Microcomputer Applications of, and Modifications to, the Modular Fault Trees

Manuscript Completed: September 1994
Date Published: October 1994

Prepared by
T. L. Zimmerman*, N. L. Graves*, A. C. Payne, Jr.,
D. W. Whitehead

Sandia National Laboratories
Albuquerque, NM 87185

Prepared for
Division of Safety Issue Resolution
Office of Nuclear Regulatory Research
U.S. Nuclear Regulatory Commission
Washington, DC 20555-0001
NRC FIN A1386

*Formerly with Energy Incorporated, Albuquerque, NM 87109

MASTER

DISTRIBUTION OF THIS DOCUMENT IS UNLIMITED

me

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, make 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.

ABSTRACT

The LaSalle Probabilistic Risk Assessment was the first major application of the modular logic fault trees after the IREP program. In the process of performing the analysis, many errors were discovered in the fault tree modules that led to difficulties in combining the modules to form the final system fault trees. These errors are corrected in the revised modules listed in this report.

In addition, the application of the modules in terms of editing them and forming them into the system fault trees was inefficient. Originally, the editing had to be done line by line and no error checking was performed by the computer. This led to many typos and other logic errors in the construction of the modular fault tree files. Two programs were written to help alleviate this problem: (1) MOEDIT - This program allows an operator to retrieve a file for editing, edit the file for the plant specific application, perform some general error checking while the file is being modified, and store the file for later use, and (2) INDEX - This program checks that the modules that are supposed to form one fault tree all link up appropriately before the files are loaded onto the mainframe computer.

Lastly, the modules were not designed for relay type logic common in BWR designs but for solid state type logic. Some additional modules were defined for modeling relay logic, and an explanation and example of their use are included in this report.

 

Table of Contents

	Page
ABSTRACT.....	iii
LIST OF FIGURES.....	vii
LIST OF TABLES.....	viii
ACKNOWLEDGMENTS.....	ix
 Chapter 1 - INTRODUCTION.....	 1
Chapter 2 - MICROCOMPUTER APPLICATIONS.....	2
Chapter 3 - REVISED MODULES.....	3
Reference	5
 Appendix A - MODULAR FAULT TREE MICROCOMPUTER SOFTWARE	
A.1 Purpose.....	A-1
A.2 Overview.....	A-1
A.3 Initiation Procedure.....	A-3
A.4 MODEDIT Operation.....	A-3
A.5 A Sample MODEDIT Session.....	A-13
A.6 Delete Gates/Events.....	A-14
A.7 Delete Single Lines.....	A-14
A.8 Add Gates/Events.....	A-14
A.9 Word Substitutions.....	A-15
A.10 Save Module.....	A-16
A.11 Edit New Module or QUIT.....	A-18
Reference	A-19
 Appendix B - INDEX DOCUMENTATION, INDEX and INDEX2	
B.1 Overview.....	B-1
B.2 Operation.....	B-1
B.3 Output.....	B-13
 Appendix C - REVISED MODULAR FAULT TREE LISTINGS	
C.1 Introduction.....	C-1
C.2 Updated Modular Fault Trees.....	C-1
Reference	C-2
 Appendix D - RELAY ACTUATION AND COMPONENT CONTROL MODULES	
D.1 Introduction.....	D-1
D.2 General Precautions.....	D-1
D.3 Overview of Use.....	D-2
D.4 Module CC1.....	D-5
D.5 Module CC2.....	D-6
D.6 Module CC3.....	D-7
D.7 Module CC4.....	D-8

Table of Contents (Cont.)

		Page
D.8	Module CC5.....	D-8
D.9	Module CC6.....	D-8
D.10	Module CC7.....	D-9
D.11	Module CC8.....	D-9
D.12	Module CC9.....	D-9
D.13	Module CC10.....	D-9
D.14	Module CC11.....	D-9
D.15	Module CC12.....	D-10
D.16	Example.....	D-10
D.17	Master Relay and Actuation Component	
	Control (RACC) Modules	D-148
Reference		D-168

List of Figures

Appendix D Figures

Figure		Page
Example System Drawings		
D-1	HIS System P&ID.....	D-37
D-2	HIS pump C001 Control Circuit.....	D-38
D-3	HIS MOV F010 Control Circuit.....	D-39
D-4	HIS System Actuation Circuits.....	D-40
D-5	System 75 Tank Level Instrumentation Channels A and B.....	D-41
D-6	Pump Start Network.....	D-42
D-7	Pump Stop Network.....	D-42
D-8	Valve F010 Open Network.....	D-43
D-9	Valve F010 Close Network (C coil).....	D-43
D-10	HIS Initiation Network.....	D-44
D-11	Pump Low Suction Pressure Network (K4).....	D-44
D-12	Valve F010 Tank Hi Level Auto Close Network (K6).....	D-45
D-13	System Low Flow Network.....	D-45
D-14	Pump Low Flow Trip Network (K10).....	D-46
Example Fault Tree Modules		
D-15 to D-65.....		D-47 to D-147
Master Relay and Actuation Component Control Modules		
CC1.....		D-149
CC2.....		D-151
CC3.....		D-152
CC4.....		D-155
CC5.....		D-157
CC6.....		D-158
CC7.....		D-159
CC8.....		D-160
CC9.....		D-161
CC10.....		D-164
CC11.....		D-166
CC12.....		D-167

List of Tables

Table		Page
A.1	Program Listing of MOEDIT.....	A-5
B.1	Program Listing of INDEX.....	B-3
B.2	Program Listing of INDEX2.....	B-7
B.3	INDEX Batch File.....	B-11
B.4	INDEX2 Batch File.....	B-12
D.1	Character String Substitution Guidelines.....	D-18

ACKNOWLEDGMENTS

The authors wish to thank S.J. Higgins for figuring out how to plot the modules after major changes to the computer systems rendered previous methods unusable.

CHAPTER 1 INTRODUCTION

The modular fault tree methodology, developed by Sandia National Laboratories [1], was used to construct the system fault trees for the Risk Methods Integration and Evaluation Program (RMIEP) probabilistic risk assessment (PRA) of the LaSalle Unit 2 nuclear power plant. During this effort, microcomputers were used to create plant-specific system modules from the generic modules in Reference 1 and generic modules developed during the course of this project. Methodologies developed and solutions to problems encountered in this application are discussed in the remainder of this report.

CHAPTER 2 MICROCOMPUTER APPLICATIONS

In this effort, the generic modules in Appendix I (volume 4) of [1] were stored, in SETS format (i.e., Set Equation Transformation System format), as individual files on a microcomputer floppy disk. A computer program (MODEDIT), developed by Energy Incorporated, was used to retrieve these files from the disk for modification into plant-specific modules. The modified file was then saved to a separate disk. When all modules applicable to a system fault tree were completed, the files were transported to a mainframe for linking into a SETS format system fault model. This completed system fault model could then be solved or plotted using the SETS code.

The MODEDIT program was developed for an IBM-PC. With this program, a computer operator can easily add events to or delete them from the generic module and make character string substitutions within the module. This enhances the speed and efficiency of creating plant-specific modules. In addition, the program contains a subroutine which checks a completed module for errors that may have been generated during modification. The MODEDIT program source code along with a complete explanation of its use, including equipment requirements, is provided in Appendix A. A copy of the MODEDIT program, the generic modules from Appendix I of [1], and the new generic control and actuation modules designed to model BWR relay logic may be obtained from the Argonne Code Library.

During the mainframe linking process, typographical errors in the completed system modules created unnecessary delays in achieving a complete system fault model. To circumvent this occurrence, an additional microcomputer program (INDEX) was developed for the IBM-PC. This program checks all unlinked system files on a disk to identify developed events which do not have a corresponding top gate in another file. These are then listed so that the analyst can make any necessary corrections before the files are transferred to the mainframe for linking. This reduces losses of mainframe time occurring as a result of linking errors. The INDEX program source code along with a complete explanation of its use, is included as Appendix B. A copy of the INDEX program may be obtained from the Argonne Code Library.

CHAPTER 3 REVISED MODULES

Typographical and logic errors, as well as other difficulties, encountered in using the generic modules from Appendix I of [1] were corrected as they were discovered. These include:

1. Generic character strings with different spellings resulting in incomplete character string substitutions.
2. Modules lacking developed events which would allow easy transition to other modules necessary to develop control, test and maintenance, (etc.) faults.
3. Generic test and maintenance modules specifically for control and actuation circuits.
4. Events specifically tailored to identify component unavailabilities due to scheduled tests or surveillances.
5. Event descriptions which were too brief to adequately convey useful information.
6. Logic structures difficult to apply to flow diversions in some applications.

In addition, a follow-on review of the generic modules was completed to identify and correct any remaining errors. The generic modules to which substantial modifications were made are identified below by their corresponding page number in Appendix I.

Modules which had control and actuation events added:

I34, I35, I36, I38, I39, I40, I41, I43, I77, I78, I79, I100, I109, and I120.

Modules which had event descriptions modified:

I4, I5, I6, I7, I8, I9, I10, I11, I12, I23, I24, I25, I26, I30, I31, I32, and I33.

These updated modules are included in Appendix C.

Difficulties were encountered in developing component control and actuation fault trees using the existing control and actuation modules. Because of this, a completely new set of control and actuation modules specifically designed for modeling BWR relay type logic was developed. These modules (designated CCl - CCl2)

along with a complete description of their use, including a comprehensive example, are included in Appendix D. A copy of these modules may be obtained from the Argonne Code Library.

REFERENCE

1. Varnado, G. B., W. H. Horton, and P. R. Lobner, Modular Fault Tree Analysis Procedures Guide, NUREG/CR-3268, SAND83-0963, (4 volumes), Sandia National Laboratories, Albuquerque, NM, August 1983.

APPENDIX A

MODULAR FAULT TREE MICROCOMPUTER SOFTWARE

A.1 Purpose

When using the modular fault tree system for developing a system fault tree, the advantages of a microcomputer based system for accessing, editing and saving the individual modules of the fault tree becomes obvious. Software has been generated that greatly reduces the time required to prepare input for the SETS fault tree analysis code. In addition, error checking within the program ensures that input errors which might prevent code execution when the modules are merged by SETS to form the system fault tree are kept to a minimum. It is much more efficient in terms of both time and cost to detect such errors in preprocessing on the microcomputer rather than have the SETS program terminate execution on the mainframe computer.

The MOEDIT program performs recall, editing, error checking, and storing on individual SETS input modules. The INDEX program prints an index of and performs error checking across all modules on a floppy disk.

A.2 Overview

The fault tree modules described and plotted in Volumes 2, 3, and 4 of [1] (hereafter called the Procedures Guide) have been pre-coded in correct SETS input format (including the block text needed for plotting) and stored as ASCII files. The MOEDIT program allows the user to call up any desired logic module using the associated Procedures Guide, Appendix I page number* (e.g., I24 or I139). A MOEDIT menu allows him to select from the following functions:

1. Delete gates and/or events from the module (including the descriptive text associated with the gate or event)
2. Delete single lines of module input coding
3. Add new gates or events
4. Make word substitutions in the module coding

* Instead of choosing to edit a "generic" module from Appendix I, the user can also re-edit a previously saved module by entering the file name under which it was saved.

5. Save the newly edited module on a disk under a user specified name
6. Edit another module or QUIT

Before storing an edited module on disk, MODEDIT automatically performs certain error checks for correct SETS input syntax and generic words that the user may have failed to replace with system specific labels. MODEDIT prompts the user when necessary and recovers from improper responses with error messages and another opportunity to respond correctly. It should be noted that the Appendix I modules have been stored as ASCII files. Hence virtually any of the commonly available PC editors/word processors (including the EDLIN editor provided with PC DOS) may be used to edit a module. The editor, of course, must be capable of operating in a mode that does not insert special control characters into the module file (usually "non-document" mode). The danger in hacking away at a module file with such an editor is that none of MODEDIT's checks for correct SETS syntax and fault tree structure will be in effect. For example, if the user deletes a line defining a gate using EDLIN, he must make certain that he also deletes all lines defining inputs to that gate and he must delete the gate name from the input list of the gate above it. He must also delete the lines of descriptive text for the deleted gate/event blocks. MODEDIT performs these functions automatically. In cases where the use of a PC editor or word processor is warranted (e.g., building a module from scratch or making extensive structural changes in a module) the user should reload the edited module file into MODEDIT and re-save it using the MODEDIT save option. This will ensure that at least some of the error checking will be performed on this new SETS input module.

Once all the modules needed to model a system have been edited and stored on a disk, further error checking and an index showing the file name, top gate name and descriptive text of each module can be printed out using the INDEX program. INDEX primarily checks for modules that do link up properly, (i.e., a "developed event" in one module that has no corresponding top event in another module) a fairly common problem due to user inconsistency in naming events. INDEX also checks for multiple occurrences of the same gate name, and module top events that do not link to any other module (there should be only one module of this last type for most fault trees). Once INDEX has been run on the modules, the data on the disk(s) is ready to be transferred onto the mainframe computer so that the modules can be merged using SETS.

A.3 Initiation Procedure

After obtaining the appropriate information from the Argonne Code library, the following files should be copied to a floppy disk using the specified directories identified below:

/ (Root Directory)

MODEDIT.BAS - The basic program

/ Thru50

Contain the generic modules found in Appendix I up to page 50.

/ Thru100

Contains the generic modules found in Appendix I from page 51 up to and including 100.

/ Thru200

Contains the generic modules found in Appendix I from page 101 up to and including page 157 and the CCl thru CCl2 modules named I300 to I311, respectively.

This is recommended so that the original files can be stored for safe keeping, and the copied files can be used for an analysis. It is highly recommended that any modifications done to these modules when developing a particular system fault tree be saved on a separate disk. The disk to be used for saving modified modules should be formatted (refer to the DOS manual for your computer) and should have a /SETS subdirectory. MODEDIT will then save the newly modified fault tree modules in the /SETS directory.

A.4 MODEDIT Operation

Version VIII of the MODEDIT program (see Table A.1 for a listing of the source code) is written in IBM PC advanced BASIC Version 3.0. The Operating system should be DOS 2.0 or higher. It does not require any special graphics capabilities. To use the MODEDIT program, BASICA must be initiated first and the MODEDIT.BAS program loaded and run.

The user is directed either by user prompts or menus to the various options within MODEDIT. Most responses are not executed until the "enter" (return) key is pressed. The CAPSLOCK key should be activated for all responses to prompts. Most improper responses result in an error message being displayed and the user

given another chance to make an acceptable response. On the rare occasions that an error is made which is not recoverable, the "FATAL ERROR" message is displayed and control is returned to the BASIC environment. The user then either enters "SYSTEM" to return to the DOS environment or "RUN" (F2 key) to restart MODEEDIT. In either case, only the changes made to the single module being edited at the time are lost.

