2329 CONTINUE 006920
GO TO 227 006930
2330 CONTINUE 006940
NP1=I 006950
CALL PROCAL(NP1,PROB) 006960
SETP(I)=PROB 006970
IF (IDEX1 .NE. 8) GO TO 2334 006980
BSUM=0. 006990
TAUMIN=1.E+50 007000
WRITE(6,274) 007010
DO 2333 J=1,NV 007020
K=SET(J,I) 007030
TEMP=TD(K)+TR(K) 007040
IF (TEMP .GE. TAUMIN .OR. TEMP .LE. 0.) GO TO 2331 007050
BSUM=BSUM + 1./TEMP 007060
2331 CONTINUE 007070
IF (IDEX2 .LT. 3) GO TO 2332 007080
WRITE(6,253) J,NAM(K),(BEV(JJ,K),JJ=1,8) 007090
GO TO 2333 007100
2332 CONTINUE 007110
```

WRITE(6,253) J,NAP(K)	007120
2333 CONTINUE	007130
IF (BSUM .LE. 0.) BSUM=1./TAUMIN	007140
SETDUR=1./BSUM	007150
SLSUM=SLSUM+SETP(I)	007160
SPSUM=SPSUM+SETDUR*SETP(I)	007170
WRITE(6,258) SETP(I),SETDUR	007180
IF (INTRANS .LT. 1) GO TO 227	007190
IF (INTRANS .LT. 2 .AND. PROB .LT. PMIN) GO TO 227	007200
WRITE(7,2465) NV,SETP(I),(SET(J,I),J=1,NV)	007210
GO TO 227	007220
2334 CONTINUE	007230
C *** ADD SET PROBABILITY TO EACH BOTTOM EVENT IMPORTANCE FOR THE SET.	007240
NV=NVNS(I)	007250
DO 2336 J=1,NV	007260
K=SET(J,I)	007270
SENSIT(K)=SENSIT(K) + SETP(I)	007280
2336 CONTINUE	007290
IF (INTRANS .LT. 2 .AND. PROB .LT. PMIN) GO TO 227	007300
IF (INTRANS .LT. 1) GO TO 2337	007310
WRITE (7,2465) NV,SETP(I),(SET(J,I),J=1,NV)	007320
2337 CONTINUE	007330
IF (PROB .LT. PMIN) GO TO 227	007340
2338 CONTINUE	007350
IF (NSSTD .GE. MSS .AND. IDEX1 .LT. 2) GO TO 2086	007360
IF (NSSTD .GE. MSS) GO TO 2343	007370
NSSTD=NSSTD+1	007380