Table A.1
Program Listing of MODEDIT

```

1 REM *****
2 REM                      MODEDIT CODE - JUNE 1988
3 REM *****
4 REM
5 REM                      ISSUED BY SANDIA NATIONAL LABORATORIES
6 REM                      A PRIME CONTRACTOR TO THE
7 REM                      UNITED STATES DEPARTMENT OF ENERGY
8 REM
9 REM ***** N O T I C E *****
10 REM
11 REM THIS CODE WAS PREPARED IN THE COURSE OF WORK SPONSORED BY THE
12 REM UNITED STATES GOVERNMENT. NEITHER THE UNITED STATES NOR
13 REM THE UNITED STATES DEPARTMENT OF ENERGY, THE UNITED STATES
14 REM NUCLEAR REGULATORY COMMISSION, NOR ANY OF THEIR EMPLOYEES,
15 REM NOR ANY OF THEIR CONTRACTORS, SUBCONTRACTORS, OR THEIR
16 REM EMPLOYEES, MAKES ANY WARRANTY, EXPRESSED OR IMPLIED, OR
17 REM ASSUMES ANY LEGAL LIABILITY OR RESPONSIBILITY FOR THE
18 REM ACCURACY, COMPLETENESS OR USEFULNESS OF ANY INFORMATION,
19 REM APPARATUS, PRODUCT OR PROCESS DISCLOSED, OR REPRESENTS THAT
20 REM ITS USE WOULD NOT INFRINGE PRIVATELY OWNED RIGHTS.
21 REM
22 REM *****
23 REM
24 REM AUTHOR: T.L. ZIMMERMAN, FORMALLY OF ENERGY INCORPORATED
25 REM                      ALBUQUERQUE, NM
26 REM
27 REM FOR INFORMATION CONTACT:
28 REM
29 REM A. C. PAYNE, JR.      PHONE: (505) 846-3568
30 REM D. W. WHITEHEAD      PHONE: (505) 844-2632
31 REM
32 REM REACTOR SYSTEMS SAFETY ANALYSIS DIVISION 8412
33 REM SANDIA NATIONAL LABORATORIES, ALBUQUERQUE, NM 87185
34 REM
35 REM *****
36 CLS
40 SCREEN 0,0,0 'To run on either color or mono screens.
50 DEFINT I,J,K,L,N,P
60 DIM TS(110),ES(50)
70 ON KEY(1) GOSUB 4140
80 KEY OFF :PRINT
90 ON ERROR GOTO 4280
100 PRINT SPC(10): COLOR 0,7: PRINT "      <<<< MODEDIT >>>>      ";
110 COLOR 7,0 : PRINT: PRINT SPC(23);"Version VIII"
120 REM *****
130 IF I>0 THEN LOCATE 25,1 : PRINT SPACES(70)
140 PRINT: COLOR 15,0 :PRINT SPC(16);"-- Edit/Generate a Module --": COLOR 7,0
150 PRINT :IF D18="" GOTO 180
160 INPUT "Disk drive containing module to edit: ",D18
170 I=ASC(D18): IF I<65 OR I>72 THEN PRINT "***INCONNECT FORMAT***":GOTO 180
180 INPUT "Enter name of module (Q to Quit)   ": MS(1)
190 IF MS(1)="Q" GOTO 4520 :IF MS(1)="" GOTO 140
200 CHDIR D18+"\":
210 IF LEFT$(MS(1),1)<>"I" GOTO 260
220 NODS=MIDS(MS(1),2): M1=VAL(NODS): IF M1=0 GOTO 260
230 IF M1<51 THEN CHDIR D18+"\\TRU50": GOTO 280
240 IF M1<101 THEN CHDIR D18+"\\TRU100": GOTO 280
250 CHDIR D18+"\\TRU200": GOTO 280
260 CHDIR D18+"\\SETS"
270 REM      Read Module File on Disk
280 FS=D18+"\\*MS(1)

```

Table A.1 (Continued)
Program Listing of MOEDIT

```

290 OPEN FS FOR INPUT AS #1
300 FOR I=1 TO 200
310 LINE INPUT #1, TS(I)
320 DUMS=LEFTS(TS(I),10)
330 IF DUMS="FAULTTREES" OR DUMS="FAULT TREE" THEN I5=I
340 IF TS(I)="FINS" GOTO 360
350 NEXT
360 CLOSE 1: KEY(1) ON
370 I9=I : I3=2 : PRINT
380 I8 = LEFTS(TS(I3),8)
390 IF I8<>"COMMENTS" GOTO 430
400 AS=MIDS(TS(I3),8)
410 PRINT AS ' Print file COMMENTS describing module
420 I3=I3+1: GOTO 380
430 PRINT :COLOR 15,0 :PRINT "  PRESSING 'F1' (ENTER) IN RESPONSE TO ANY QUESTION GENERATES A CURRENT"
440 PRINT "  LISTING OF THE MODULE AND RETURNS TO THE QUESTION": COLOR 7,0
450 LIN = CSRLIN
460 LOCATE 25,25:COLOR 0,7:PRINT "(KEY 'F1' GENERATES A MODULE LISTING)";
470 COLOR 7,0 :LOCATE LIN,1
480 PRINT :COLOR 15,0: PRINT "Menu": COLOR 7,0
490 PRINT "  1. Delete gates/events      4. Word substitutions"
500 PRINT "  2. Delete single lines      5. Save Module"
510 PRINT "  3. Add gates/events          6. Edit New Module or QUIT"
520 PRINT :COLOR 15,0
530 INPUT "Enter # of selection: ";SEL
540 COLOR 7,0
550 ON SEL GOSUB 670,560,1750,2800,3240
560 IF SEL=6 GOTO 130
570 GOTO 480
580 REM *****
590 COLOR 15,0:PRINT:PRINT SPC(12);"--- Delete Single Lines from Module ---":PRINT
600 INPUT "Enter line # of line to be deleted (END to terminate): ";DELS
610 IF SEL=99 OR DELS="" THEN SEL=0: GOTO 500
620 IF DELS="END" THEN GOTO 650 ELSE I=VAL(DELS)
630 PRINT :PRINT "DELETED LINE:": PRINT TS(I): PRINT
640 TS(I)="000" : GOTO 600
650 RETURN
660 REM *****
670 COLOR 15,0
680 PRINT:PRINT SPC(12);"--- Delete Events/Gates from Module ---":PRINT:COLOR 7,0
690 I2=I3: I4=I5-1: I6=I5+1
700 PRINT "Indicate whether you wish to delete any of the following gates/events:": PRINT
710 FOR I=I2 TO I4
720 TIS=TS(I) : PE=0
730 IF TIS="000" GOTO 1730 ' Already deleted
740 IF LEFTS(TIS,3)=" " GOTO 1730
750 PD=INSTR(TIS,".")
760 IF MIDS(TIS,PD-1,1)=" " THEN PD=PD-1 : GOTO 750
770 PB=INSTR(TIS,"$")+1
780 IF MIDS(TIS,PB,1)=" " THEN PB=PB+1 : GOTO 780
790 P2=PD-PB
800 EVENTS=MIDS(TIS,PB,P2)
810 ISET=I : L9=LEN(EVENTS)
820 FOR J=I6 TO I9
830 P=INSTR(TS(J),EVENTS)
840 IF P>0 THEN IF ASC(MIDS(TS(J),P+L9,1))<37 THEN GOTO 880
850 NEXT J
860 PRINT "WARNING --No descriptive match was found for ";TS(I): J=I10
870 TS(I10) = " -NONE-      $"
880 PRINT :PRINT "Event/Gate:"
890 IJ=I : IJ1=I : J2=J

```

Table A.1 (Continued)
Program Listing of MODEDIT

```

900 PRINT TS(IJ)
910 IF LEFT$(TS(IJ+1),3)=" " THEN IJ=IJ+1:GOTO 900 'Print all lines of this gate
920 PRINT "Description:"
930 PRINT TS(J2)
940 IF RIGHT$(TS(J2),1)<>"$" THEN J2=J2+1: GOTO 930
950 COLOR 15,0
960 INPUT "Enter 'D' to delete (ENTER to include): ";D$
970 COLOR 7,0
980 IF SEL=99 THEN SEL=0: GOTO 940
990 IF D$<>"D" GOTO 1730
1000 IF INSTR(TIS,"GS")>0 GOTO 1290
1010 FOR IK=IJ1 TO IJ ' Mark gate/event as 'deleted'
1020 TS(IK)="000"
1030 NEXT IK
1040 FOR IK=J TO J2 ' Mark description as 'deleted'
1050 TS(IK)="000"
1060 NEXT IK
1070 FOR IL=I2 TO I4
1080 IK=IL
1090 PI=INSTR(1,TS(IK),EVENTS) ' Find gates having deleted event as an input
1100 IF PI=0 GOTO 1260
1110 IF ASC(MID$(TS(IK),PI-1,1))>44 GOTO 1260
1120 L3=ASC(MID$(TS(IK),PI+LN,1)) : IF L3>45 OR L3=45 GOTO 1260
1130 PD=INSTR(PI,TS(IK),".")
1140 IF PD<>0 GOTO 1230
1150 PD=INSTR(PI-4,TS(IK),".") ' Event is the last of the gate's inputs
1160 IF PD<>0 THEN TS(IK)=LEFT$(TS(IK),PD-1)+".": GOTO 1260
1170 P2=INSTR(TS(IK),"INS"): IF P2>0 THEN PD=P2+3: GOTO 1160 'Event is ONLY input to the gate
1180 TS(IK)="000" ' Event is last of the gate's inputs and alone on line
1190 IF TS(IK-1)="000" THEN IK=IK-1: GOTO 1180
1200 PD=INSTR(LEN(TS(IK-1))-5,TS(IK-1),".")
1210 IF PD=0 THEN IK=IK-1: GOTO 1170
1220 MID$(TS(IK-1),PD)=".": GOTO 1260
1230 BS=LEFT$(TS(IK),PI-1) ' Eliminate event name & concatenate
1240 ES=MID$(TS(IK),PD+1)
1250 TS(IK)=BS+ES
1260 NEXT IL
1270 GOTO 1730
1280 REM -----
1290 IK=I : K=0 'Find gate's inputs for deletion...
1300 PB=INSTR(TS(IK),"INS")+3 : PP=0
1310 PD=INSTR(PB,TS(IK),".")
1320 IF PD<>0 GOTO 1360
1330 PD=INSTR(PB,TS(IK),".")
1340 IF PD<>0 THEN PP=9: GOTO 1360 'Found the end of gate input
1350 IK=IK+1: PP=1: PB=1: GOTO 1310
1360 IF PP=0 GOTO 1380
1370 PP=0: PB=PD-17: IF PB<1 THEN PB=1
1380 IF MID$(TS(IK),PB,1)=" " THEN PB=PB+1: GOTO 1360
1390 IF MID$(TS(IK),PD-1,1)=" " THEN PD=PD-1: GOTO 1390
1400 K=K+1
1410 ES(K)=MID$(TS(IK),PB,PD-PB) 'Store this gate input name.
1420 IF PP=0 THEN PB=INSTR(PD,TS(IK),".")+1: GOTO 1310
1430 PP=0 : K=K
1440 IF PP=0 GOTO 1540
1450 PRINT " --The following inputs to gate ";EVENTS;" are also being deleted:"
1460 KS=1
1470 FOR L1=I2 TO I4 'Find events/gates to be deleted
1480 IF L1=ISKT GOTO 1520
1490 PE=INSTR(TS(L1),ES(KS))
1500 IF PE=0 GOTO 1520

```

Table A.1 (Continued)
Program Listing of MODEBIT

```

1510 IF PE<INSTR(T$(L1),"INS") OR PE<INSTR(T$(L1),"OUTS") GOTO 1540
1520 NEXT L1
1530 GOTO 1730
1540 IF ASC(MID$(T$(L1),PE-1,1))>38 GOTO 1520
1550 PL=LEN(ES(KS)): L3=ASC(MID$(T$(L1),PE+PL,1))
1560 IF L3<>32 AND L3<>48 GOTO 1520
1570 IF INSTR(T$(L1),"G")>0 THEN IK=L1: GOTO 1300
1580 FOR L2=16 TO 19 'Find corresponding descriptions to delete
1590 PE=INSTR(T$(L2),ES(KS))
1600 IF PE=0 AND PE<INSTR(T$(L2),"$") GOTO 1630
1610 NEXT L2
1620 L2=0
1630 L=L1: PRINT
1640 PRINT SPC(5);T$(L)
1650 T$(L)="000"
1660 PD=INSTR(T$(L+1),"$")+1 'Check for continuation lines...
1670 IF INSTR(PD,T$(L+1),"$")>0 GOTO 1700
1680 IF L>15 THEN IF MID$(T$(L+1),PD,1)>" " OR T$(L+1)="END$" GOTO 1710
1690 L=L+1: GOTO 1640
1700 IF L<L2 THEN L=L2: GOTO 1640
1710 IF KS<K9 THEN KS=KS+1: GOTO 1470
1720 GOTO 1010 'Go back and delete gate.
1730 NEXT I
1740 COLOR 15,0: PRINT: PRINT SPC(15);"---Finished---": RETURN
1750 REM *****
1760 COLOR 15,0
1770 PRINT :PRINT SPC(12);"--- Add Events/Gates to Module ---": COLOR 7,0: PRINT
1780 I4=15: E1$="" : GATINS="" : GATOUT$=""
1790 COLOR 15,0: PRINT "Define SETS event or gate ": COLOR 7,0
1800 PRINT " (Enter 'COPY' to copy an existing definition, 'END' to terminate input)"
1810 LINE INPUT ES
1820 IF SEL=99 THEN IF GATINS="" THEN SEL=0: GOTO 2430 ELSE SEL=0: GOTO 1790
1830 IF ES="END" GOTO 2780
1840 IF ES="COPY" THEN GOTO 1860 ELSE IF INSTR(ES,".")>0 OR INSTR(ES,"$")>0 GOTO 1920
1850 BEEP:PRINT "****IMPROPER FORMAT****": GOTO 1800
1860 INPUT " --Enter the line # of definition to be copied (END to abort):";LINS
1870 IF SEL=99 THEN SEL=0: GOTO 1860
1880 IF LINS="END" THEN IF GATOUT$="" GOTO 1790 ELSE GOTO 2430
1890 I=VAL(LINS): ES=T$(I)
1900 IF I>15 THEN BEEP: PRINT ES:PRINT "****THIS IS NOT A DEFINITION": GOTO 1860
1910 PRINT "LINE COPIED:" : PRINT ES
1920 IF INSTR(ES,"G$")=0 GOTO 2010 'This is an event.
1930 FOR IK=3 TO 15 'Find the last gate...
1940 IF INSTR(T$(IK),"ES")>0 GOTO 1960
1950 NEXT IK
1960 I4=IK: P=4
1970 IF MID$(T$(IK),P,1)=" " THEN P=P+1: GOTO 1970
1980 PP=INSTR(T$(IK),".")
1990 IF MID$(T$(IK),PP-1,1)=" " THEN PP=PP-1: GOTO 1980
2000 E1$=MID$(T$(IK),P,PP-5) 'Record name of first event.
2010 IF GATINS="" GOTO 2080
2020 PE=INSTR(ES,"$")+1
2030 IF PE=1 GOTO 2330
2040 IF MID$(ES,PE,1)=" " THEN PE=PE+1: GOTO 2040
2050 PD=INSTR(PE,ES,".")
2060 IF MID$(ES,PD-1,1)=" " THEN PD=PD-1: GOTO 2060
2070 GATINS=MID$(ES,PE,PD-PE) 'Record the new event/gate name.
2080 PB=INSTR(1,ES,"OUTS")+4 'Find output(s) of event/gate.
2090 IF PB=4 THEN GOTO 2410 ELSE A$=","
2100 PD=INSTR(PB,ES,A$)
2110 IF PD=0 THEN IF A$="," THEN A$="." : GOTO 2100 ELSE PRINT " --Output for this event/gate is incomplete.":

```

Table A.1 (Continued)
Program Listing of MOEDIT

```

GOTO 2410
2120 IF MID$(ES,PD-1,1)="" THEN PD=PD-1: GOTO 2120
2130 IF MID$(ES,PD,1)="" THEN PD=PD+1: GOTO 2130
2140 GATOUTS=MID$(ES,PD,PD-PD) 'Output of new event/gate.
2150 FOR IJ=2 TO 14
2160 P=INSTR(3,TS(IJ),GATOUTS)
2170 IF P>0 GOTO 2220
2180 NEXT IJ
2190 PRINT " ***WARNING -The gate you have given as the output does not currently exist..."
2200 PRINT " Go back and create the output gate ";GATOUTS;" first."
2210 GOTO 1780
2220 PI=INSTR(P+1,TS(IJ),"INS") : IF PI=0 GOTO 2180
2230 IF INSTR(PI,TS(IJ),GATINS)=0 GOTO 2280
2240 PRINT " ---";GATINS;" is already an input to gate ";GATOUTS
2250 INPUT " Do you want it added to the inputs again (Y or N) ";FS
2260 IF SEL=99 THEN SEL=0: GOTO 2240
2270 IF FS="N" GOTO 2410
2280 PP=INSTR(PI,TS(IJ),".")
2290 IF PP=0 THEN IJ=IJ+1: GOTO 2280
2300 P2=INSTR(TS(IJ),"OUTS")
2310 IF P2>0 AND P2<PP THEN PP=PI+2: GOTO 2330
2320 IF INSTR(TS(IJ),"INS.")=0 THEN MID$(TS(IJ),PP)=", " ELSE MID$(TS(IJ),PP)=""
2330 B$=LEFT$(TS(IJ),PP) : E2$=MID$(TS(IJ),PP+1)
2340 TS(IJ)=B$+GATINS+" "+E2$ 'Add new event/gate to input of this gate...
2350 IF LEN(TS(IJ))<=80 GOTO 2360
2360 TS(IJ)=B$
2370 II=IJ+1 : TIS=SPACES(20)+GATINS+" "+E2$
2380 GOSUB 2440
2390 PRINT :PRINT " --This event/gate has been added to the inputs of gate ";GATOUTS
2400 IF AS="" THEN PD=PD+1: GOTO 2100
2410 II=I4 : TIS=E$ : GOSUB 2440 'Insert event/gate.
2420 IF GATOUTS="" GOTO 2510
2430 PRINT :PRINT SPC(10);"Continue definition..." : GOTO 1800
2440 FOR I=I9 TO II STEP -1 'Line Insertion Subroutine:
2450 TS(I+1)=TS(I)
2460 NEXT I
2470 TS(II)=TIS
2480 IF II<=I5 THEN I5=I5+1 : I4=I4+1
2490 I9=I9+1
2500 RETURN 'End of Subroutine.
2510 I6=I9-1
2520 COLOR 15,0: PRINT :PRINT "Enter descriptive label for this event or gate": COLOR 7,0
2530 PRINT " (Enter 'COPY' to copy an existing description, 'END' to terminate input)"
2540 LINE INPUT E$
2550 IF SEL=99 THEN IF GATINS="A" THEN SEL=0: GOTO 2760 ELSE SEL=0: GOTO 2520
2560 IF E$="END" GOTO 1780
2570 IF E$="COPY" GOTO 2610
2580 IF INSTR(E$,"$")=0 THEN BEEP:PRINT "***IMPROPER FORMAT***": GOTO 2530
2590 IF LEN(E$)<=80 GOTO 2670
2600 BEEP:PRINT " ***LINE HAS OVER 80 CHRS (Re-enter as 2 lines)***":GOTO 2520
2610 PRINT " (NOTE: Press 'F1' to determine new line numbers)"
2620 INPUT " --Enter the line # of description to be copied: ";LINS
2630 IF SEL=99 THEN SEL=0: GOTO 2620
2640 I=VAL(LINS) : E$=TS(I)
2650 IF I<I5 THEN BEEP:PRINT E$:PRINT "***THIS IS NOT A DESCRIPTION": GOTO 2620
2660 PRINT "LINE COPIED:" : PRINT E$
2670 IF E1$="" GOTO 2710
2680 FOR IK=I5 TO I9 'Find the last gate description...
2690 IF INSTR(1,TS(IK),E1$)>0 THEN I6=IK : E1$="" : GOTO 2710
2700 NEXT
2710 FOR I=I9 TO I6 STEP -1

```


Table A.1 (Continued)
Program Listing of MOEDBIT

```

2720 TS(I+1)=TS(I)
2730 NEXT
2740 TS(15)=E$
2750 I9=I9+1 : I5=I5+1 : GATIN$="a"
2760 PRINT :PRINT SPC(10);"Continue descriptive label..."
2770 GOTO 2530
2780 RETURN
2790 REM *****
2800 COLOR 15,0: PRINT
2810 PRINT "    ---Module Word Substitution---" : COLOR 7,0: PRINT
2820 COLOR 15,0:LINE INPUT "Enter label to be replaced (END to terminate): ",L1$
2830 IF L1$="END" THEN COLOR 7,0: GOTO 3210
2840 IF SEL=99 OR L1$="" THEN SEL=0: GOTO 2820
2850 LINE INPUT "Enter label to be substituted (END to abort): ",L2$: COLOR 7,0
2860 IF L2$="END" GOTO 2820
2870 IF L1$="COMP" THEN L1$=L1$+" " : L2$=L2$+" " 'Prevents unwanted COMP subst.
2880 IF L1$="TYP" THEN L1$="-"+L1$+"-" : L2$="-"+L2$+"-"
2890 L=LEN(L1$) : L2=LEN(L2$)
2900 PRINT : PRINT "SUBSTITUTIONS: "
2910 COLOR 15,0: PRINT "    Press: ENTER to complete a substitution."
2920 PRINT "    Any key and ENTER to skip the substitution.": COLOR 7,0
2930 I8=I9 : F3=0
2940 FOR I=1 TO I8
2950 M=1
2960 P=INSTR(M,TS(I),L1$)
2970 IF P=0 GOTO 3160 ELSE F3=1: NL=LEN(SIN$(I))+3
2980 PRINT :PRINT I;": "TS(I) : LTI=LEN(TS(I)) : C1=1
2990 IF LTI+NL<80 THEN GOTO 3010
3000 IF LTI+NL=80 THEN C1=2 : GOTO 3010 ELSE NL=0
3007 IF LTI>79 THEN C1=2
3010 LIN = C$RLIN-C1 : LOCATE LIN,P+NL
3020 COLOR 0,7: PRINT L1$ : COLOR 7,0
3030 INPUT "    Make Substitution ";A$
3040 IF A$>" " THEN L2=1: GOTO 3140
3050 B$=LEFT$(TS(I),P-1)
3060 E$=MID$(TS(I),P,L)
3070 TS(I)=B$+L2$+E$
3080 IF LEN(TS(I))<80 THEN GOTO 3140 ELSE PD=LEN(TS(I))-2: I1=I+1
3090 A1$=MID$(TS(I),PD,1)
3100 IF A1$=" " OR A1$="," OR A1$="." THEN GOTO 3110 ELSE PD=PD-1: GOTO 3090
3110 IF PD>79 THEN PD=PD-1 : GOTO 3090
3120 T1$=SPACES(20)+MID$(TS(I),PD+1) : TS(I)=LEFT$(TS(I),PD) 'New split lines
3130 GOSUB 2440
3140 W=P+L2
3150 GOTO 2960
3160 NEXT I
3170 IF F3=0 THEN PRINT L1$;" NOT FOUND..."
3180 SOUND 1046.3,2: SOUND 784,2
3190 COLOR 15,0:PRINT SPC(9);"***Substitution Complete***":COLOR 7,0: PRINT
3200 GOTO 2820
3210 BEEP:PRINT:COLOR 15,0:PRINT SPC(9);"DON'T FORGET TO SAVE MODULE...": COLOR 7,0
3220 RETURN
3230 REM *****
3240 PRINT :COLOR 15,0 : PRINT SPC(10);"    --- Save Module ---": COLOR 7,0:PRINT
3250 I4=I5-1 : I6=I5+1 : I7=I9-2 : I8=I9-1 : I0=I9+2
3260 CLS: PRINT "    ----- ERROR CHECKING ROUTINE -----":PRINT
3270 FOR I1=2 TO 5
3280 IF INSTR(TS(I1),"COMMENT")=0 GOTO 3300
3290 NEXT
3300 IF INSTR(TS(I1),"G$")>0 OR TS(I1)="000" GOTO 3320
3310 PRINT "TOP GATE FAULT..." :PRINT I1;TS(I1) :PRINT

```

Table A.1 (Continued)
Program Listing of MODEDIT

```

3320 FOR I=11 TO 14
3330 L$=TS(I) : X1=CSRLIN
3340 IF L$="000" THEN GOTO 3310 ELSE IF LEN(L$)>0 GOTO 3380
3350 PRINT "LINE ";I;" IS BLANK (It has been deleted)." : TS(I)="000" : GOTO 3310
3360 IF INSTR(L$,"")>0 OR INSTR(L$,".")>0 GOTO 3380
3370 PRINT "Line is missing delimiter(s) or contains only blanks..."
3380 IF INSTR(L$,"SGS")>0 THEN PRINT "WARNING: Module contains a Special Gate."
3390 IF LEN(L$)>80 THEN PRINT "Line exceeds 80 chrs..."
3400 P=INSTR(L$,"COMP") : IF P=0 GOTO 3420
3410 PRINT "Unsubstituted 'COMP'..."
3420 P=INSTR(L$,"YP") : IF P=0 GOTO 3440
3430 PRINT "Possible bad 'YP' substitution..."
3440 P=INSTR(L$,"GS") : P=P+INSTR(L$,"ES")
3450 IF P=0 THEN GOTO 3500 ELSE P=4
3460 IF MID$(L$,P,1) = " " THEN P=P+1 : GOTO 3460
3470 PP=INSTR(L$,".")
3480 IF MID$(L$,PP-1,1) = " " THEN PP=PP-1 : GOTO 3480
3490 IF PP-P > 16 THEN PRINT "Gate/Event name is longer than 16 chrs..."
3500 X2=CSRLIN : IF X2>X1 THEN PRINT I;L$ : PRINT
3510 NEXT
3520 FOR I=16 TO 17
3530 L$=TS(I) : X1=CSRLIN
3540 IF L$="000" THEN GOTO 3640 ELSE IF LEN(L$)>0 GOTO 3560
3550 PRINT "LINE ";I;" IS BLANK (It has been deleted)." : TS(I)="000" : GOTO 3540
3560 IF INSTR(L$,"S")>0 GOTO 3580
3570 PRINT "Text line contains only blanks or is fragmented (no terminating 'S')..."
3580 IF LEN(L$)>80 THEN PRINT "Line exceeds 80 chrs..."
3590 P=INSTR(L$,"COMP") : IF P=0 GOTO 3630
3600 IF MID$(L$,P,8)="COMPONENT" GOTO 3630
3610 IF MID$(L$,P,5) = "COMPLETE" GOTO 3630
3620 PRINT "Unsubstituted 'COMP'..."
3630 X2=CSRLIN : IF X2>X1 THEN PRINT I;L$ : PRINT
3640 NEXT
3650 IF TS(1)<>"ENDS" THEN PRINT "Missing 'ENDS'..." : PRINT I;TS(I) : PRINT
3660 IF TS(1+1)<>"FINS" THEN PRINT "Missing 'FINS'..." : PRINT I;TS(I) : PRINT
3670 PRINT:COLOR 15,0:PRINT SP(12);"*** ERROR CHECKING COMPLETE ***":COLOR 7,0
3680 REM ----- Create SETS File -----
3690 IF D$="" THEN D$=D1$: D2$=D1$
3700 COLOR 7,0: PRINT :PRINT :PRINT "      'DF' defaults to the following..."
3710 PRINT "      Disk drive for SETS module storage : ";D$
3720 PRINT "      Module file name : ";M$(1)
3730 PRINT "      Module fault tree name : ";M$(1)
3740 LIN = CSRLIN-3
3750 LOCATE LIN,1 :COLOR 15,0: INPUT "Enter disk drive for SETS module storage (END to return to menu):";D3$
3760 I=ASC(D3$) :PRINT: PRINT: PRINT: PRINT
3770 IF D3$="END" GOTO 4110
3780 IF D3$="DF" THEN S1$=M$(1) : A$=S1$ ELSE D3$=D3$
3790 I=ASC(D3$) :PRINT: PRINT: PRINT: PRINT
3800 IF I<65 OR I>72 THEN PRINT "***INCORRECT DRIVE FORMAT**":GOTO 3700
3810 CHDIR D3$+"\SETS" : GOTO 3860
3820 PRINT "This disk DOES NOT contain the SETS directory..."
3830 INPUT "Do you want to create it (Y or N): ";Q$
3840 IF Q$="N" GOTO 3700
3850 MKDIR D3$+"\SETS" : CHDIR D3$+"\SETS"
3860 IF D3$="DF" GOTO 3900
3870 PRINT "Enter file name under which module will be stored"
3880 PRINT "      (ENTER defaults to ";M$(1);"): ";
3890 INPUT S1$ : IF S1$="" THEN S1$=M$(1)
3900 S2$=D3$+"\S1$
3910 OPEN S2$ FOR INPUT AS #1 ' Check for duplicate file.
3920 BEEP: PRINT "      ***A file already exists on Disk ";D3$;" under this name..."

```

Table A.1 (Continued)
Program Listing of MODEIT

```

3930 CLOSE #1: BEEP: INPUT "    Do you want to write over it (Y or N)";Q$
3940 IF Q$="N" THEN GOTO 490
3950 IF Q$="DF" GOTO 3990
3960 PRINT :PRINT "Enter a fault tree name (ENTER defaults to ";S1$;"")
3970 INPUT "    Note: This is NOT the filename for the module. ";A$
3980 IF A$="" THEN A$=S1$: GOTO 4010
3990 IF LEN(A$)>16 THEN PRINT "    ***TOO LONG -16 char. maximum":GOTO 3950
4000 IF MID$(A$,1,5)="FAULT" THEN PRINT "    ***INCORRECT FORMAT (Omit 'FAULTREE' from name)": GOTO 3950
4010 C1$="FAULTREE$"+A$+"."
4020 C2$="FAULTREE$"+A$+"S"
4030 T$(1)=C1$: T$(15)=C2$
4040 OPEN S2$ FOR OUTPUT AS #2
4050 FOR I=1 TO 19
4060 IF T$(I)="" GOTO 4080
4070 PRINT #2,T$(I)
4080 NEXT I
4090 CLOSE #2 : COLOR 15,0: PRINT
4100 PRINT SPC(7);"*** Module Saved in SETS Directory as File ";S1$ :COLOR 7,0
4110 RETURN
4120 REM *****
4130 REM FUNCTION KEY 1 --Module Listing
4140 PRINT :COLOR 15,0 :PRINT "--Current Listing of Module ";MS(1): COLOR 7,0
4150 FOR J5=1 TO 19
4160 IF T$(J5)="" GOTO 4180
4170 PRINT J5; T$(J5)
4180 IF J5<>21 GOTO 4210
4190 PRINT "    ...Press any key to continue:"
4200 A$=INKEY$: IF A$="" THEN 4200
4210 NEXT J5
4220 SEL=99
4230 RETURN
4240 REM *****
4250 REM Error trapping routine
4260 IF ERR<>53 GOTO 4290
4270 IF ERR=3910 THEN RESUME 3950
4280 SOUND 220,16: PRINT "****FILE NOT FOUND (on this disk)****": RESUME 160
4290 IF ERR<>64 AND ERR<>52 AND ERR<>75 GOTO 4310
4300 SOUND 164,8,20: PRINT: PRINT "    *** ILLEGAL FILENAME ***": RESUME 480
4310 IF ERR<>61 GOTO 4330 ELSE SOUND 220!,3!: SOUND 130.5,6!
4320 PRINT "    *** DISK IS FULL ***": RESUME 480
4330 IF ERR<>62 THEN GOTO 4350 ELSE SOUND 1568!,8!: SOUND 1780!,6!: COLOR 15,0
4340 PRINT "**** WARNING: File does not end with 'PINS' ":COLOR 7,0: RESUME 360
4350 IF ERR<>67 GOTO 4370 ELSE PRINT
4360 PRINT "    *** ILLEGAL FILENAME ***": RESUME 480
4370 IF ERR<>70 GOTO 4400
4380 PRINT "**** DISK IS WRITE PROTECTED ****"
4390 RESUME 3700
4400 IF ERR<>76 GOTO 4460
4410 IF ERR<260 THEN RESUME NEXT
4420 IF ERR=260 THEN CHR$(13): RESUME NEXT
4430 IF ERR=290 THEN CHR$(13): RESUME
4440 SOUND 1907,16 :PRINT "UNABLE TO STORE FILE:"
4450 RESUME 3820
4460 SOUND 196,12 : PRINT "    ***** FATAL ERROR *****"
4470 PRINT "    Enter 'SYSTEM' if you wish to exit BASIC.": SOUND 145,8,20
4480 SOUND 130,8,20: FOR I=1 TO 6000 : NEXT
4490 ON ERROR GOTO 0
4500 CHR$(13):END
4510 REM *****
4520 CHR$(13) : SYSTEM
4530 END

```

A.5 A Sample MODEDIT Session:

Disk drive containing module to edit:

User may enter a single letter (A through H) to indicate the drive containing the module to be loaded into memory.