```
NSTR=NSSTD 007390
2339 CONTINUE 007400
      DO 2340 J=1,NV 007410
        SSET(J,NSTR)=SET(J,I) 007420
2340 CONTINUE 007430
      NVNSS(NSTR)=NVNS(I) 007440
      SETPS(NSTR)=SETP(I) 007450
      IF (IDEX1 .LT. 2) GO TO 227 007460
      IF (SETP(I) .LT. PRM) PRM=SETP(I) 007470
      GO TO 227 007480
2343 CONTINUE 007490
      IF (PROB .LT. PRM) GO TO 227 007500
      C *** SEARCH "SSET" ARRAY FOR SET WITH LOWEST PROBABILITY. 007510
      PRM=1. 007520
      NJ=MSS 007530
      DO 2345 J=1,NJ 007540
        PROB=SETPS(J) 007550
        IF (PROB .GE. PRM) GO TO 2345 007560
        PRM=SETPS(J) 007570
        NSTR=J 007580
      C *** NSTR IS THE INDEX FOR THE MINIMUM PROBABILITY ELEMENT OF "SSET". 007590
2345 CONTINUE 007600
      GO TO 2339 007610
240 CONTINUE 007620
      IF (SLSUM .LE. 0. .OR. SPSUM .LE. 0. .OR. IDEX1 .NE. 8) GO TO 241 007630
      SYSTAU=SPSUM/SLSUM 007640
      WRITE (6,262) SLSUM,SYSTAU 007650
```

241 CONTINUE	007660
C*****	007670
C *** BEGIN OUTPUT OF STORED SETS AND DATA.	007680
C*****	007690
245 FORMAT(1H0,*THIS TREE HAS *,I4,* CUTSETS CONTAINING */	007700
2 * FROM *,I2,* TO *,I2,* EVENTS*)	007710
2455 FORMAT (1H ,* AND PROBABILITY OF *,1PE13.6,* OR GREATER.*)	007720
246 FORMAT(1H0,*THE MAX NUMBER OF SETS STORED INTERNALLY WAS *,I4)	007730
2465 FORMAT(1H ,I6,E14.8,10I5)	007740
WRITE(6,274)	007750
WRITE(6,246) MAXETS	007760
WRITE(6,274)	007770
WRITE(6,245) NSSTD,MIN,MAX	007780
IF (IDEX1 .GT. 1 .AND. IDEX1 .LT. 7) WRITE(6,2455) PRM	007790
WRITE(6,274)	007800
WRITE(6,2472) NSTSF	007810
WRITE(6,274)	007820
WRITE(6,247) TOPHUN	007830
WRITE(6,274)	007840
247 FORMAT (1H0,*THE TOP GATE OF THIS TREE IS *,A8)	007850
2472 FORMAT(1H0,*THE TOTAL NUMBER OF SETS GENERATED WAS *,I9)	007860
IF (NABS .LT. 2 .OR. NSSTD .LT. 2) GO TO 2500	007870
C *** CHECK FOR SET ABSORPTIONS.	007880
NI=NSSTD-1	007890
NJ=NSSTD	007900
DO 2486 I=1,NI	007910
NNI=NVNSS(I)	007920

IF (NNI .LT. 1) GO TO 2486	007930
NP1=I+1	007940
DO 2484 J=NP1,NJ	007950
NNJ=NVNSS(J)	007960
IF (NNJ .LT. 1) GO TO 2484	007970
NNI=NVNSS(I)	007980
IF (NABS .LT. 3 .AND. NNJ .NE. NNI) GO TO 2484	007990
KI=J	008000
KJ=I	008010
NNI=NVNSS(KI)	008020
NNJ=NVNSS(KJ)	008030
IF (NNI .LE. NNJ) GO TO 2474	008040
C *** SPECIFY INDICES SO THAT NVNSS(KI) .LE. NVNSS(KJ).	008050
KI=I	008060
KJ=J	008070
2474 CONTINUE	008080
C *** CHECK WHETHER SSET(KJ) IS ABSORBED BY SSET(KI).	008090
NNI=NVNSS(KI)	008100
NNJ=NVNSS(KJ)	008110
DO 2480 II=1,NNI	008120
DO 2478 JJ=1,NNJ	008130
IF (SSET(JJ,KJ) .EQ. SSET(II,KI)) GO TO 2480	008140
2478 CONTINUE	008150
C *** MEMBER OF SSET(KI) NOT DUPLICATED IN SSET(KJ), GO TO 2484.	008160
GO TO 2484	008170
2480 CONTINUE	008180
C *** SSET(KJ) IS ABSORBED BY SSET(KI).	008190

PROB=SETPS(KJ)	008200
C *** THE STATEMENT JUST BELOW PRESERVES THE HIGHEST PROBABILITY.	008210
IF (SETPS(KJ) .GT. SETPS(KI)) SETPS(KI)=SETPS(KJ)	008220
NV=NVNSS(KJ)	008230
DO 2482 II=1,NV	008240
JJ=SSET(II,KJ)	008250
SENSIT(JJ)=SENSIT(JJ)-PROB	008260
2482 CONTINUE	008270
NVNSS(KJ)=0	008280
IF (KJ .EQ. I) GO TO 2486	008290
2484 CONTINUE	008300
2486 CONTINUE	008310
NI=NSSTD	008320
DO 2495 I=1,NI	008330
2491 CONTINUE	008340
IF (I .GT. NSSTD) GO TO 2498	008350
NV=NVNSS(I)	008360
IF (NV .GT. 0) GO TO 2495	008370
NJ=NSSTD	008380
NV=NVNSS(NJ)	008390
SETPS(I)=SETPS(NJ)	008400
NVNSS(I)=NVNSS(NJ)	008410
DO 2493 J=1,NV	008420
SSET(J,I)=SSET(J,NJ)	008430
2493 CONTINUE	008440
NSSTD=NSSTD-1	008450
GO TO 2491	008460

2495	CONTINUE	008470
2498	CONTINUE	008480
	WRITE(6,2499) NSSTD	008490
	WRITE (6,274)	008500
2499	FORMAT(1H0,*THERE ARE *,I5,* SETS REMAINING AFTER ABSORPTION CHECK	008510
	2ING.*)	008520
2500	CONTINUE	008530
	GO TO (2510,2501,2501,2501,2501,2501,500,265),INDEX1	008540
2501	CONTINUE	008550
C ***	BEGIN ORDERING SETS BY DECREASING SET OCCURRENCE PROBABILITY.	008560
	IF (NSSTD .LT. 2) GO TO 2509	008570
	NI=NSSTD-1	008580
	DO 2508 I=1,NI	008590
	NSM=I	008600
	NJ=I+1	008610
	PHAX=SETPS(I)	008620
	DO 2503 J=NJ,NSSTD	008630
	PROB=SETPS(J)	008640
	IF (PROB .LE. PHAX) GO TO 2503	008650
	PHAX=SETPS(J)	008660
	NSM=J	008670
2503	CONTINUE	008680
	IF (NSM .EQ. I) GO TO 2508	008690
	NVNSS(MSSP1)=NVNSS(I)	008700
	SETPS(MSSP1)=SETPS(I)	008710
	NV=NVNSS(I)	008720
	DO 2504 K=1,NV	008730

2504 SSET(K,MSSP1)=SSET(K,I)	008740
2505 SETPS(I)=SETPS(NSM)	008750
2506 NVNSS(I)=NVNSS(NSM)	008760
2507 NV=NVNSS(NSM)	008770
DO 2506 K=1,NV	008780
2506 SSET(K,I)=SSET(K,NSM)	008790
2507 SETPS(NSM)=SETPS(MSSP1)	008800
2508 NVNSS(NSM)=NVNSS(MSSP1)	008810
2509 NV=NVNSS(MSSP1)	008820
DO 2507 K=1,NV	008830
2507 SSET(K,NSM)=SSET(K,MSSP1)	008840
2508 CONTINUE	008850
2509 CONTINUE	008860
C *** SET ORDERING COMPLETED.	008870
GO TO (500,2510,2510,2510,2641,2641,500,500),IDEX1	008880
2510 CONTINUE	008890
C *** BEGIN OUTPUT OF SETS BY SET SIZE.	008900
NI=0	008910
SLSUM=0.	008920
SPSUM=0.	008930
DO 260 I=MIN,MAX	008940
GLSUM=0.	008950
GPSUM=0.	008960
WRITE (6,273)	008970
WRITE (6,250) I	008980
WRITE (6,274)	008990
250 FORMAT (1H0,*SETS WITH *,I2,* COMPONENTS---*)	009000

NJ=0	009010
DO 259 J=1, NSSTO	009020
NV=NVNSS(J)	009030
IF (NV .NE. I) GO TO 259	009040
NJ=NJ+1	009050
NI=NI+1	009060
WRITE(6,251) NI,NJ	009070
251 FORMAT(1H0,*CUT SET NO. *,I4,* NUMBER *,I4,* OF THIS GROUP.*)	009080
BSUM=0.	009090
TAUMIN=1.E+50	009100
DO 2540 JL=1,NV	009110
K=SSET(JL,J)	009120
TEMP=TD(K)+TR(K)	009130
IF (TEMP .GE. TAUMIN .OR. TEMP .LE. 0.) GO TO 2534	009140
TAUMIN=TEMP	009150
BSUM=BSUM+1./TEMP	009160
2534 CONTINUE	009170
IF (IDEX2 .LT. 3) GO TO 2538	009180
WRITE (6,253) JL,NAM(K),(BEV(JJ,K),JJ=1,8)	009190
GO TO 2540	009200
2538 CONTINUE	009210
WRITE (6,253) JL,NAM(K)	009220
2540 CONTINUE	009230
IF (BSUM .LE. 0.) BSUM=1./TAUMIN	009240
SETDUR=1./BSUM	009250
252 FORMAT (1H ,(1X,7(2X,A7)/))	009260
253 FORMAT(1H ,I3,1X,A7,3X,4(2A10,1X))	009270

IF (IDEX1 .LT. 2) GO TO 259	009280
255 FORMAT (1H0, *SETS IN ORDER OF DECREASING PROBABILITY.*)	009290
WRITE (6,258) SETPS(J),SETDUR	009300
258 FORMAT(1H , *OUTSET PROBABILITY IS *,1PE16.6,	009310
2 * . EFFECTIVE DURATION TIME IS *,1PE16.6)	009320
GO TO (259,259,2586,2584,2641,2641,500,500),IDEX1	009330
2584 CONTINUE	009340
SLSUM=SLSUM+SETPS(J)	009350
SPSUM=SPSUM+SETDUR*SETPS(J)	009360
2586 CONTINUE	009370
GLSUM=GLSUM+SETPS(J)	009380
GPSUM=GPSUM+SETDUR*SETPS(J)	009390
259 CONTINUE	009400
IF (IDEX1 .LT. 2) GO TO 260	009410
IF (GLSUM .LE. 0. .OR. GPSUM .LE. 0.) GO TO 260	009420
GRTAU=GPSUM/GLSUM	009430
WRITE(6,263) GLSUM,GRTAU	009440
260 CONTINUE	009450
IF (IDEX1 .LT. 4) GO TO 265	009460
2605 CONTINUE	009470
IF (SLSUM .LE. 0. .OR. SPSUM .LE. 0.) GO TO 265	009480
SYSTAU=SPSUM/SLSUM	009490
WRITE (6,262) SLSUM,SYSTAU	009500
GO TO 265	009510
262 FORMAT(1H1,7(2H *)//5X, *THE PROBABILITY OF SYSTEM FAILURE DUE TO T	009520
2HE LISTED OUTSETS IS *,1PE16.6/2X,	009530
3 *SYSTEM EFFECTIVE FAILURE DURATION TIME IS *,1PE16.6,	009540

THIS PAGE
WAS INTENTIONALLY
LEFT BLANK

BSUM=BSUM+1./TEMP	009820
2644 CONTINUE	009830
IF (IDEX2 .LT. 3) GO TO 2648	009840
WRITE (6,253) J,NAM(K),(BEV(JJ,K),JJ=1,8)	009850
GO TO 2650	009860
2648 CONTINUE	009870
WRITE(6,253) J,NAM(K)	009880
2650 CONTINUE	009890
IF (BSUM .LE. 0.) BSUM=1./TAUMIN	009900
SETDUR=1./BSUM	009910
WRITE (6,258) SETPS(I),SETDUR	009920
IF (IDEX1 .LT. 6) GO TO 2655	009930
SLSUM=SLSUM+SETPS(I)	009940
SPSUM=SPSUM+SETPS(I)*SETDUR	009950
2655 CONTINUE	009960
IF (IDEX1 .EQ. 6) GO TO 2605	009970
265 CONTINUE	009980
IF (IDEX1 .LT. 2 .OR. IDEX1 .EQ. 7) GO TO 500	009990
IF (IDEX2 .NE. 2 .AND. IDEX2 .NE. 4) GO TO 500	010000
WRITE(6,15)	010010
WRITE(6,20) TITLE	010020
WRITE(6,2775)	010030
2775 FORMAT(1H0,7(2H*),*COMPONENT IMPORTANCE FACTORS, PROBABILITIES, A010040	
2ND IMPORTANCES.*//3X,*COMPONENT*,3X,*IMPORTANCE*,4X,*COMPONENT*,	010050
3 6X,*COMPONENT*/3X,*LABEL*,7X,*FACTOR*,8X,*PROBABILITY*,4X,	010060
4 *IMPORTANCE*//)	010070
2776 FORMAT(3X,A7,4X,1PE13.6,1X,1PE13.6,2X,1PE13.6)	010080

2777	CONTINUE	010090
2778	FORMAT(5X,4(2A10,1X)/)	010100
	PMAX=0.	010110
	NJ=NC	010120
	NTEST=0	010130
	DO 2779 J=1,NJ	010140
	IF (PMAX .GE. SENSIT(J)) GO TO 2779	010150
	NTEST=1	010160
	IMAX=J	010170
	PMAX=SENSIT(J)	010180
2779	CONTINUE	010190
	IF (NTEST .LT. 1) GO TO 2781	010200
	CP=LAMDA(IMAX) + Q(IMAX)	010210
	IF (CP .GT. 1.) CP=1.	010220
	FACTOR=0.	010230
	IF (CP .GT. 0.) FACTOR=SENSIT(IMAX)/CP	010240
	WRITE(6,2776) NAM(IMAX),FACTOR,CP,SENSIT(IMAX)	010250
	IF (IDEX2 .LT. 3) GO TO 2780	010260
	WRITE(6,2778) (BEV(L,IMAX),L=1,8)	010270
2780	CONTINUE	010280
	SENSIT(IMAX)=0.	010290
	GO TO 2777	010300
2781	CONTINUE	010310
	GO TO 500	010320
	C*****	010330
	C *** ERROR MESSAGES	010340
	C*****	010350

270 ERROR = 1	010360
WRITE(6, 274)	010370
WRITE(6, 276)	010380
WRITE(6, 274)	010390
273 FORMAT(1H1,7(2H *))	010400
274 FORMAT(1H0,7(2H *))	010410
GO TO 500	010420
276 FORMAT(1H0,7(1H*),*READ THROUGH END OF FILE WAS ATTEMPTED.*)	010430
310 FORMAT(1H0,7(1H*),*THE ABOVE COMPONENT IS NOT INPUT TO ANY GATE.*)	010440
312 ERROR=1	010450
WRITE (6,314) NAME	010460
314 FORMAT(1H0,7(1H*),*THIS COMPONENT NAME IS DUPLICATED.*,2X,A7)	010470
GO TO 125	010480
316 ERROR=1	010490
WRITE(6,318) NAME	010500
318 FORMAT(1H0,7(1H*),* THIS COMPONENT NAME, *,A7,* DUPLICATES A GATE	010510
2NAME, IGNORED.*)	010520
GO TO 125	010530
320 ERROR = 1	010540
WRITE(6, 274)	010550
WRITE(6, 330)	010560
330 FORMAT(1H0,7(1H*),* A CONTROL CARD WAS OMITTED OR MISPUNCHED.*)	010570
WRITE(6, 35) CARD	010580
GO TO 500	010590
340 ERROR = 1	010600
WRITE(6,350)	010610
350 FORMAT(1H0,7(1H*),*THE ABOVE GATE CARD HAS BEEN MISPUNCHED.*)	010620