Enter name of module to edit:

This is the file name of either a generic Appendix I module (e.g., I4.MOD, I117.MOD, etc.) or a previously edited and stored module (e.g., AWF1.MOD). The generic modules all have file names corresponding to their Appendix I page numbers (e.g., I4.MOD is the file name for the module that appears on page I4 of Appendix I). These modules are stored in three directories (THRU50, THRU100, and THRU200) on the Fault Tree Modules disk. MODEDIT automatically directs the search to the correct directory based on the Appendix I page number entered. All modules previously edited and saved by the user are automatically stored in a SETS directory by MODEDIT. Hence, if the file name entered by the user does not begin with "I," MODEDIT directs the search to the SETS directory on the disk drive specified. If the file is not found in any directory (or the directory does not exist), a message is displayed and the user may enter a new drive specifier and file name. All modules to be edited with MODEDIT must have the gates defined first and the events defined last.

If the requested file module is found, it is immediately loaded (into memory) and the following is displayed:

COMMENT\$MODULE DESCRIPTIVE TEXT\$

PRESSING "F1" (ENTER) IN RESPONSE TO ANY QUESTION GENERATES A CURRENT LISTING OF THE MODULE AND RETURNS TO THE QUESTION MENU:

1. DELETE GATES/EVENTS	4. WORD SUBSTITUTIONS
2. DELETE SINGLE LINES	5. SAVE MODULE
3. ADD GATES/EVENTS	6. EDIT NEW MODULE OR QUIT

Enter # of selection:

If the module contains any descriptive comment lines at the beginning (as is the case with all generic Appendix I modules) they are displayed. The next message on the screen indicates that the user can list the lines of the module code by pressing the F1 function key (and ENTER) in response to any program question. MODEDIT generates line numbers for the listing. When the listing is complete, the original question is redisplayed and the user may respond.

The user may select any of the menu options by entering the number. In editing a generic Appendix I module, it is recommended that one first delete gates/events not needed (Op. 1), then do the necessary substitutions (Op. 4), then if necessary add any gates/events using Option 3. This prevents needless word substitutions and helps avert problems that can sometimes arise in the complex process of adding gates or events. Finally, the user must remember to save the edited module on disk (Op. 5).

A.6 Delete Gates/Events

This menu option lets the user select which of the gates or events in a module he wants to delete. One at a time the gate definition lines of coding and their associated descriptive text are displayed on the screen. The user may either enter "D" to delete a given gate or simply press ENTER to retain it. If a gate is to be deleted, MOEDIT deletes both the indicated gate and all inputs to the gate. If the deleted gate has another gate as input, the inputs to the lower gate are also deleted (etc.). MOEDIT also removes the descriptive text associated with each gate or event deleted. Any screen listing of the module after gates or events are deleted will show blank lines where the code input lines existed previously. These lines will be removed when the module is saved.

A.7 Delete Single Lines

This mode of deletion is used less often than option 1. Its primary use is to remove stray lines from modules that have had gates or events added or modules edited with some other editor/word processor. The single line delete does just that. There is no search for associated descriptive text or inputs to gates if the deleted line is a gate/event definition.

A.8 Add Gates/Events

This option currently works well for adding single events to existing module gates and not so well for building more complex structuring involving several gates. The user is first asked to define the event or gate in terms of SETS input. Proper SETS syntax must be followed (e.g. AG\$ GATE 7. IN\$ EVNT4,EVNT8. OUT\$GATE 6.)

The user may alternatively choose to copy an existing event/gate definition by entering COPY. He will then be prompted for the line number of the definition to be copied (Press F1 for a listing with line numbers). Regardless of how the user chooses to define an event or gate, he may continue the definition on another line by ending the current line with a ",".

Example:

```
OG$ GATE8. IN$EVNT5,GATE 9,  
Continue definition . . .  
(Enter COPY to copy an existing definition, "END" to  
terminate input)  
EVNT6,EVENT12. OUT$GATE4.
```

A maximum of 80 characters per line is permitted. MODEIT automatically attempts to insert the gate or event name into the input list of the gate(s) the user indicates after OUT\$. If the output gate cannot be found within the modules, the user is prompted to go back and create the higher level gate first. If MODEIT succeeds in locating the output gate definition and the name of the event/gate being created is already present in its input list, the user is asked if he wishes to add the name to the list again. This case may arise when copying an existing event definition; MODEIT will go to insert the newly created event name into the input list of the gate given as an output and find that the name already appears there. In this case, the user will generally choose to add the name again, and later go back and change one of the gate input list names using the "Word Substitution" option. MODEIT adds new gate definitions after the last gate currently defined and new events after the last gate defined. Once the definition is completely entered the user is prompted for the descriptive label associated with it (i.e., the test that is to appear in the event/gate blocks of fault tree plots). Syntax must follow that required by the FTD (Fault Tree Draw) program. MODEIT adds the descriptive label to the block of text labels after the module gate/event definitions.

When adding complex gate structures with more than 1 or 2 gates, it is highly recommended that at most 2 gates be added and the resulting module saved. Once it has been saved, recall the module and repeat the above process until all gates have been added. Massive changes done in one editing session lead to file errors which can be avoided and less time is ultimately required if the file is saved at several intermediate points.

A.9 Word Substitutions

This is the routine used to change the "generic" system and component names that appear in Appendix I modules to the plant specific names needed to relate the fault tree to the system being modeled. The user is prompted to enter the label (word) to be replaced and then the substitution for this word. Entering "END" in response to either of these prompts terminates the Word Substitution routine. The label to be replaced may be several words long and contain embedded delimiters and blanks. It is best to substitute words that are shorter than or, at most, equal

to the length of the labels being substituted in the generic file.

Once provided with a "word" to be replaced, MODEDIT searches for the first occurrence of this string within the module. The search string will be exactly as entered including letter case and any trailing or leading blanks.** When the first occurrence is located, the line and line number are displayed and the string is highlighted in reverse video. The user may choose to complete the substitution by pressing ENTER or skip the substitution by pressing any character key followed by ENTER. MODEDIT then searches for the next occurrence and displays it. Note that the label to be replaced may occur more than once in a given line. In that case, the line will be displayed once for each substitution with the reverse video highlight pointing out the location of the string to be replaced. Best results and most efficient operation are realized when the longest words are substituted first since some of the shorter generic words are also used within the larger substitutable words.

The Word Substitution routine may also be used to make changes in the modules structure (e.g., replace BE\$EVTN8. OUT\$GATE4. with OG\$GATE12. IN\$ EVNT2, EVNT3. OUT\$ GATE 3). However, this is not a very efficient method compared to using another editor.

A.10 Save Module

Once the module editing process is complete, the user must remember to save the module as a file on disk. When the "Save Module" option is selected, a subroutine first performs a number of error checks on the module SETS input. The following checks are made:

- Correct location of top gate definition (immediately after any COMMENT lines at beginning of module).
- Gate/event definition lines with no delimiters ("," or ".") or containing all blanks.
- Check for "special gates." This is a WARNING to remind the user that a Boolean gate definition must be provided or the special gate changed to an AND or OR gate.
- Lines exceeding 80 characters.

** The only exception to this exact searching are the labels COMP and TYP that commonly appear in Appendix I modules. The Word Substitution routine automatically adds a blank to COMP and a "-" to TYP (-TYP) in order to prevent unwanted substitutions in text words like "COMPONENT" and TYPE.

- Gate/event definitions and event descriptions with "COMP" in them. (COMPONENT is okay.) This probably indicates a substitution has not been made.
- Gate/event definitions with "YP" in them. This may indicate an incorrect substitution that changed the "-T" in -TYP to some other character (and hence a subsequent substitution for TYP did not work properly).
- Gate/event names that exceed 16 characters.
- Missing END\$ or FIN\$ lines at the end of the module.

The error checking routine displays a short description of any detected errors or possible errors, and then displays the line number and line in which the error occurs. If there are many errors (e.g., a dozen unsubstituted "COMP's in the module), all lines containing errors may not be displayed. The only module changes made by the error checking subroutine is that blank lines are deleted.

When the error checking is complete, the user is prompted for a disk drive on which to store the module file. (By entering END the user may return to the editing menu to correct any module errors.) Default file storage options are displayed and include:

- Disk drive for module storage (the drive used to store previous modules)
- Module file name (the same name used to load the module)
- Module fault tree name (the name appearing in the first line of the module, e.g., FAULT TREE \$FTNAME)

The user may accept the default options by entering "DF." However, if generic Appendix I modules are being edited for the first time, the user will want to rename the edited modules and will probably want to store them on a different disk. By entering a valid disk drive instead of "DF," he will be prompted for a file name* (a module page number assigned by the fault tree analyst is recommended) and a fault tree name. The fault tree name serves only to identify a module internally and will be stripped out of the SETS input when the modules of a system fault tree are merged by SETS. Hence the name used is generally not of much importance here, and using the same name as the file name is suggested.

*NOTE: It is recommended that the file name not include an extension (e.g., AFW1 instead of AFW1.EXT) since the INDEX and INDEX2 programs will not work if extensions are added to the file name.

The module save routine attempts to store all modules in a SETS directory on the disk drive indicated. If no SETS directory exists, the user is asked if one should be created. A "no" response returns the user to the main menu. Before storing a module in the SETS directory, the directory is searched to ensure the user has selected a unique file name. If a file with that name already exists, the user has the option to over- write it or to enter a new file name for the module.

A.11 Edit New Module or QUIT

This option is selected once a module has been successfully saved to a disk or when the user wishes to exit MODEDIT. In either case, the user is prompted for the name of the new module to edit. If "Q" is entered, the program is terminated and control is returned to DOS. For any other entry, MODEDIT attempts to locate a file by that name on the same drive used to load the previous module. If it cannot be located, the user is prompted to enter a new drive designator and file name.

REFERENCE

1. Varnado, G. B., W. H. Horton, and P. R. Lobner, Modular Fault Tree Analysis Procedures Guide, NUREG/CR-3268, SAND83-0963, (4 volumes), Sandia National Laboratories, Albuquerque, NM, August 1983.

APPENDIX B

INDEX DOCUMENTATION INDEX AND INDEX2

B.1 Overview

The INDEX programs (see Tables B.1 and B.2 for a listing of the source code for the INDEX programs) perform checks for errors related to module linkage and are generally run once all the modules for a given fault tree have been edited and stored. By flagging these errors for correction before SETS attempts to operate on the modules, expensive and time-consuming aborts on the mainframe can be greatly reduced. The INDEX programs can also generate an index listing of all modules being processed indicating module file name, fault tree name, date created and module top event description.

B.2 Operation

If all modules for a given fault tree have been stored on a single disk (stored as files with no file name extensions), the INDEX program (Table B.1) is adequate to perform the error checking and index listing. If the fault tree modules are spread across two or more disks, INDEX2 (Table B.2) should be used to check for linkage errors across two disks at a time. In all other respects, the functions performed by the two programs are identical. In order to run INDEX on a single disk, the INDEX.BAS, INDEX.BAT (see Table B.3 for a listing of the batch file commands), SORT.EXE (a DOS program), and BASICA.COM files must be copied to the disk. The user initiates the program by entering:

INDEX [SORT/NOSORT/RESUME]

The SORT/NOSORT options are used to indicate whether or not the modules should be sorted by file name prior to listing and processing. The RESUME option is used to resume error checking with the last module processed if the program was interrupted for some reason (e.g., printer jams, manual termination, etc.). The INDEX program generates a file on disk containing a listing of modules, called FILELIST. If the INDEX program is run more than once on a given disk without the RESUME options, this FILELIST file must be deleted or renamed before the second attempt.

In order to run INDEX2 on two disks, the INDEX2.BAS, INDEX2.BAT (see Table B.4 for a listing of the batch file commands), and BASICA.COM files must be copied onto one disk and that disk should be inserted into default drive A. The second disk to be co-processed should be inserted in disk drive B. The user

initiates the program by entering:

INDEX2 [SORT/NOSORT/RESUME]

where the options have the same functions as described for INDEX above.

Both INDEX and INDEX2 prompt the user to select one of three output options. The programs read each module (NOTE: Module should be stored in a file with no file name extension e.g., AFW1 instead of AFW1.EXT.) off the disk(s) first and store the names of all top events, developed events, and transfers. While this process is underway, an index of the modules appears on the CRT screen for all options. Similarly, the results of error checking are displayed on the screen for all options. The user may choose to route all output to the printer, only the error checking to the printer, or no output to the printer. The program was written for an EPSON-type printer.