```
GO TO 69 010630
362 CONTINUE 010640
WRITE (6,364) 010650
364 FORMAT (1H0,7(1H*),* TWO CONSECUTIVE MOD SYMBOLS WERE ENCOUNTERED 010660
2.*) 010670
ERROR=1 010680
GO TO 69 010690
370 FORMAT(1H0,7(1H*),* THE GATE, *,A7,* HAS AN INPUT, *,A7, 010700
2 * FOR WHICH NO DATA WAS SUPPLIED.*) 010710
380 WRITE(6, 274) 010720
WRITE (6,390) ERROR,(NGP(I),I=1,NX) 010730
ERROR=1 010740
WRITE(6, 274) 010750
390 FORMAT(1H0,7(1H*),I3,* GATES HAVE BEEN DESCRIBED WHICH ARE NOT INP 010760
2UT TO ANY OTHER GATES.*,(/1X,10(A7,1X))) 010770
GO TO 107 010780
400 ERROR=1 010790
WRITE (6,401) 010800
401 FORMAT(1H0,7(1H*),* ZERO SETS FOUND FOR THIS MAX SET SIZE.*) 010810
GO TO 500 010820
420 ERROR=1 010830
IF (KK .LT. 1 .OR. KK .GT. NG) GO TO 500 010840
WRITE(6,430) GATE(1,KK) 010850
430 FORMAT(1H0,7(1H*),* THE GATE NAME *,A7,* APPEARS MORE THAN ONCE.*) 010860
GO TO 500 010870
440 ERROR = 1 010880
WRITE(6, 274) 010890
```

WRITE(6, 450) NP, MCOMPS	010900
450 FORMAT(1H0,7(1H*),* THE NUMBER OF COMPONENTS IN THE INPUT DATA IS	010910
2*,I6,*. THIS EXCEEDS THE *,I4,* ALLOWED.*)	010920
WRITE(6, 274)	010930
GO TO 500	010940
452 ERROR=1	010950
WRITE (6,454)	010960
454 FORMAT(1H0,7(1H*),* ATTEMPTED TO READ MORE THAN THE ALLOWED NUMBER	010970
2 OF GATE CARDS.*)	010980
GO TO 500	010990
460 ERROR=1	011000
WRITE (6,465)	011010
465 FORMAT(1H0,7(1H*),* ERRONEOUS DATA FOR ABOVE COMPONENT, IGNORED.*)	011020
GO TO 125	011030
470 ERROR=1	011040
WRITE (6,475) (NGP(M),M=1,LEVL5)	011050
GO TO 500	011060
475 FORMAT(1H1,7(1H*),* A LOOP IN THE DATA IS SUSPECTED.*//10X,	011070
2 * THE FOLLOWING GATES INCLUDE THE LOOP PATH.*//	011080
3 100(10X,A10//)	011090
500 IF (ERROR .LT. 1) GO TO 900	011100
510 FORMAT (1H0,7(1H*),* ERROR CONDITION ENCOUNTERED.*//	011110
WRITE (6,510)	011120
900 CONTINUE	011130
NV=0	011140
IF (NTRANS .GT. 0) WRITE (7,2465) NV	011150
STOP	011160

THIS PAGE
WAS INTENTIONALLY
LEFT BLANK

SUBROUTINE SDLITR(I)	011180
REAL GP(425),SETP(100),TD(175),TR(175),Q(175),LAMDA(175)	011190
INTEGER NGC(425),GATE(9,425),SET(10,100)	011200
2 ,NVNS(100)	011210
4 ,NLEVL(10,50),NGP(50)	011220
COMMON GATE,SET,GP,SETP,NGC,NGP,NLEVL,TD,TR,Q,LAMDA,	011230
2 NVNS,NMAX,NMAX,MIN,MCOMPS,NGATES,NG,NC,NSETS,LEVL	011240
IF (I .GT. NSETS) GO TO 90	011250
IF (I .LT. 1) GO TO 100	011260
IF (I .EQ. NSETS) GO TO 55	011270
SETP(I)=SETP(NSETS)	011280
NVNS(I)=NVNS(NSETS)	011290
NJ=NVNS(NSETS)	011300
DO 40 J=1,NJ	011310
NLEVL(J,I)=NLEVL(J,NSETS)	011320
NLEVL(J,NSETS)=0	011330
SET(J,I)=SET(J,NSETS)	011340
SET(J,NSETS)=0	011350
40 CONTINUE	011360
55 CONTINUE	011370
SETP(NSETS)=1.	011380
NVNS(NSETS)=0	011390
NSETS=NSETS-1	011400
RETURN	011410
90 CONTINUE	011420
WRITE (6,95)	011430
95 FORMAT(1H0,*SDLITR ERROR, I GREATER THAN NSETS.*)	011440

RETURN	011450
100 CONTINUE	011460
WRITE (6,110).	011470
110 FORMAT(1H0,*SOLITR ERROR, I LESS THAN ONE.*)	011480
RETURN	011490
END	011500

SUBROUTINE DUPCEK(NST)	011510
REAL GP(425),SETP(100),TO(175),TR(175),Q(175),LAMDA(175)	011520
INTEGER NGC(425),GATE(9,425),SET(10,100)	011530
2 NVNS(100)	011540
4 NLEVL(10,50),NGP(50)	011550
COMMON GATE,SET,GP,SETP,NGC,NGP,NLEVL,TO,TR,Q,LAMDA,	011560
2 NVNS,MAX,NMAX,MIN,MCOMPS,MGATES,NG,NC,NSETS,LEVL	011570
NV=NVNS(NST)	011580
IF (NV .LT. 2) GO TO 60	011590
NI=NV-1	011600
KK=MGATES+NC	011610
DO 50 I=1,NI	011620
10 CONTINUE	011630
NVM1=NV-1	011640
IF (I .GT. NVM1) GO TO 55	011650
K=SET(I,NST)	011660
IDEX=I	011670
IF (K .LT. 1 .OR. K .GT. KK) GO TO 33	011680
IP1=I+1	011690
DO 30 J=IP1,NV	011700
KJ=SET(J,NST)	011710
IDEX=J	011720
IF (KJ .LT. 1 .OR. KJ .GT. KK) GO TO 33	011730
IF (KJ .EQ. K) GO TO 35	011740
30 CONTINUE	011750
GO TO 50	011760
33 CONTINUE	011770

WRITE (6,34) NST, IDEX, NVNS(NST), (SET(J,NST), J=1, NV)	011780
34 FORMAT (1H0, *BAD INDEX FOUND IN DUPCEK*, (2X, 8I6/))	011790
35 CONTINUE	011800
K=SET(IDEX, NST)	011810
IF (NV .EQ. IDEX) GO TO 38	011820
IF (NV .LT. IDEX) GO TO 45	011830
NJ=NV-1	011840
DO 37 J=IDEX, NJ	011850
JP1=J+1	011860
SET(J, NST)=SET(JP1, NST)	011870
NLEVL(J, NST)=NLEVL(JP1, NST)	011880
37 CONTINUE	011890
38 CONTINUE	011900
SET(NV, NST)=0	011910
NLEVL(NV, NST)=0	011920
NV=NV-1	011930
IF (K .LT. 1 .OR. K .GT. KK) GO TO 45	011940
IF (K .GT. NG) GO TO 10	011950
GPROB=GP(K)	011960
IF (GPROB .LE. 0. .OR. GPROB .GT. 1.) GO TO 10	011970
SETP(NST)=SETP(NST)/GPROB	011980
GO TO 10	011990
45 CONTINUE	012000
WRITE (6,46) NST, SETP(NST), K, (SET(J, NST), J=1, NV)	012010
46 FORMAT(1H0, 7(1H*), * DUPCEK ERROR *, I4, 1PE12.5, (6(2X, I4)/))	012020
GO TO 10	012030
50 CONTINUE	012040

55 CONTINUE

012050

NVNS(NST)=NV

012060

60 CONTINUE

012070

RETURN

012080

END

012090

SUBROUTINE SHRINK(NST,NGT)	012100
REAL GP(425),SETP(100),TD(175),TR(175),Q(175),LAMDA(175)	012110
INTEGER NGC(425),GATE(9,425),SET(10,100)	012120
2 ,NVNS(100),ADARA(6)	012130
4 ,NLEVL(10,50),NGP(50)	012140
COMMON GATE,SET,GP,SETP,NGC,NGP,NLEVL,TD,TR,Q,LAMDA,	012150
2 NVNS,MAX,NMAX,MIN,MCOMPS,MGATES,NG,NC,NSETS,LEVL	012160
NI=NGC(NGT)-1	012170
NVG=NVNS(NST)	012180
IF (NI .LT. 1 .OR. NVG .LT. 1) GO TO 90	012190
DO 10 I=1,6	012200
ADARA(I)=0	012210
10 CONTINUE	012220
IF (NI .GT. 6) GO TO 90	012230
DO 20 I=1,NI	012240
COPY GATE INPUTS, NOT INCLUDING THE FIRST, TO ADARA(I)	012250
IP3=I+3	012260
ADARA(I)=GATE(IP3,NGT)	012270
20 CONTINUE	012280
NVG=NVNS(NST)	012290
IF (NVG .LT. 2) GO TO 25	012300
CALL DUPCEK(NST)	012310
NVG=NVNS(NST)	012320
25 CONTINUE	012330
IF (NVG .LT. 1) GO TO 90	012340
DO 50 I=1,NI	012350
NJ=NVG	012360

K=ADARA(I)	012370
IF (K .LT. 1) GO TO 50	012380
DO 40 J=1,NJ	012390
IF (SET(J,NST) .EQ. K) GO TO 50	012400
CONTINUE	012410
NVG=NVG+1	012420
IF (NVG .GT. NMAX) GO TO 80	012430
IF (K .LE. NG .AND. K .GT. 0) SETP(NST)=SETP(NST)*GP(K)	012440
SET(NVG,NST)=K	012450
NLEVELS(NVG,NST)=LEVELS	012460
50 CONTINUE	012470
NVNS(NST)=NVG	012480
GO TO 90	012490
80 CONTINUE	012500
NVNS(NST)=0	012510
90 CONTINUE	012520
RETURN	012530
END	012540

SUBROUTINE GATCOM(NGC,NGG,NCC)

012550

NGG=NGC/10

012560

NCC=NGC-10*NGG

012570

RETURN

012580

END

012590

```
SUBROUTINE PROCAL(NST,PROB)                                012600
REAL GP(425),SETP(100),TO(175),TR(175),Q(175),LAMDA(175) 012610
INTEGER NGC(425),GATE(9,425),SET(10,100)                 012620
2",NVNS(100)                                              012630
4 ,NLEVL(10,50),NGP(50)                                  012640
COMMON GATE,SET,GP,SETP,NGC,NGP,NLEVL,TO,TR,Q,LAMDA,      012650
2 NVNS,MAX,NMAX,MIN,PCOMPS,MGATES,NG,NC,NSETS,LEVL
FUNM=0.                                                    012670
NI=NVNS(NST)                                              012680
DO 20 I=1,NI                                              012690
KI=SET(I,NST)                                             012700
TREP= TO(KI)+TR(KI)                                       012710
17 CONTINUE                                              012720
IF (TR(KI) .LE. 0.) FUNM=FUNM + 1.                       012730
20 CONTINUE                                              012740
IF (FUNM .LT. 1.) GO TO 23                               012750
FUNMP1=1./(FUNM + 1.)                                     012760
FUNM=1./FUNM                                              012770
GO TO 25                                                  012780
23 CONTINUE                                              012790
FUNM=1.                                                  012800
FUNMP1=1.                                                 012810
25 CONTINUE                                              012820
TRMSUM=0.                                                 012830
NJ=NVNS(NST)                                             012840
DO 50 I=1,NI                                             012850
KI=SET(I,NST)                                             012860
```

IF (LAMDA(KI) .LE. 0.) GO TO 50	012870
TERM=FUNMP1	012880
IF (TR(KI) .LE. 0.) TERM=FUNM	012890
DO 40 J=1,NJ	012900
KJ=SET(J,NST)	012910
IF (J .EQ. I) GO TO 35	012920
TERM=TERM*Q(KJ)	012930
GO TO 40	012940
35 CONTINUE	012950
TERM=TERM*LAMDA(KJ)	012960
40 CONTINUE	012970
TRMSUM=TRMSUM+TERM	012980
50 CONTINUE	012990
PROB=SETP(NST)*TRMSUM	013000
RETURN	013010
END	013020

(This page has been intentionally left blank)

APPENDIX F

SOURCE LANGUAGE LISTING FOR PROGRAM BRANCH

The following pages are a source language listing for BRANCH, written in Fortran IV for the CSC Richland Cyber 70 Computer. It uses the same data which is input to ALLCUTS.

PROGRAM BRANCH (DATREE,REFILE,TAPE5=DATREE,TAPE4=REFILE)

C

C

PROGRAM BRANCH IS ONE OF THREE PROGRAMS WRITTEN TO PROVIDE
ANALYSIS OF FAULT TREES.

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

THE PURPOSE OF BRANCH IS TO PROVIDE A CROSS-REFERENCE MAP
OF ALL GATES AND EVENTS IN A FAULT TREE.
IN THE CROSS-REFERENCE MAP ALL THE INPUTS TO EACH GATE ARE
LISTED AS WELL AS ALL THE GATES INTO WHICH EACH GATE OR BOTTOM
EVENT IS INPUT . BOTTOM EVENTS ARE DISTINGUISHED FROM GATES
BY A LEADING ASTERISK.

VERSION 1.0

WRITTEN BY: DAN GRIFFING, ATLANTIC RICHFIELD HANFORD COMPANY
RICHLAND, WA., JANUARY 1975.

LOGICAL FLOW

1.0 INPUT TREE DATA FROM FILE DATREE (LOGICAL UNIT 5)
AND STORE THE PERTINANT DATA INTO VARIOUS ARRAYS.

1.1 TEST FOR BEGINNING OF TREE DATA, BYPASSING
RECORDS UNTIL AN *TREE RECORD IS REACHED.

1.2 STORE THE GATES WHICH ARE INPUT TO A GIVEN GATE
IN ARRAY IGATE (M,N) WHERE M GOES FROM 1 TO NG.
STORE THE BOTTOM EVENTS WHICH ARE INPUT TO A GIVEN

C GATE IN ARRAY IBEV(M,N) WHERE M GOES FROM 1 TO NB.
C SINCE NG+NB IS LESS THAN OR EQUAL TO 7,
C WE CAN STORE 10*NG+NB IN ARRAY NGB(N) TO GIVE THE
C NUMBER OF GATES AND BOTTOM EVENTS INPUT TO A GIVEN
C GATE.

C 1.3 TEST FOR THE BEGINNING OF RATE DATA, BYPASSING
C RECORDS UNTIL AN *RATE RECORD IS READ.

C 1.4 STORE BOTTOM EVENT NAMES IN ARRAY SNAME(N).
C ON ENCOUNTERING AN END CARD OR AN END OF FILE
C GO TO SECTION 2.0

C 2.0 THIS SECTION WRITES THE CROSS-REFERENCE MAP OF
C THE FAULT TREE.

C 2.1 FIND THE GATES INTO WHICH EACH GATE IS INPUT.

C 2.2 NAMES OF ALL INPUT GATES AND BOTTOM EVENTS
C ALONG WITH THE NAMES OF ALL GATES INTO WHICH
C THE GATE IS INPUT ARE WRITTEN TO FILE REFILE.
C REFILE IS WRITTEN IN PAGES 124 CHARACTERS WIDE
C BY 49 LINES LONG WITH COLUMN HEADINGS AT THE
C TOP OF EACH PAGE.

C 2.3 FIND THE NAMES OF THE GATES INTO WHICH EACH BOTTOM
C EVENT IS INPUT.

+	LABELS/3HAND,	2HOR,	7HINHIBIT,	7HDIAMOND,
+	6HCIRCLE,	5HHOUSE,	7H*ERROR*/	

IF (BLANK.NE.(7H*TREE)) GO TO 1