Table B.1
Program Listing of INDEX

```

1 REM *****
2 REM                               INDEX CODE - JUNE 1968
3 REM *****
4 REM
5 REM                               ISSUED BY SANDIA NATIONAL LABORATORIES
6 REM                               A PRIME CONTRACTOR TO THE
7 REM                               UNITED STATES DEPARTMENT OF ENERGY
8 REM
9 REM ***** N O T I C E *****
10 REM
11 REM THIS CODE WAS PREPARED IN THE COURSE OF WORK SPONSORED BY THE
12 REM UNITED STATES GOVERNMENT. NEITHER THE UNITED STATES NOR
13 REM THE UNITED STATES DEPARTMENT OF ENERGY, THE UNITED STATES
14 REM NUCLEAR REGULATORY COMMISSION, NOR ANY OF THEIR EMPLOYEES,
15 REM NOR ANY OF THEIR CONTRACTORS, SUBCONTRACTORS, OR THEIR
16 REM EMPLOYEES, MAKES ANY WARRANTY, EXPRESSED OR IMPLIED, OR
17 REM ASSUMES ANY LEGAL LIABILITY OR RESPONSIBILITY FOR THE
18 REM ACCURACY, COMPLETENESS OR USEFULNESS OF ANY INFORMATION,
19 REM APPARATUS, PRODUCT OR PROCESS DISCLOSED, OR REPRESENTS THAT
20 REM ITS USE WOULD NOT INFRINGE PRIVATELY OWNED RIGHTS.
21 REM
22 REM *****
23 REM
24 REM AUTHOR: T.L. ZIMMERMAN, FORMALLY OF ENERGY INCORPORATED
25 REM                               ALBUQUERQUE, NM
26 REM
27 REM FOR INFORMATION CONTACT:
28 REM
29 REM A. C. PAYNE, JR.             PHONE: (505) 846-3568
30 REM D. W. WHITEHEAD             PHONE: (505) 844-2632
31 REM
32 REM REACTOR SYSTEMS SAFETY ANALYSIS DIVISION 5412
33 REM SANDIA NATIONAL LABORATORIES, ALBUQUERQUE, NM 87185
34 REM
35 REM *****
40 SCREEN 0,0,0  'To run on either color or mono screens.
45 PRINT "      --- INDEX ---"; PRINT
50 REM This routine prints an index of and performs error checking on
60 REM all SETS files (modules) residing on a given disk...
70 DEFINT P,I,J,K,L,N,P
80 WIDTH "LET1:",160 : CENDIR "\"
90 DIM IS(300,2),ES(50),TOPS(300,2),DES(1000,2)
100 ON ERROR GOTO 1840
110 SOUND 440!,4! :SOUND 37!,2! :SOUND 440!,4! :SOUND 880!,37!
120 PRINT :COLOR 15,0: PRINT "OUTPUT OPTIONS:" :COLOR 7,0
130 PRINT "  1. Complete output/Routed to printer.
140 PRINT "  2. No printed index list/Route all checks to printer."
150 PRINT "  3. Complete output/No routing to printer.
160 PRINT :INPUT "Enter Selection: ";N7
170 IF N7=1 THEN Q$="Y": Q1$="Y": GOTO 210
180 IF N7=2 THEN Q$="Y": Q1$="N": GOTO 210
190 IF N7=3 THEN Q$="N": Q1$="N": GOTO 210
200 N7=1 : GOTO 170
210 OPEN "FILELIST" FOR INPUT AS #3
220 FOR I=1 TO 300
230 IF EOF(3) GOTO 260
240 LINE INPUT #3,T$(I,1)
250 NEXT
260 CLOSE #3
270 N=I-1 : F=6 : F1=6
280 IF INSTR(T$(2,1),"Volume")=0 GOTO 310

```

Table B.1 (Continued)
Program Listing of INDEX

```

280 N=1-2: F=7: F1=5 'Unsorted FILELIST...
300 IF Q18="Y" THEN LPRINT CHR$(12);CHR$(14);"Unsorted Index:" ;GOTO 320
310 IF Q19="Y" THEN LPRINT CHR$(12);CHR$(14);"Sorted Index:"
320 IF LEFT$(TS(N,1),2)<>".." GOTO 360
330 SOUND 248,8,20 : SOUND 987,8,20 : FOR J=1 TO 3000 : NEXT
340 PRINT :PRINT "*** THERE ARE NO SETS FILES IN DIRECTORY 'SETS' ***"
350 GOTO 2020
360 FOR I=F TO N
370 FILES=LEFT$(TS(I,1),8)
380 TMS=MID$(TS(I,1),24,16)
390 TS(I,1)=FILES : TS(I,2)=TMS
400 NEXT
410 IF Q18="Y" THEN GOSUB 1040
420 DIM DEL(100) : DEL(1)=99 : I2=0 : K=0 : K=0
430 CHDIR "\SETS"
440 FOR I=F TO N
450 S3$=TS(I,1)
460 IF K>52 THEN K=0: GOSUB 1030
470 OPEN S3$ FOR INPUT AS #1
480 LINE INPUT #1,ES(1)
490 P1=INSTR(ES(1),"S")+1 : P2=INSTR(P1,ES(1),",")
500 IF P1=1 OR P2=0 THEN FTMS="?????" : GOTO 520
510 FTMS=MID$(ES(1),P1,P2-P1)
520 IF EOF(1) GOTO 710
530 LINE INPUT #1,ES(2)
540 P3=INSTR(ES(2),"GS")+2 : IF P3=2 GOTO 520 'Find the top gate...
550 B9=MID$(ES(2),P3,1) :IF B9=" " THEN P3=P3+1 :GOTO 550
560 P4=INSTR(P3,ES(2),",")-1
570 B9=MID$(ES(2),P4,1) :IF B9=" " THEN P4=P4-1: GOTO 570
580 TOP$(I,1)=MID$(ES(2),P3,P4-P3+1) :TOP$(I,2)=FTMS 'Save top gate & tree name
590 IF EOF(1) GOTO 740
600 LINE INPUT #1,ES(1)
610 P3=INSTR(ES(1),"DES")+3 : IF P3>3 GOTO 630 'Find any developed events...
620 P3=INSTR(ES(1),"TES")+3 : IF P3>3 GOTO 680
630 B5=MID$(ES(1),P3,1) :IF B5=" " THEN P3=P3+1 :GOTO 630
640 P4=INSTR(P3,ES(1),",")-1
650 B5=MID$(ES(1),P4,1) :IF B5=" " THEN P4=P4-1: GOTO 650
660 L9=L9+1
670 DES(L9,1)=MID$(ES(1),P3,P4-P3+1) :DES(L9,2)=FTMS 'Save developed event name
680 IF INSTR(ES(1),"TREES")>0 GOTO 780
690 IF INSTR(ES(1),"FAULT TREE")>0 GOTO 780
700 GOTO 590
710 BEEP: PRINT "*** Incomplete Module: ";TS(I,1)
720 IF Q18="Y" THEN LPRINT "*** Incomplete Module: ";TS(I,1): LPRINT: K=K+1
730 CLOSE #1 : GOTO 990
740 BEEP: PRINT "*** Second 'FAULTTREES' delimiter not found for module ";TS(I,1)
750 IF Q18="N" GOTO 770
760 LPRINT: LPRINT "***Second 'FAULTTREES' delimiter not found for module ";TS(I,1) :K=K+2
770 ES(1)="": IJ=1: GOTO 840
780 IJ=1
790 LINE INPUT #1,ES(IJ)
800 Z$=RIGHT$(ES(IJ),1) :IF Z$="S" GOTO 840
810 IF Z$<>" " THEN GOTO 830 ELSE L2=LEN(ES(IJ))-1
820 ES(IJ)=LEFT$(ES(IJ),L2) : GOTO 800
830 IJ=IJ+1 : GOTO 790
840 CLOSE #1
850 PRINT :PRINT "Filename: ";TS(I,1): PRINT "Fault Tree Name: ";FTMS
860 PRINT "Created on: ";TS(I,2)
870 PRINT "Fault Tree (Module) Top Event:"
880 FOR IK=1 TO IJ
890 PRINT SPC(3);ES(IK)
900 NEXT IK
910 K9=K9+1
920 IF Q18="N" GOTO 990
930 LPRINT: IK=1

```

Table B.1 (Continued)
Program Listing of INDEX

```

940 LPRINT USING "\          \";TS(I,1);FTNS;
950 LPRINT TS(I,2);SPC(5);ES(1) :K=K+2
960 IF IK=IJ GOTO 990
970 IK=IK+1 :K=K+1
980 LPRINT SPC(40);ES(IK) : GOTO 960
990 NEXT I
1000 PRINT :PRINT "   ---";K9;" Files/Modules in Index ---"
1010 IF Q19="Y" THEN LPRINT :LPRINT "   ---";K9;" Files/Modules in Index ---"
1020 GOTO 1090
1030 LPRINT CHR$(12)
1040 LPRINT CHR$(27);"4";
1050 LPRINT:LPRINT"Filename      Fault Tree Name      Created on          Fault Tree (Module) Top Event"
1060 LPRINT CHR$(27);"5"
1070 RETURN
1080 REM ***** MODULE ERROR CHECKING ROUTINES *****
1090 BEEP: PRINT :COLOR 15,0 :PRINT "Duplicate Top Gates:" :PRINT: COLOR 7,0
1100 IF Q8="N" GOTO 1120
1110 BEEP: LPRINT CHR$(12);CHR$(14);"Duplicate Top Gates:" :LPRINT
1120 K=0: BEEP
1130 FOR J1=P TO N
1140 GATES=TOP$(J1,1) :FL=0
1150 IF INSTR(GATES,"*")>0 GOTO 1200
1160 TOP$(J1,1)=TOP$(J1,1)+"*"
1170 FOR J2=P TO N
1180 IF TOP$(J2,1)=GATES THEN GOSUB 1250
1190 NEXT J2
1200 NEXT J1
1210 IF K>0 GOTO 1340
1220 PRINT "          NONE"
1230 IF Q8="Y" THEN LPRINT"          NONE"
1240 GOTO 1340
1250 FL=FL+1 : K=1 : IF FL>1 GOTO 1310
1260 PRINT "GATE: ";GATES
1270 PRINT "      Module:" :PRINT "          ";TOP$(J1,2)
1280 IF Q8="N" GOTO 1310
1290 LPRINT: LPRINT "GATE: ";GATES
1300 LPRINT "      Module:" :LPRINT "          ";TOP$(J1,2)
1310 PRINT "          ";TOP$(J2,2)
1320 IF Q8="Y" THEN LPRINT"          ";TOP$(J2,2)
1330 RETURN
1340 FOR J1=P TO N 'Strip top mark from gates array...
1350 P1=LEN(TOP$(J1,1))-1
1360 TOP$(J1,1)=LEFT$(TOP$(J1,1),P1)
1370 NEXT
1380 PRINT:COLOR 15,0:PRINT "   --- MODULE LINKAGE CHECK ---":PRINT:COLOR 7,0
1390 IF Q8="N" GOTO 1410
1400 LPRINT CHR$(12);CHR$(14);"   --- Module Linkage Check ---":LPRINT
1410 FOR I=1 TO L9
1420 DEVS=DES(I,1) : FL=0 : L8=0
1430 IF Q28="N" GOTO 1440
1440 PRINT :PRINT "Dev Event: ";DEVS;"      Module: ";DES(I,2)
1450 IF Q8="Y" THEN LPRINT :LPRINT "DE: ";DEVS;"      Module: ";DES(I,2)
1460 FOR J=P TO N
1470 IF TOP$(J,1)=DEVS THEN GOSUB 1570
1480 NEXT J
1490 IF FL>0 GOTO 1530
1500 PRINT "****NO MODULE FOUND CORRESPONDING TO DE: ";DEVS;"      -Module: ";DES(I,2)
1510 IF Q8="N" GOTO 1530
1520 LPRINT "****NO MODULE FOUND CORRESPONDING TO DE: ";DEVS;"      -Module: ";DES(I,2)
1530 NEXT I
1540 PRINT "   *** COMPLETED ***"

```

Table B.1 (Concluded)
Program Listing of INDEX

```

1550 IF Q$="Y" THEN LPRINT:LPRINT "      *** COMPLETED ***"
1560 GOTO 1620
1570 FL=FL+1 : IF Q2$="N" GOTO 1610
1580 PRINT "      Linked to Module: ";TOP$(J,2)
1590 IF Q$="Y" THEN LPRINT "      Link to Module: ";TOP$(J,2)
1600 TOP$(J,2)=TOP$(J,2)+*****
1610 RETURN
1620 COLOR 15,0: PRINT :PRINT "Unlinked Top Gates:" :COLOR 7,0
1630 IF Q$="Y" THEN LPRINT CHR$(12);CHR$(14);"Unlinked Top Gates:"
1640 K=0
1650 FOR J=F TO H
1660 IF INSTR(TOP$(J,2),*****)=0 THEN GOSUB 1720
1670 NEXT J
1680 IF K>0 GOTO 1760
1690 PRINT :PRINT "      NONE"
1700 IF Q$="Y" THEN LPRINT :LPRINT "      NONE"
1710 GOTO 1760
1720 K=K+1
1730 PRINT "Module: ";TOP$(J,2); "      Top Gate: ";TOP$(J,1)
1740 IF Q$="Y" THEN LPRINT:LPRINT "Module: ";TOP$(J,2); "      Top Gate: ";TOP$(J,1)
1750 RETURN
1760 PRINT :COLOR 15,0: PRINT "      --- ALL CHECKS COMPLETED ---": COLOR 7,0
1770 IF Q$="N" GOTO 1790
1780 LPRINT :LPRINT:LPRINT CHR$(14); "      --- ALL CHECKS COMPLETED ---"
1790 SOUND 261.6,9: SOUND 523.2,9: SOUND 1046.5,9:
1800 FOR J=1 TO 3000 : NEXT
1810 GOTO 2020
1820 REM *****
1830 REM Error trapping routine
1840 IF ERR<>53 GOTO 1830
1850 IF ERR<>170 GOTO 1890
1860 SOUND 174.6,40 : FOR JL=1 TO 3000 : NEXT
1870 COLOR 15: PRINT :PRINT "      *** 'FILELIST' NOT FOUND ON DISK ***"
1880 GOTO 2020
1890 SOUND 220,16: PRINT "****FILE ";S3$;" NOT FOUND (on this disk)****"
1900 IF Q$="N" GOTO 1920 ELSE K=K+2
1910 LPRINT; LPRINT CHR$(14);"FILE ";S3$;" NOT FOUND ON DISK ****": LPRINT
1920 RESUME 990
1930 IF ERR<>75 GOTO 1980
1940 PRINT "This disk DOES NOT contain the SETS directory."
1950 CHDIR "\": RESUME NEXT
1960 SOUND 196,12 : PRINT "      ***** FATAL ERROR *****"
1970 PRINT "      Enter 'SYSTEM' if you wish to exit BASIC.": SOUND 146.8,20
1980 SOUND 130.8,20: FOR IL=1 TO 6000 : NEXT
1990 ON ERROR GOTO 0
2000 CHDIR "\": END
2010 REM *****
2020 CHDIR "\": SYSTEM
2030 END