```
IF (BLANK.EQ.(7HEND      )) GO TO 9
```

```

NUMG=N
DECODE (80,6,AREA) GNAME(N),NG,NB,(INAME(M),M=1,7)
6 FORMAT (A10,11X,2I1,1X,7(1X,A7))
DO 7 M=1,7
  IGATE(M,N)=IBEV(M,N)=7H
  IF (M.LE.NG) IGATE(M,N)=INAME(M)
  IF ((M.GT.NG).AND.(M.LE.NG+NB)) IBEV(M-NG,N)=INAME(M)
7 CONTINUE
  NGB(N)=10*NG+NB
8 CONTINUE
C      1.3 TEST FOR BEGINNING OR RATE DATA:
9 DECODE (7,17,AREA(9)) BLANK
  IF (BLANK.EQ.(7H*RATES)) GO TO 11
10 READ (5,2) (AREA(I),I=1,8)
  DECODE (7,17,AREA) BLANK
  IF (EOF(5).NE.0) GO TO 14
  IF (BLANK.NE.(7H*RATES)) GO TO 10
C
C      1.4 STORE BOTTOM EVENTS.
C
11 DO 13 N=1,149
  READ (5,4) AREA
  IF (EOF(5).NE.0) GO TO 14
  DECODE (10,12,AREA) BNAME(N)
12 FORMAT (A10)
  IF (BNAME(N).EQ.(7HEND)) GO TO 14
13 NUMB=N

```

```

14 CONTINUE
    WRITE (4,15)
15 FORMAT (1H1)
16 FORMAT (1H1////////1X,4HGATE,6X,4HGATE,8X,9HINPUTS TO,51X,9HTHIS GA
    +TE/1X,4HNAME,6X,4HTYPE,8X,9HTHIS GATE,51X,19HIS INPUT TO GATES: /1
    +X,7(1H-),3X,7(1H-),3X,54(1H-),7X,48(1H-)/)
17 FORMAT (A7)
C      2.1 FIND INPUTS AND OUTPUTS OF EACH GATE.
    NWRITE=46
    DO 28 M=1,NUMG
    MATCH=0
    DO 19 N=1,NUMG
    NG=NGB(N)/10
    IF (NG.LT.1) GO TO 19
    DO 18 J=1,NG
    IF ((GNAME(M).A.MASK).NE.(IGATE(J,N).A.MASK)) GO TO 18
    MATCH=MATCH+1
    GMATCH(MATCH)=(GNAME(N).A.MASK).O.(BLANK2.A..N.MASK)
    GO TO 19
18 CONTINUE
19 CONTINUE
C      2.2 WRITE A GATENAME LINE WITH ASSOCIATED INPUTS
C      AND OUTPUTS.
    NG=NGB(M)/10
    NB=NGB(M)-NG*10
    DO 22 N=1,7
    IF (N.GT.NG) GO TO 20

```

```
      INAME(N)=IGATE(N,M)
      BAST(N)=BLANK2
      GO TO 22
20  IF (N.GT.NG+NB) GO TO 21
      INAME(N)=IBEV(N-NG,M)
      BAST(N)=ASTER
      GO TO 22
21  INAME(N)=7H
      BAST(N)=BLANK2
22  CONTINUE
      IF (MATCH.GE.18) GO TO 24
      J=MATCH+1
      DO 23 N=J,18
23  GMATCH(N)=7H
24  CONTINUE
      IF ((NG+NB.LT.7).AND.(MATCH.LT.7)) NWRITE=NWRITE+2
      IF ((NB+NG.EQ.7).OR.(MATCH.GT.6)) NWRITE=NWRITE+3
      IF (MATCH.GT.12) NWRITE=NWRITE+1
      IF ((NWRITE.LE.45).AND.(NWRITE.GT.1)) GO TO 25
      WRITE (4,16)
      NWRITE=1
      GO TO 24
25  DECODE (10,26,GNAME(M) )GNAM,GTYP
26  FORMAT (A7,1X,A2)
      K=7
      IF (GTYP.EQ.2HAN) K=1
      IF (GTYP.EQ.2HOR) K=2
```

```

      IF (GTYP.EQ.2HIN) K=3
      WRITE (4,29) GNAM,LABELS(K),((BAST(I),INAME(I)),I=1,6),(GMATCH(I),
+I=1,6)
      IF ((NG+NB.LT.7).AND.(MATCH.LT.7)) GO TO 27
      WRITE (4,30) BAST(7),INAME(7),(GMATCH(I),I=7,12)
      IF (MATCH.LT.13) GO TO 27
      WRITE (4,31) (GMATCH(I),I=13,18)
27 WRITE (4,32)
28 CONTINUE
29 FORMAT (1X,A7,3X,A7,3X,6(A2,A7),7X,6(1X,A7))
30 FORMAT (21X,A2,A7,52X,6(1X,A7))
31 FORMAT (82X,6(1X,A7))
32 FORMAT (1X)
33 FORMAT (5X,A7,10X,A7,7X,6(1X,A7))
34 FORMAT (36X,6(1X,A7))
35 FORMAT (1H1////////1X,12HBOTTOM EVENT,6X,12HBOTTOM EVENT,6X,17HTHIS
+ BOTTOM EVENT/5X,4HNAME,14X,4HTYPE,10X,18HIS INPUT TO GATES://
+2X,10(1H-),8X,10(1H-),7X,48(1H-)/)
C      2.3 FIND THE OUTPUT GATES ASSOCIATED WITH EACH BOTTOM
C      EVENT.