```


Table B.2
Program Listing for INDEX2

```

1 REM *****
2 REM                               INDEX2 CODE - JUNE 1988
3 REM *****
4 REM
5 REM                               ISSUED BY SANDIA NATIONAL LABORATORIES
6 REM                               A PRIME CONTRACTOR TO THE
7 REM                               UNITED STATES DEPARTMENT OF ENERGY
8 REM
9 REM ***** NOTICE *****
10 REM
11 REM THIS CODE WAS PREPARED IN THE COURSE OF WORK SPONSORED BY THE
12 REM UNITED STATES GOVERNMENT. NEITHER THE UNITED STATES NOR
13 REM THE UNITED STATES DEPARTMENT OF ENERGY, THE UNITED STATES
14 REM NUCLEAR REGULATORY COMMISSION, NOR ANY OF THEIR EMPLOYEES,
15 REM NOR ANY OF THEIR CONTRACTORS, SUBCONTRACTORS, OR THEIR
16 REM EMPLOYEES, MAKES ANY WARRANTY, EXPRESSED OR IMPLIED, OR
17 REM ASSUMES ANY LEGAL LIABILITY OR RESPONSIBILITY FOR THE
18 REM ACCURACY, COMPLETENESS OR USEFULNESS OF ANY INFORMATION,
19 REM APPARATUS, PRODUCT OR PROCESS DISCLOSED, OR REPRESENTS THAT
20 REM ITS USE WOULD NOT INFRINGE PRIVATELY OWNED RIGHTS.
21 REM
22 REM *****
23 REM
24 REM AUTHOR: T.L. ZIMMERMAN, FORMALLY OF ENERGY INCORPORATED
25 REM                               ALBUQUERQUE, NM
26 REM
27 REM FOR INFORMATION CONTACT:
28 REM
29 REM A. C. PAYNE, JR.           PHONE: (505) 846-3568
30 REM D. W. WHITEHEAD           PHONE: (505) 844-2832
31 REM
32 REM REACTOR SYSTEMS SAFETY ANALYSIS DIVISION 6412
33 REM SANDIA NATIONAL LABORATORIES, ALBUQUERQUE, NM 87185
34 REM
35 REM *****
40 SCREEN 0,0,0 'To run on either color or mono screens.
45 PRINT "      --- INDEX2 ---"; PRINT
50 REM This routine prints an index and checks linkage errors for all SETS
60 REM files (modules) residing on the disks in drives A and B...
70 DEFINT F,I,J,K,L,M,P : OPTION BASE 1
80 WIDTH "LPT1:",160 : CHDIR "A:\": CHDIR "B:\": IO=1 : DIS="A:"
90 DIM TS(600,2),ES(50),TOPS(600,2),DES(1600,2)
100 ON ERROR GOTO 1800
110 SOUND 4401,41 : SOUND 371,21 : SOUND 4401,41 : SOUND 8801,371
120 PRINT : COLOR 15,0: PRINT "OUTPUT OPTIONS:" : COLOR 7,0
130 PRINT "  1. Complete output/Routed to printer.
140 PRINT "  2. No printed index list/Route all checks to printer."
150 PRINT "  3. Complete output/No routing to printer.
160 PRINT : INPUT "Enter Selection: ";N7
170 IF N7=1 THEN QS="Y": Q18="Y": GOTO 210
180 IF N7=2 THEN QS="Y": Q18="N": GOTO 210
190 IF N7=3 THEN QS="N": Q18="N": GOTO 210
200 N7=1 : GOTO 170
210 OPEN DIS+"FILELIST" FOR INPUT AS #3
220 IF IO>1 THEN FOR I=1 TO F-1: LINE INPUT #3,TS(I,1): NEXT
230 FOR I=IO TO 1000
240 IF EOF(3) GOTO 270
250 LINE INPUT #3,TS(I,1)
260 NEXT
270 CLOSE #3
280 N=I-1 : F=0 : F1=6

```

Table B.2 (Continued)
Program Listing for INDEX2

```

290 IF INSTR(T$(2,1),"Volume")=0 THEN IF I0=1 THEN GOTO 320 ELSE GOTO 370
300 M=I-2: P=7: P1=5: IF I0=1 GOTO 370 'Unsorted FILELIST...
310 IF Q1$="Y" THEN LPRINT CHR$(12);CHR$(14);"Unsorted Index:" :GOTO 330
320 IF Q1$="Y" THEN LPRINT CHR$(12);CHR$(14);"Sorted Index:"
330 IF LEFT$(T$(M,1),2)<>".." THEN DIS$="B:" :I0=N+1: M1=M: GOTO 210
340 SOUND 246.9,20 : SOUND 987.8,20 : FOR J=1 TO 3000 : NEXT
350 PRINT :PRINT "**** THERE ARE NO SETS FILES IN DIRECTORY 'SETS' ****"
360 GOTO 2090
370 M2=M
380 FOR I=P TO M
390 FILES=LEFT$(T$(I,1),5)
400 FIMS=MIDS(T$(I,1),24,16)
410 T$(I,1)=FILES : T$(I,2)=FIMS
420 NEXT
430 IF Q1$="Y" THEN LPRINT SPC(25);"***** DISK A FILES *****":GOSUB 1100
440 DIM DEL(100) : DEL(1)=99 : I2=0 : K=0 : K=1
450 DIS$="A:" : CHDIR "A:\SETS" : P0=P : M=M1 : I=PRE(M)
460 FOR I=P0 TO M
470 S3$=DIS$+T$(I,1)
480 IF K>52 THEN K=0:GOSUB 1090
490 OPEN S3$ FOR INPUT AS #1
500 LINE INPUT #1,E$(1)
510 P1=INSTR(E$(1),"$")+1 : P2=INSTR(P1,E$(1),".")
520 IF P1=1 OR P2=0 THEN FIMS="?????" : GOTO 540
530 FIMS=MIDS(E$(1),P1,P2-P1)
540 IF EOP(1) GOTO 730
550 LINE INPUT #1,E$(2)
560 P3=INSTR(E$(2),"GS")+2 : IF P3=2 GOTO 540 'Find the top gate...
570 B$=MIDS(E$(2),P3,1) :IF B$=" " THEN P3=P3+1 :GOTO 570
580 P4=INSTR(P3,E$(2),".")-1
590 B$=MIDS(E$(2),P4,1) :IF B$=" " THEN P4=P4-1:GOTO 590
600 TOP$(I,1)=MIDS(E$(2),P3,P4-P3+1) :TOP$(I,2)=FIMS 'Save top gate & tree name
610 IF EOP(1) GOTO 760
620 LINE INPUT #1,E$(1)
630 P3=INSTR(E$(1),"DES")+3 : IF P3=3 GOTO 650 'Find any developed events...
640 P3=INSTR(E$(1),"TES")+3 : IF P3=3 GOTO 700
650 B$=MIDS(E$(1),P3,1) :IF B$=" " THEN P3=P3+1 :GOTO 650
660 P4=INSTR(P3,E$(1),".")-1
670 B$=MIDS(E$(1),P4,1) :IF B$=" " THEN P4=P4-1:GOTO 670
680 L$=L$+1
690 DES(L$,1)=MIDS(E$(1),P3,P4-P3+1) :DES(L$,2)=FIMS 'Save developed event name
700 IF INSTR(E$(1),"TREES")>0 GOTO 800
710 IF INSTR(E$(1),"FAULT TREE")>0 GOTO 800
720 GOTO 610
730 BEEP: PRINT "*** Incomplete Module: ";T$(I,1)
740 IF Q1$="Y" THEN LPRINT "*** Incomplete Module: ";T$(I,1): LPRINT: K=K+1
750 CLOSE #1 : GOTO 1010
760 BEEP: PRINT "*** Second 'FAULTTREES' delimiter not found for module ";T$(I,1)
770 IF Q1$="N" GOTO 790
780 LPRINT: LPRINT "***Second 'FAULTTREES' delimiter not found for module ";T$(I,1) :K=K+2
790 E$(1)=" " : IJ=1: GOTO 860
800 IJ=1
810 LINE INPUT #1,E$(IJ)
820 Z$=RIGHT$(E$(IJ),1) :IF Z$="S" GOTO 860
830 IF Z$<>" " THEN GOTO 850 ELSE L$=LEN(E$(IJ))-1
840 E$(IJ)=LEFT$(E$(IJ),L$) : GOTO 820
850 IJ=IJ+1 : GOTO 810
860 CLOSE #1
870 PRINT :PRINT "Filename: ";T$(I,1): PRINT "Fault Tree Name: ";FIMS
880 PRINT "Created on: ";T$(I,2)
890 PRINT "Fault Tree (Module) Top Event:"

```

Table B.2 (Continued)
Program Listing for INDEX2

```

900 FOR IK=1 TO IJ
910 PRINT SPC(3);ES(IK)
920 NEXT IK
930 K=K+1
940 IF Q18="N" GOTO 1010
950 LPRINT: IX=1
960 LPRINT USING "\          \";TS(I,1);PTNS;
970 LPRINT TS(I,2);SPC(3);ES(1):K=K+2
980 IF IX=IJ GOTO 1010
990 IX=IX+1:K=K+1
1000 LPRINT SPC(48);ES(IK):GOTO 980
1010 NEXT I
1020 IF N=N2 GOTO 1060
1030 PRINT:COLOR 15,0: BEEP: PRINT "  DISK B FILES:": COLOR 7,0
1040 IF Q18="Y" THEN LPRINT CHR$(12);SPC(25);"***** DISK B FILES *****":GOSUB 1100
1050 K=1:N=N2:P0=N1+1:DIS="B":CHDIR"B:\SETS":I=FRE(KS):GOTO 460
1060 PRINT:PRINT "  ---";K9;" Files/Modules in Index ---"
1070 IF Q18="Y" THEN LPRINT:LPRINT "  ---";K9;" Files/Modules in Index ---"
1080 GOTO 1150
1090 LPRINT CHR$(12)
1100 LPRINT CHR$(27);"4";
1110 LPRINT:LPRINT"Filename      Fault Tree Name      Created on          Fault Tree (Module) Top Event"
1120 LPRINT CHR$(27);"5"
1130 RETURN
1140 REM ***** MODULE ERROR CHECKING ROUTINES *****
1150 BEEP: PRINT:COLOR 15,0:PRINT "Duplicate Top Gates:":PRINT:COLOR 7,0
1160 IF Q8="N" GOTO 1180
1170 BEEP: LPRINT CHR$(12);CHR$(14);"Duplicate Top Gates:":LPRINT
1180 K=0: BEEP
1190 FOR J1=F TO N
1200 GATES=TOPS(J1,1):FL=0
1210 IF INSTR(GATES,"")>0 GOTO 1260
1220 TOPS(J1,1)=TOPS(J1,1)+" "
1230 FOR J2=F TO N
1240 IF TOPS(J2,1)=GATES THEN GOSUB 1310
1250 NEXT J2
1260 NEXT J1
1270 IF K=0 GOTO 1400
1280 PRINT "      NONE"
1290 IF Q8="Y" THEN LPRINT "      NONE"
1300 GOTO 1400
1310 FL=FL+1:K=1:IF FL>1 GOTO 1370
1320 PRINT "GATE: ";GATES
1330 PRINT "      Module:":PRINT "      ";TOPS(J1,2)
1340 IF Q8="N" GOTO 1370
1350 LPRINT: LPRINT "GATE: ";GATES
1360 LPRINT "      Module:":LPRINT "      ";TOPS(J1,2)
1370 PRINT "      ";TOPS(J2,2)
1380 IF Q8="Y" THEN LPRINT "      ";TOPS(J2,2)
1390 RETURN
1400 FOR J1=F TO N 'Strip top mark from gates array...
1410 P1=LEN(TOPS(J1,1))-1
1420 TOPS(J1,1)=LEFT$(TOPS(J1,1),P1)
1430 NEXT
1440 PRINT:COLOR 15,0:PRINT "  --- Module Linkage Check ---":PRINT:COLOR 7,0
1450 IF Q8="N" GOTO 1470
1460 LPRINT CHR$(12);CHR$(14);"  --- Module Linkage Check ---":LPRINT
1470 FOR I=1 TO L9
1480 DEV$=DES(I,1):FL=0:L8=0
1490 IF Q28="N" GOTO 1500
1500 PRINT:PRINT "Dev Event: ";DEV$;"  Module: ";DES(I,2)

```

Table B.2 (Concluded)
Program Listing for INDEX2

```

1510 IF Q8="Y" THEN LPRINT :LPRINT "DE: ";DEV$;"  Module: ";DES(I,2)
1520 FOR J=F TO N
1530 IF TOPS(J,1)=DEV$ THEN GOSUB 1630
1540 NEXT J
1550 IF FL>0 GOTO 1590
1560 PRINT "****NO MODULE FOUND CORRESPONDING TO DE: ";DEV$;"  -Module: ";DES(I,2)
1570 IF Q8="N" GOTO 1590
1580 LPRINT "****NO MODULE FOUND CORRESPONDING TO DE: ";DEV$;"  -Module: ";DES(I,2)
1590 NEXT I
1600 PRINT "  *** COMPLETED ***"
1610 IF Q8="Y" THEN LPRINT:LPRINT "      *** COMPLETED ***"
1620 GOTO 1690
1630 FL=FL+1 : IF Q2$="N" GOTO 1670
1640 PRINT "      Linked to Module: ";TOPS(J,2)
1650 IF Q8="Y" THEN LPRINT "      Link to Module: ";TOPS(J,2)
1660 TOPS(J,2)=TOPS(J,2)+*****
1670 RETURN
1680 COLOR 15,0: PRINT :PRINT "Unlinked Top Gates:" :COLOR 7,0
1690 IF Q8="Y" THEN LPRINT CHR$(12);CHR$(14);"Unlinked Top Gates:"
1700 K=0
1710 FOR J=F TO N
1720 IF INSTR(TOPS(J,2),****)=0 THEN GOSUB 1760
1730 NEXT J
1740 IF K=0 GOTO 1820
1750 PRINT :PRINT "      NONE"
1760 IF Q8="Y" THEN LPRINT :LPRINT "      NONE"
1770 GOTO 1820
1780 K=K+1
1790 PRINT "Module: ";TOPS(J,2);"  Top Gate: ";TOPS(J,1)
1800 IF Q8="Y" THEN LPRINT:LPRINT "Module: ";TOPS(J,2);"  Top Gate: ";TOPS(J,1)
1810 RETURN
1820 PRINT :COLOR 15,0: PRINT "  --- ALL CHECKS COMPLETED ---": COLOR 7,0
1830 IF Q8="N" GOTO 1850
1840 LPRINT :LPRINT:LPRINT CHR$(14); "      --- ALL CHECKS COMPLETED ---"
1850 SOUND 261.6,91 :SOUND 323.2,91 :SOUND 1046.5,91
1860 FOR J=1 TO 3000 : NEXT
1870 GOTO 2080
1880 REM *****
1890 REM Error trapping routine
1900 IF ERR<>53 GOTO 1990
1910 IF ERR<>170 GOTO 1950
1920 SOUND 174.8,40 : FOR JL=1 TO 3000 : NEXT
1930 COLOR 15: PRINT :PRINT "      *** 'FILELIST' NOT FOUND ON DISK ***"
1940 GOTO 2080
1950 SOUND 228,18: PRINT "****FILE ";S3$;" NOT FOUND (on this disk)****"
1960 IF Q8="N" GOTO 1980 ELSE K=K+2
1970 LPRINT: LPRINT CHR$(14);"FILE ";S3$;" NOT FOUND ON DISK ****": LPRINT
1980 RESUME 1910
1990 IF ERR<>78 GOTO 2020
2000 PRINT "This disk DOES NOT contain the SETS directory."
2010 CHDIR "\": RESUME NEXT
2020 SOUND 198,12 : PRINT "      ***** FATAL ERROR *****"
2030 PRINT "  Enter 'SYSTEM' if you wish to exit BASIC.": SOUND 146.8,20
2040 SOUND 130.8,20: FOR IL=1 TO 6000 : NEXT
2050 ON ERROR GOTO 0
2060 CHDIR "\": END
2070 REM *****
2080 CHDIR "\": CHDIR "B:\": SYSTEM
2090 END

```

Table B.3
INDEX Batch File

```
ECHO OFF
ECHO          -- SETS FILE INDEXING PROGRAM --
ECHO
ECHO *** Generates an index of the files in the SETS directory on a disk
ECHO      and performs module linkage error checking on these files ***
IF %1--NOSORT GOTO F1
IF %1--RESUME GOTO F3
ECHO          (Files are indexed in ASCII sorted order)
:F1
ECHO
IF EXIST FILELIST GOTO HALT1
ECHO A disk directory listing will now be printed...
DIR \SETS >PRN
IF %1--NOSORT GOTO F2
DIR \SETS |SORT >FILELIST
ECHO OFF
GOTO F3
:F2
DIR \SETS >FILELIST
:F3
BASICA INDEX
ECHO  ----- DISK FILE INDEX COMPLETED -----
GOTO END
:HALT1
ECHO  THE FILE 'FILELIST' ALREADY EXISTS ON THIS DISK...
ECHO  THIS FILE MUST BE DELETED OR RENAMED PRIOR TO RUNNING INDEX.
:END
```

Table B.4
INDEX2 Batch File

```

ECHO OFF
ECHO          -- SETS FILE INDEXING PROGRAM II--
ECHO
ECHO *** Generates an index of the files in the SETS directories on
ECHO      TWO disks and performs module linkage error checks across
ECHO      all these files ***
IF %1==NOSORT GOTO F1
IF %1==RESUME GOTO F3
ECHO      (Files within each disk are indexed in ASCII sorted order)
:F1
ECHO
IF EXIST FILELIST GOTO HALT1
ECHO The SETS directory for Disk A will now be printed...
DIR A:\SETS >PRN
ECHO The SETS directory for Disk B will now be printed...
DIR B:\SETS >PRN
IF %1==NOSORT GOTO F2
DIR A:\SETS |SORT >A:FILELIST
ECHO OFF
DIR B:\SETS |B:SORT >B:FILELIST
ECHO OFF
GOTO F3
:F2
DIR A:\SETS >A:FILELIST
DIR B:\SETS >B:FILELIST
:F3
BASICA INDEX2
ECHO  ----- DISK FILE INDEX COMPLETED -----
GOTO END
:HALT1
ECHO  THE FILE 'FILELIST' ALREADY EXISTS ON DISK A...
ECHO  THIS FILE MUST BE DELETED OR RENAMED PRIOR TO RUNNING INDEX.
:END

```

B.3 Output

If requested, the printed index testing shows for each module on the disk(s):/ the disk file name, the date file was created, the fault tree name (from the first line in the module), and the descriptive text for the module top event. When all files/modules have been read, the total number of files is printed.

The following checks are made by the error checking routine:

Duplicate Top Gates	A check for modules that have top events with the same name. If any are discovered, the gate name, file name and top event descriptions are printed for both modules involved.
Developed Event Links	All modules containing developed events are listed and a search is made for the module (gate) to which it should link. If the linked module is found, a "linked" message is printed and the module name is listed. If no module with the same top event name as the developed event is located, the message "****NO MODULE FOUND CORRESPONDING TO DE:" followed by the unlinked developed event is printed.
Unlinked Top Gates	All modules with top events that do not link to any other module on the disk(s) are listed.

In most cases, a correctly constructed fault tree and its modules should have one unlinked top gate (the top gate of the tree), no duplicate top gates, and perhaps several unmatched developed events if parts of the tree (e.g., support systems) have been developed and stored on other disks.

APPENDIX C

REVISED MODULAR FAULT TREE LISTINGS

C.1 Introduction

This appendix contains a complete set of the modified and corrected modular fault trees originally contained in Appendix I of Reference [1]. These changes* were made to reduce difficulties encountered in their use during the construction of LaSalle Unit 2 system fault trees as part of the Risk Methods Integration and Evaluation Program (RMIEP) program. In addition it contains three logic modules that were added to the modules of Appendix I. Each module is identified by its title and the corresponding Appendix I page number.

C.2 Updated Modular Fault Trees

The fault tree modules that follow are reproduced in the same order as they appear in Appendix I. The corrected files can be obtained from the Energy Science and Technology Software Center (ESTSC).

*NOTE: The symbol b is used to represent a required blank space.

REFERENCES

1. Varnado, G. B., W. H. Horton, and P. R. Lobner, Modular Fault Tree Analysis Procedures Guide, NUREG/CR-3268, SAND83-0963, (4 volumes), Sandia National Laboratories, Albuquerque, NM, August 1983.

Fault Tree Module for Fluid System Top Event

SYS OF INTRES =

SOI =

PERFORM FUNCTION =

M OF N =

:

X OF Y =

COMPTYP1 =

:

COMPTYPN =

MOFNT =

:

XOFYZ =

DSCRIP1 =

:

DSCRIPN =

SS1 =

:

SSN =

BOOLEAN =

Fault Tree Module for Fluid System Rule 1
(2-of-2 In, 1 Out) (1 APP/NODE)

SYS OF INTRES -

SOI -

DSCRIP0 -

DSCRIP1 -

DSCRIP2 -

OPS -

IP1 -

IP2 -

16

Fault Tree Module for Fluid System Rule 2
(1-of-2 In, 1 Out) (1 APP/NODE)

SYS OF INTRES =

SOI =

DSCRIP0 =

OPS =

DSCRIP1 =

IP1 =

DSCRIP2 =

IP2 =

17

Fault Tree Module for Fluid System Rule 3
(1 In, N Out) (N APP/NODE)

SYS OF INTRES =
SOI =
DSCRIPD =
OPI =
DSCRIPS =
IPS =

I8

Fault Tree Module for Fluid System Rule 4
(1 In, 2-of-2 Out) (1 APP/NODE, 2 R3 APP/NODE)

SYS OF INTRES =

SO1 =

FAULT X = FAULT

FLT-X = FLT-

DSCRIP1 =

OP1 =

DSCRIP2 =

OP2 =

Fault Tree Module for Fluid System Rule 5
(M-of-N In, 1 Out) (1 APP/NODE)

SYS OF INTRES =

SOI =

DSCRIP0 =

OPS =

BOOLEAN =

DSCRIP1 =

IP1 =

IP2 =

:

IPN =

110

Fault Tree Module for Fluid System Rule 6
(1 In, M-of-N Out) (1 APP/NODE, N R3 APP/NODE)

SYS QF INTRES =

SOI =

FAULT X = FAULT

FLT-X = FLT-

BOOLEAN =

IPS =

DSCRIP1 =

DSCRIP2 =

:

DSCRIPN =

OP1 =

OP2 =

:

OPN =

111

Fault Tree Module for Fluid System Rule 7
(1 In, 1 DVT) (1 APP/NODE)

SYS OF INTRES -

SOI -

DSCRIP0 -

DSCRIP1 -

DSCRIP2 -

IPS -

OP1 -

OP2 -

Fault Tree Module for Fluid System Rule 8
(2-of-2 In, 2 Out) (2 APP/NODE)

SYS OF INTRES -

SOI -

DESCRIP0 -

DESCRIP1 -

DESCRIP2 -

DESCRIP3 -

DESCRIPC -

OP1 -

OP2 -

CTS -

IP1 -

IP2 -

113

Fault Tree Module for Fluid System Rule 9
(2 In, 2-of-2 Out) (1 APP/NODE; 2 R8 APP/NODE)

Fault Tree Module for Fluid System Rule 10 (Sheet 1 of 2)
(3-of-3 In, 2 Out) (2 APP/NODE)

SYS OF INTRES =

SOI =

OP1 =

CTX =

IP1 =

CT1 =

I15

Fault Tree Module for Fluid System Rule 10 (Sheet 2 of 2)
(3-of-3 In, 2 Out) (2 APP/NODE)

116

Fault Tree Module for Fluid System Rule 11
(3 In, 2-of-2 Out) (1 APP/NODE, 2 RIO APP/NODE)

SYS OF INTRES =

SOI =

FAULT X = FAULT

FLT-X = FLT-

OP1 =

OP2 =

CTX =

CT1 =

CT2 =

IP1 =

IP2 =

IP3 =

Fault Tree Module for Fluid System Rule 12 (Sheet 1 of 2)
(2-of-2 In, 3 Out) (3 APP/NODE)

Fault Tree Module for Fluid System Rule 12 (Sheet 2 of 2)
(2-of-2 In, 3 Out) (3 APP/NODE)

Fault Tree Module for Fluid System Rule 13 (Sheet 1 of 4)
 (2 In, 2-of-3 Out) (1 APP/NODE, 3 R12 APP/NODE)

SYS OF INTRES =
 SOI =
 FAULT W = FAULT
 FLT-W = FLT-

OP1 =
 OP2 =
 OP3 =
 CT1 =
 CT2 =

Fault Tree Module for Fluid System Rule 13 (Sheet 2 of 4)
 (2 In, 2-of-3 Out) (1 APP/NODE, 3 R12 APP/NODE)

SYS OF INTRES =

SOI =

FAULT X = FAULT

FLT-X = FLT-

OP1 =

OP3 =

CTX =

CT1 =

CT2 =

IP1 =

IP2 =

Fault Tree Module for Fluid System Rule 13 (Sheet 3 of 4)
(2 IN, 2-of-3 Out) (1 APP/NODE, 3 RI2 APP/NODE)

SYS OF INTRES =

SOI =

FAULT Y =

FLT-Y = FLT-

OP1 =

OP2 =

CTYA =

CTYB =

CT1 =

CT2 =

CTY =

IP1 =

IP2 =

OP3 =

Fault Tree Module for Fluid System Rule 13 (Sheet 4 of 4)
 (2 In, 2-of-3 Out) (1 APP/NODE, 3 R12 APP/NODE)

SYS OF INTRES *

SOI *

FAULT Z = FAULT

FLT-Z = FLT-

OP2 *

OP3 *

CTZA *

CTZB *

CTZ *

CT2 *

IP1 *

IP2 *

OP1 *

CT1 *

123

Fault Tree Module for Fluid System Rule 14
(N In, 1 DVT) (N APP/NODE)

SYS OF INTRES =

SOI =

DSCRIP1 =

DSCRIP0 =

IP1 =

OPS =

Fault Tree Module for Fluid System Rule 15
(1 In, M-of-N DVT) (1 APP/NODE)

SYS OF INTRES =

SOI =

DSCRIP1 =

IPS =

DSCRIP1 =

:

DSCRIPN =

OP1 =

:

OPN =

BOOLEAN =

I25

Fault Tree Module for Fluid System Rule 16
(1 Source) (1 APP/NODE)

SYS OF INTRES =

SOI =

DSCRIPE =

EPS =

Fault Tree Module for Fluid System Rule 17
(1 Sink) (1 APP/NODE)

SYS OF INTRES =

SOI =

DESCRIBE =

EPS =

FLOW DIVERTED DUE TO
FAULT AFTER INPUT OF
SYS OF INTRES LINE
DESCRIBE PIPE SEG EPS

FOR-IN-SOI-EPS

DIVERTED FLOW DUE TO
COMPONENT FAULT IN
SYS OF INTRES LINE
DESCRIBE PIPE SEG EPS

DIVRT-SOI-EPS

Alignment Fault Interface Module 1
(Top Event Interface)

SYS OF INTRES =

SOI =

M OF N =

MOFNT =

X OF Y =

XOFYZ =

COMPTYP1 =

COMPTYP2 =

FAULT X = FAULT

FLT-X = FLT-

FAULT Y = FAULT

FLT-Y = FLT-

BOOLEAN =

SS1 =

:

SSN =

Alignment Fault Interface Module 2
(Rules 1, 5, 8, 9, 10, 11, 12, and 13 Interface)

SYS OF INTRES =

SOI =

OPS =

FAULT X = FAULT

:

FAULT Y = FAULT

FLT-X = FLT-

:

FLT-Y = FLT-

Alignment Fault Interface Module 3
(Rule 2 Interface)

SYS OF INTRES =

SOI =

OPS =

IP1 =

IP2 =

FAULT X = FAULT

:

FAULT Y = FAULT

FLT-X = FLT-

:

FLT-Y = FLT-

Fault Tree Module for Loss of Function in Active Segments

SYS OF INTRES =
 SOI =
 DSCRIPT =
 SGT =
 COMPIDPP =
 COMPIDHX =
 COMPIDTK =
 COMPIDMC =
 WS =

COMPIDMP =
 XS =
 COMPIDTP =
 VS =
 COMPIDEP =
 ZS =
 COMPIDRV =
 UC =

COMPIDCV =
 WC =
 COMPIDMV =
 XC =
 COMPIDNV =
 YC =
 COMPIDSV =
 ZC =
 COMPIDXV =
 VC =

Fault Tree Module for Loss of Function in Passive Segments

SYS OF INTRES =

SOI =

DSCRIPS =

SGT =

DSCRIPT =

NO FLOW DUE TO
COMPONENT FAULT IN
SYS OF INTRES LINE
DSCRIPS PIPE SEG SGT

NO-FL-501-SGT

SYS OF INTRES LINE
DSCRIPS PIPE SEG SGT
LOSS OF FUNCTION HAS
CONTRIBUTING FAULTS

EE-OL-501-SGT

Fault Tree Module for Diversion of Flow in Active Segments

SYS OF INTRES =
 SOI =
 DSCRIPT =
 SGT =
 COMPIDRV =
 UO =
 COMPIDCV =
 WO =

COMPIDMV =
 XO =
 COMPIDNV =
 YO =
 COMPIDSV =
 ZO =
 COMPIDXV =
 VO =

COMPIDMC =
 WR =
 COMPIDMP =
 XR =
 COMPIDTP =
 VR =
 COMPIDEP =
 ZR =
 COMPIDPP =

Fault Tree Module for Diversion of Flow in Passive Segments

DIVERTED FLOW DUE TO
COMPONENT FAULT IN
SYS OF INTRES LINE
DSCRIPT PIPE SEG SGT

01PRT-501-SGT

SYS OF INTRES LINE
DSCRIPT PIPE SEG SGT
ITSELF CONTRIBUTES
TO DIVERSION FAULTS

11-010-501-SGT

SYS OF INTRES =

SOI =

DSCRIPT =

SGT =

Fault Tree Module for a Motor-Driven Pump

COMPIDEN =

PMX =

XX =

ROOMRRR =

SYSTEM =

TYP =

MECFATSTATDES =

FLST =

Fault Tree for a Turbine-Driven Pump

COMPIDEN =
PTX =
XX =
ROOMRRR =

SYST6 =
TYP =
MECFALSTATDES =
FLST =

Fault Tree Module for an Engine-Driven Pump (Sheet 1 of 2)

COMPIDEN =
 PEX =
 XX =
 ROOMRRR =

SYSTM =
 TYP =
 MECFALTSTATDES =
 FLST =

Fault Tree Module for an Engine-Driven Pump (Sheet 2 of 2)

COMPIDEN =
 PEX =
 COMPCPCL =
 VDC =
 COMPCPBS =
 COMPCPFS =
 COMPCPCB =
 XO =

Fault Tree for a Motor-Driven Compressor

COMPIDEN =
 EMX =
 XX =
 ROOMRRR =

SYST6 =
 TYP =
 MECFALTSTATDES =
 FLST =

Fault Tree Module for a Motor-Operated Valve

Fault Tree Module for a Pneumatic-Hydraulic Valve

COMPIDEN =
 XX =
 COMPIDCL =
 VLT =
 COMPIDBS =
 COMPIDFS =
 COMPIDCB =
 XO =

SYSTØ =
 TYP =
 MECFALSTATDES =
 FLST =

Fault Tree Module for a Solenoid-Operated Valve

COMPIDEN =
 XX =
 COMPIDCL =
 VLT =
 COMPIDBS =

COMPIDFS =
 COMPIDCB =
 XO =
 SYSTB =
 TYP =
 MECFALTSTATDES =
 FLST =

Fault Tree Module for a Safety/Relief Valve (Sheet 1 of 2)

COMPIDEN =
XX =
BOOLEAN =

Fault Tree Module for a Safety/Relief Valve (Sheet 2 of 2)