      NWRITE=46
      DO 42 M=1,NUMB
      MATCH=0
      DO 37 N=1,NUMG
      NG=NGB(N)/10
      NB=NGB(N)-10*NG
      IF (NB.LT.1) GO TO 37

```



```
DO 36 J=1,NB
  IF ((BNAME(M).A.MASK).NE.(IBEV(J,N).A.MASK)) GO TO 36
  MATCH=MATCH+1
  GMATCH(MATCH)=(GNAME(N).A.MASK).O.(BLANK2.A..N.MASK)
  GO TO 37
36 CONTINUE
37 CONTINUE

C          2.4 WRITE A BOTTOM EVENT LINE.
  IF (MATCH.GE.18) GO TO 39
  J=MATCH+1
  DO 38 N=J,18
38 GMATCH(N)=7H
39 IF (MATCH.LT.7) NWRITE=NWRITE+2
  IF (MATCH.GT.6) NWRITE=NWRITE+3
  IF (MATCH.GT.12) NWRITE=NWRITE+1
  IF ((NWRITE.LE.45).AND.(NWRITE.GT.1)) GO TO 40
  WRITE (4,35)
  NWRITE=1
  GO TO 39
40 DECODE (10,26,BNAME(M) )BNAM,GTYP
  K=7
  IF (GTYP.EQ.2HDM) K=4
  IF (GTYP.EQ.2HCR) K=5
  IF (GTYP.EQ.2HMS) K=6
  WRITE (4,33) BNAM,LABELS(K),(GMATCH(I),I=1,6)
  IF (MATCH.LT.7) GO TO 41
  WRITE (4,34) (GMATCH(I),I=7,12)
```

```
IF (MATCH.LT.13) GO TO 41  
WRITE (4,34) (GMATCH(I),I=13,18)  
41 WRITE (4,32)  
42 CONTINUE  
WRITE (4,15)  
END
```

APPENDIX G

METHOD FOR PROBABILITY EVALUATION OF CUTSETS*Introduction

It is assumed that a computer code or other means is available which lists all the minimal cutsets of a fault tree. Cutsets will have the following appearance: x_1 , x_3x_5 , $x_9x_{11}x_{42}$, etc. Each x_i can be an input event, such as a diamond, a circle, or a house (or switch).

The failure probability of the top event is required. This is conservatively approximated by the sum of the probabilities of all the cutsets.

The failure probability of each cutset is thus required. This is the probability that each input event of a particular cutset will occur concurrently at some time during the mission period.

Equations for Evaluation of Cutsets⁽¹⁾A. All Nonrepairable Components

If all of the n components in a cutset are nonrepairable,

*Please see the Acknowledgements for the source of this material.

¹The approach for the majority of this material stems from the August 1974, draft of WASH-1400, Appendix II, Volume 1, pp. 51-87, and Appendix II, Volume 3, pp. 38-29.

P (defined as the probability of failure in the time interval from zero to t) is given by

$$P = (1 - e^{-\lambda_1 t})(1 - e^{-\lambda_2 t}) \dots (1 - e^{-\lambda_n t}) \quad (1A)$$

where

λ_i = operating failure rate of component i

t = operating time or mission time

If $\lambda t < 0.1$, then $(1 - e^{-\lambda t})$ can be replaced by λt with very little error. The approximation, which will be used throughout the remainder of this discussion, is always conservative. Equation (1A) becomes

$$P = \lambda_1 t \lambda_2 t \dots \lambda_n t \quad (1)$$

For purposes of computing P, all one-component cutsets may be considered to be nonrepairable.

B. All Repairable Components

If all of the n components in a two-event or larger cutset are repairable, the failure probability is

$$P = Q_2 Q_3 \dots Q_n \lambda_{o_1} t + Q_1 Q_3 \dots Q_n \lambda_{o_2} t + \dots + Q_1 Q_2 \dots Q_{n-1} \lambda_{o_n} t \quad (2)$$

where

λ_{o_i} = operating failure rate of component i

Q_i = unavailability of component i, defined as the probability of being in a failed state when required

or as the fraction of the time that the component is is down. The values of Q_i for Eq. (2) are given in the following paragraphs.

For an on-line component, e.g., the event

$$Q_i = \lambda_{o_i} \tau_i + \frac{t_{M_i}}{t_{TI_i}} + \frac{t_{M_i}}{t_{MI_i}} \quad (3)$$

where

λ_{o_i} = operating failure rate

$\tau_i = \tau_{D_i} + \tau_{R_i}$ = average duration time of the fault

τ_{D_i} = average time to detect failure⁽²⁾

τ_{R_i} = average time to repair a failure, once detected

T_i = average test downtime (some components can be tested on-line, in which case $t_{T_i} = 0$)

t_{TI_i} = average interval between tests

t_{M_i} = average maintenance downtime

t_{MI_i} = average maintenance interval

²If the component is not monitored, but is periodically tested,

$\tau_{D_i} = t_{TI_i}/2$

A standby component is one which is not on-line (and generally not not operating) until demanded. All repairable components can be considered either on-line or standby, but not both. Thus, the two categories are all-encompassing, as well as mutually exclusive. For a standby component, e.g., the event

Q_i consists of contributions which originate prior to the demand (undetected failures, test and maintenance) and of those which occur at the demand (startup transients, etc.).

$$Q_i = \lambda_{s_i} \tau_i + Q_{d_i} + \frac{t_{T_i}}{t_{TI_i}} + \frac{t_{M_i}}{t_{MI_i}} \quad (4)$$

where

- $\tau_i, t_{T_i}, t_{TI_i}, t_{m_i},$ and t_{MI_i} are as used previously,
- λ_{s_i} = standby failure rate (related to a standby component failing before it is demanded). λ_{s_i} may or may not equal λ_{o_i} . If the standby component is kept operating (but not on-line), λ_{s_i} often equals λ_{o_i} .
- Q_{d_i} = the probability of component i failing to function on demand, given that it was capable of functioning immediately prior to the demand.

C. Additional Considerations

Equations (1) - (4) are the basic results for evaluation of cutsets. However, several considerations necessitate adjustments to these equations. The considerations are discussed in the following paragraphs.

1. Special Considerations for Standby Components

If component i is a standby component, the value of λ_{oi} in Eq. (2) is set equal to zero. This causes the i^{th} term to drop out. For example, given a two-member cutset, x_1x_2 , where component 1 is on-line and component 2 is its standby, Eq. (2) becomes

$$P = Q_2 \lambda_{o1} t.$$

This expression consists of the probability of (on-line) component 1 failing during the mission multiplied by the probability that (standby) component 2 is unavailable when demanded.

An additional term could be added to the right side of P representing the situation in which component 1 fails, then component 2 comes on-line but fails before component 1 is repaired. The probability of this sequence can be shown to be approximately

$$P' = (\lambda_{o1} t)(1 - Q_2)(\lambda_{o2} \tau_1)$$

if $\lambda_{o1} t$ and $\lambda_{o1} \tau_1$ are both < 0.1 . The three terms on the right side of P' may be thought of as representing (1) the

probability that component 1 stops operating before time t ,
 (2) the probability that component 2 comes on-line, and (3)
 the probability that component 2 fails between the time that
 1 goes down and the time that 1 is repaired, i.e., the repair
 time of component 1.

The system failure contribution, P' , from standby components
 starting on demand, then failing, can generally be neglected
 compared with the contribution, P , from their failing to start.
 Since $(1 - Q_2) \approx 1$,

$$\frac{P'}{P} \approx \frac{\lambda_{O_2} \tau_1}{Q_2}$$

The time τ_1 for which a standby component must operate in an
 emergency while its (formerly) on-line counterpart is repaired
 is generally no more than a few hours. (Of course, if compo-
 nent 1 were nonrepairable, τ_1 could approach the mission dura-
 tion, t). Consequently, $\lambda_{O_2} \tau_1$ will generally be $\ll Q_2$. For
 example, a standby fan with $\lambda_{O_2} = 10^{-6}/\text{hr}$ and $\tau_1 = 2 \text{ hr}$ gives
 $\lambda_{O_2} \tau_1 = 2 \times 10^{-6}$, whereas [see Eq. (4)] $Q_d = 10^{-4}$, $t_T/t_{TI} =$
 $t_M/t_{MI} = 10^{-3}$, so Q_2 is at least 2×10^{-3} . Thus,
 $P'/P \approx 10^{-3}$.

2. Mixed Cutsets

If m of the n components in a cutset are nonrepairable, then

Eq. (2) is modified so that the factor

$$\frac{(m-1)!}{m!} = \frac{1}{m} \quad m \neq 0$$

multiplies each term whose $\lambda_{o_i} t$ represents a nonrepairable component. Thus, in general, m of the n additive terms in Eq. (2) would be multiplied by this factor. It can be shown that the remaining $n-m$ additive terms are each multiplied by the factor $\frac{1}{m+1}$. These $n-m$ terms are those whose $\lambda_{o_i} t$ represents a repairable component.

It is noted that the unavailability Q_i for a nonrepairable component i is $\lambda_i t$.

Example - given the cutset $x_1 x_2 x_3 x_4$

where x_1, x_3, x_4 are nonrepairable.

Thus $n = 4, m = 3$

Equation (2) gives

$$P = Q_2 Q_3 Q_4 \lambda_1 t + Q_1 Q_3 Q_4 \lambda_2 t + Q_1 Q_2 Q_4 \lambda_3 t \\ + Q_1 Q_2 Q_3 \lambda_4 t.$$

This result is adjusted by $\frac{1}{m} = \frac{1}{3}$ for terms involving λt of a nonrepairable component as the last factor.

The term involving $\lambda_2 t$ is adjusted by $\frac{1}{m+1} = \frac{1}{4}$. The result is

$$P = \frac{1}{3} Q_2 Q_3 Q_4 \lambda_1 t + \frac{1}{4} Q_1 Q_3 Q_4 \lambda_2 t + \frac{1}{3} Q_1 Q_2 Q_4 \lambda_3 t \\ + \frac{1}{3} Q_1 Q_2 Q_3 \lambda_4 t$$

3. Allowed Outage Times

If $\tau_{\max}$ is the maximum allowed outage time without propagating a failure sequence, then for each component i with this condition, the factor

$$f_i = P(\tau_{D_i} + \tau_{R_i} < \tau_{\max_i}) = e^{-\tau_{\max_i}/(\tau_{D_i} + \tau_{R_i})} \quad (5)$$

is applied to every such Q_i or $\lambda_{o_i} t$ appearing in Eq. (2). In some cases, more than one component within a cutset may have such a factor. If m of the n components in the cutset have such factors, then each of the n additive terms in Eq. (2) will have m multiplicative f -factors.

4. Inhibit Conditions, Houses, and Switches

Term x_i of a cutset may represent an inhibit condition to term x_j with an inhibit fraction k (where $0 < k < 1$). If all components in the cutset are nonrepairable, then the factor k multiplies $\lambda_j t$ in Eq. (1). If there are ℓ such inhibit conditions in the cutset, there will be ℓ multiplicative k -values in Eq. (1). For cutsets of repairable components, the factor k (where k is an inhibit fraction for x_j) multiplies each $\lambda_j t$ and Q_j , for each additive term in Eq. (2). If there are ℓ inhibit conditions in the cutset, there will be ℓ multiplicative k -values in each additive term of Eq. (2).

If term x_i of a cutset is a house (condition which is always present, an "on switch" with a probability of one) an analogous

approach is used. Then $\lambda_{o_i} = 0$ and $Q_i = 1$ in Eq. (2).

The opposite "off switch" condition of zero probability is handled by setting $\lambda_{o_i} = 0$ and $Q_i = 0$.

(This page has been intentionally left blank)

DISTRIBUTION

Number of Copies

10	<u>Battelle Pacific Northwest Laboratories</u> H. C. Burkholder J. D. Kaser E. S. Murphy P. H. Pelto L. D. Perrigo R. I. Smith T. H. Smith S. L. Sutter L. D. Williams K. M. Winegardner
4	<u>Computer Sciences Corporation</u> J. H. Diven G. C. Main J. J. Ratcliffe A. L. Thorpe
5	<u>U. S. Energy Research and Development Administration, Richland Operations Office</u> O. J. Elgert
450	<u>U. S. Energy Research and Development Administration, Technical Information Center; Oak Ridge, TN</u>
7	<u>Hanford Engineering Development Laboratory</u> J. W. Hagan J. P. Hale W. W. Little D. E. Simpson R. J. Slember D. D. Stepnewski W. E. Taylor
1	<u>United Nuclear Industries, Inc.</u> P. A. Crosetti

DISTRIBUTION (Continued)

Number of Copies6 Olympic Engineering

R. Parks
R. Prosser (5)

38 Atlantic Richfield Hanford Company

G. E. Backman
L. E. Brownell
R. E. Felt
R. G. Geiger
D. E. Griffing (5)
N. L. Gross
D. Gruden
J. E. Hammelman
J. C. Hanson
O. F. Hill
H. H. Hopkins
R. E. Isaacson
D. E. Larson
W. E. Matheison
B. J. McMurray
J. V. Panesko
C. M. Peabody
K. R. Price
W. C. Schmidt
H. P. Shaw
S. P. Stalos
G. L. Troyer
W. J. Van Slyke (10)
R. A. Watrous
D. D. Wodrich