COMPIDEN =
 XX =
 COMPIDCL =
 VLT =
 COMPIDBS =

COMPIDFS =
 COMPIDCB =
 XO =
 SYSTEM =
 TYP =

MECFALTSTATDES =
 FLST =

I44
Fault Tree Module for a Manual Valve

Fault Tree Module for a Check Valve

Fault Tree Module for a Heat Exchanger

Fault Tree Module for a Tank

Fault Tree Module for a Pipe Segment

COMPIDEN =

-M =

Fault Tree Module for Component Interface with Fluid Support Systems

PERFORM COMP SUPPORT
OR SERVICE FUNCTS TO -

FNC -

COMPIDEN -

Z -

SYS OF INTRES -

SOI -

IFS -

M OF N -

COMPTYP1 -

NOFNT -

X OF Y -

COMPTYP2 -

XO FY2 -

COMPIDNV -

VC -

COMPIDNV -

XC -

COMPIDNV -

YC -

COMPIDSV -

ZC -

COMP -

Fault Tree Module for the Primary Reactivity Control System (Sheet 1 of 2)

III =

FFF =

WWW =

COMPIDEA =

:

COMPIDEY =

FTI BOOLEAN =

TYP =

Fault Tree for the Primary Reactivity Control System (Sheet 2 of 2)

III =
 WWW =
 UI BOOLEAN =
 UW BOOLEAN =
 COMPIDEY =
 :
 COMPIDEA =
 TYP =

Fault Tree Module for a Motor-Driven Reactivity Control Rod - Fail to Insert (Sheet 1 of 3)

Fault Tree Module for a Motor-Driven Reactivity Control Rod - Fail to Insert (Sheet 2 of 3)

COMPIDEN =

EVENT TYPE (EVENT)	EVENT IDENTIFICATION (ID)				
TRAN					
LOCA

Fault Tree Module for a Motor-Driven Reactivity Control Rod - Fail to Insert (Sheet 3 of 3)

GRP =
 VLT =
 COMPIDB1 =
 COMPIDB2 =
 COMPIDEA =
 :
 COMPIDED =
 COMPIDEE =
 :
 COMPIDEN =

Fault Tree Module for a Motor-Driven Reactivity Control Rod - Unintentional Withdrawal

COMPIDEN *

Fault Tree Module for a Motor-Driven Reactivity Control Rod - Unintentional Insertion (Sheet 1 of 3)

COMPIDEN =

GRP =

ROOMRRR =

Fault Tree Module for a Motor-Driven Reactivity Control Rod - Unintentional Insertion (Sheet 2 of 3)

GRP =
 COMPIDC1 =
 VLT =
 COMPIDB1 =
 COMPIDX1 =
 COMPING1 =
 COMPIDEA =
 :
 COMPIDED =

Fault Tree Module for a Motor-Driven Reactivity Control Rod - Unintentional Insertion (Sheet 3 of 3)

GRP =
 COMPIDC2 =
 VLT =
 COMPIDB2 =
 COMPIDX2 =
 COMPIMG2 =
 COMPIDEE =
 :
 COMPIDEN =

Fault Tree Module for a Hydraulically-Driven Reactivity Control Rod - Fail to Insert (Sheet 1 of 2)

CONFIDEN =

Fault Tree Module for a Hydraulically-Driven Reactivity Control Rod - Fail to Insert (Sheet 2 of 2)

Fault Tree Module for a Hydraulically-Driven Reactivity Control Rod - Unintentional Withdrawal

COMPIDEN =

Fault Tree Module for a Hydraulically-Driven Reactivity Control Rod - Unintentional Insertion

Fault Tree Module for a Ventilation System Top Event

SYSTEM-OF-INTEREST =

SOI =

M-OF-N =

COMPTYP1 =

X-OF-Y =

COMPTYP2 =

MOFNT =

XOFYZ =

BOOLEAN =

SS1 =

i

SSN =

Fault Tree Module for Fluid System Rule 3

SYSTEM-OF-INTEREST =

SOI =

OP1 =

IPS =

Fault Tree Module for Fluid System Rule 5

169

Fault Tree Module for Fluid System Rule 14

Fault Tree Module for Fluid System Rule 15

Fault Tree Module for Fluid System Rule 16

FAULT OCCURS PRIOR
TO THE OUTPUT OF
SYSTEM-OF-INTEREST
SEGMENT EPS
FOP-OUT-SOI-EPS

SYSTEM-OF-INTEREST =
SOI =
EPS =

COMPONENT FAULT
OCCURS IN
SYSTEM-OF-INTEREST
SEGMENT EPS
FAULT-SOI-EPS

Fault Tree Module for Fluid System Rule 17

FAULT OCCURS AFTER
THE INPUT OF
SYSTEM-OF-INTEREST
SEGMENT EPS

FOR-IN-SOI-EPS

DIVERGED FLOW OCCURS
IN
SYSTEM-OF-INTEREST
SEGMENT EPS

DIVRT-SOI-EPS

SYSTEM-OF-INTEREST =

SOI =

EPS =

Fault Tree Module for Loss of Function in Active Ventilation System Segments

SYSTEM-OF-INTEREST =

SOI =

SGT =

COMPIDMF =

XS =

COMPIDAC =

COMPIDAH =

COMPIDAX =

COMPIDXD =

XC =

COMPIDMD =

YC =

COMPIDND =

BC =

COMPIDDD =

COMPIDHR =

Fault Tree Module for Loss of Function in Passive Ventilation System Segments

SYSTEM-OF-INTEREST =

SOI =

SGT =

Fault Tree Module for Diversion of Flow in Active Ventilation System Segments

SYSTEM-OF-INTEREST =
 SOI =
 SGT =
 COMPIDDD =
 COMPIDMF =
 KR =
 COMPIDXD =
 XO =
 COMPIDMD =
 YO =
 COMPIDND =
 ZO =

Fault Tree Module for Diversion of Flow in Passive Ventilation System Segments

DIVERTED FLOW OCCURS
IN
SYSTEM-OF-INTEREST
SEGMENT SGT

DIVRT-SOI-SGT

SYSTEM-OF-INTEREST =

SOI =

SGT =

SYSTEM-OF-INTEREST
SEGMENT SGT IS IN
ITSELF CONTRIBUTING
TO DIVERSION FAULTS

EC-CFD-SOI-SGT

Fault Tree Module for a Motor-Driven Fan

Fault Tree Module for a Motor-Operated Damper

Fault Tree Module for a Pneumatic/Hydraulic Damper

COMPIDEN =
 XX =
 SYSTP =
 TYP =

MECFATSTATDES =
 FLST =
 COMPIDCL =
 VLT =
 COMPIDBS =
 COMPIDFS =
 COMPIDCB =
 XO =

Fault Tree Module for a Manual Damper

Fault Tree Module for an Air Cooling Heat Exchanger

Fault Tree Module for an Air Heating Unit

Fault Tree Module for an Air Cleaning Unit

Fault Tree Module for a Hydrogen Recombiner Unit

Fault Tree Module for a Duct Segment

Fault Tree Module for Component Interface with Room Cooling Systems

Fault Tree Module for Room Ventilation System Interface

ROOMRRR =
 TYPE =
 -T = -
 SYSTEM-OF-INTEREST =
 SOI =
 IFS =
 M-OF-N =
 COMPTYP1 =
 X-OF-Y =
 COMPTYP2 =
 MOFNT =
 XOFYZ =
 COMPIDXD =
 XC =
 COMPIDMD =
 YC =
 COMPIDND =
 ZC =

Fault Tree Module for Electric Power Faults (Sheet 1 of 5)

COMPIDEN =
 COMP =
 TYP =
 -T- =

Fault Tree Module for Electric Power Faults (Sheet 2 of 5)

Fault Tree Module for Electric Power Faults (Sheet 3 of 5)

Fault Tree Module for Electric Power Faults (Sheet 4 of 5)

Fault Tree Module for Electric Power Faults (Sheet 5 of 5)

Fault Tree Module for Loss of Power at Bus

Fault Tree Module for Loss of AC Power Sources for a Bus (Sheet 1 of 2)

Fault Tree Module for Loss of AC Power Sources for a Bus (Sheet 2 of 2)

COMPIDEN =
 VLT =
 COMPIDAC =
 COMPIDXF =
 VAC =
 COMPIDCL =

COMPIDCB =
 XO =
 COMPIDRC =
 COMPIDSW =
 COMPIDMG =

Fault Tree Module for Loss of DC Power Sources for a Bus

COMPIDEN =
 VLT =
 COMPIDDC =
 COMPIDBT =
 COMPIDCL =
 VDC =
 COMPIDMG =
 COMPIDIN =
 COMPIDCB =
 XO =
 COMPIDSW =

Fault Tree Module for Loss of Power from a Cross-Tied Source

Fault Tree Module for an Electric Power Circuit Breaker (Sheet 1 of 3)

Note: Events COMPIDEN-BXX-CC and COMPIDEN-BXX-CTP may be considered as local faults in some cases.

Note: When event COMPIDEN-BXX-CC is developed, events COMPIDEN-BXX-CTP and COMPIDEN-BXX-AS should be truncated. Operator errors developed under event COMPIDEN-BXX-OE should only include local operation of wrong circuit breaker.

Fault Tree Module for an Electric Power Circuit Breaker (Sheet 2 of 3)

I102

Fault Tree Module for an Electric Power Circuit Breaker (Sheet 3 of 3)

COMPIDEN =
 XX =
 COMPIDCL =
 VLT =
 COMPIDBS =
 COMPIDFS =
 COMPIDCB =
 XO =
 COMPIDPS =

Fault Tree Module for a Diesel Generator (Sheet 1 of 2)

Fault Tree Module for a Diesel Generator (Sheet 2 of 2)

COMPIDEN =
 SYST~~T~~ =
 TYP =
 MECFATSTATDES =
 FLST =

COMPIDCL =
 VDC =
 COMPIDBS =
 COMPIDFS =
 COMPIDCB =
 XO =

1105

Fault Tree Module for a Battery

COMPIDEN =
VLT =
COMPIDDC =
ROOMRRR =
COMPIDCL =
COMPIDCB =

Fault Tree Module for Electric Power Faults, Short Term (Sheet 1 of 5)

COMPIDEN =
 COMP =
 TYP =
 -T- =

Fault Tree Module for Electric Power Faults, Short Term (Sheet 2 of 5)

Fault Tree Module for Electric Power Faults, Short Term (Sheet 3 of 5)

Fault Tree Module for Electric Power Faults, Short Term (Sheet 4 of 5)

I110

Fault Tree Module for Electric Power Faults, Short Term (Sheet 5 of 5)

1111

Fault Tree Module for Loss of Power at a Bus, Short Term

Fault Tree Module for Loss of AC Power Sources from a Bus, Short Term (Sheet 1 of 2)

COMPIDEN =
VLT =
COMPIDDG =
COMPIDAC =

Fault Tree Module for Loss of AC Power Sources for a Bus, Short Term (Sheet 2 of 2)

COMPIDAC =
VLT =
COMPIDEN =
COMPIDXF =
VAC =
COMPIDCL =

COMPIDRC =
COMPIDMG =
COMPIDSW =
COMPIDCB =
XO =

1114

Fault Tree Module for Loss of DC Power Sources for a Bus, Short Term

COMPIDEN =
VLT =
COMPIDDC =
COMPIDBT =
COMPIDCL =
VDC =
COMPIDMG =
COMPIDIN =

COMPIDCB =
XO =
COMPIDSW =

1115

Fault Tree Module for Loss of Power from a Cross-Tied Source, Short Term

COMPIDEN =
VLT =
COMPIDCL =
COMPIDXT =
COMPIDCB =
XO =
COMPIDSW =

Fault Tree Module for an Electric Power Circuit Breaker, Short Term (Sheet 1 of 3)

COMPIDEN -

XX -

Note: Events COMPIDEN-BXX-CC and COMPIDEN-BXX-CTP may be considered as local faults in some cases.

Note: When events COMPIDEN-BXX-CC are developed, events COMPIDEN-BXX-CTP and COMPIDEN-BXX-AS should be truncated. Operator errors developed under event COMPIDEN-BXX-OE should only include local operation of wrong circuit breaker.

Fault Tree Module for an Electric Power Circuit Breaker, Short Term (Sheet 2 of 3)

I118

Fault Tree Module for an Electric Power Circuit Breaker, Short Term (Sheet 3 of 3)

Fault Tree Module for a Diesel Generator, Short Term (Sheet 1 of 2)

Fault Tree Module for a Diesel Generator, Short Term (Sheet 2 of 2)

COMPIDEN =
 SYST =
 TYP =
 MECFATSTATDES =
 FLST =
 COMPIDEN =
 VDC =
 COMPIDEN =
 COMPIDEN =
 COMPIDEN =
 XO =

Fault Tree Module for a Battery, Short Term

COMPIDEN =
VLT =
COMPIDDC =
COMPIDCL =
COMPIDCB =

Fault Tree Module for a Component Control Circuit (Sheet 1 of 5)

COMPIDEN =
 COMP =
 TYP =
 -T =
 ROOMRRR =
 XX =

Fault Tree Module for a Component Control Circuit (Sheet 2 of 5)

COMPIDEN =
 COMP =
 -T =
 FAULTTYPN =
 NN =
 FAULTTYPD =
 MM =
 FLC =
 FLO =

Fault Tree Module for a Component Control Circuit (Sheet 3 of 5)

COMPIDEN =
 COMP =
 -T =
 FAULTTYPE =
 NN =
 FLT =

Fault Tree Module for a Component Control Circuit (Sheet 4 of 5)

Fault Tree Module for a Component Control Circuit (Sheet 5 of 5)

Fault Tree Module for a Control Circuit Network (Sheet 1 of 3)

Note: Truncate event CTP-COMPIDEN-TNN if control power has already been treated in the circuit-level module.

Fault Tree Module for a Control Circuit Network (Sheet 2 of 3)

NOTE: The NN used in the -TNN substitution must be used in the NN substitution that follows.

Fault Tree Module for a Control Circuit Network (Sheet 3 of 3)

1131

Fault Tree Module for a Control Circuit Signal Path (Fails as an Open Circuit)

COMPIDENTSGPT -
COMPIDEN -
-T -
SGPT -
COMPIDCL -

COMPIDVS -
WO -
CONTAKPIDEN -
XO -
COMPIDMS -
YO -
RELAYCLIDEN -
MAGSTRIDEN -

Fault Tree Module for a Control Circuit Signal Path
(Fails as a Complete Circuit)

Fault Tree Module for a Contact Pair (Sheet 1 of 2)

CONTAKPIDEN -
 XX -
 SGPT -
 COMPIDEN -
 COMP -
 FAULTTYPE -
 TYP -
 -T -
 NN -
 FLT -

I134

Fault Tree Module for a Contact Pair (Sheet 2 of 2)

CONTACTPIDEN =

XX =

ACTUATION SYS FAULT SGPA OCCURS =

ASY =

FLT =

SGPA =

Fault Tree Module for Component Actuation System Faults (Sheet 1 of 3)

COMPIDEN =
 COMP =
 TYP =
 -T =
 ROOMRRR =

Fault Tree Module for Actuation System Faults (Sheet 2 of 3)

COMPIDEN =
 COMP =
 TYP =
 ASY =
 OUTS =

Fault Tree Module for Actuation System Faults (Sheet 3 of 3)

COMPIDEN =
 COMP =
 TYP =
 ASY =
 OUTS =

Fault Tree Module for Actuation System Rule 1 (Sheet 1 of 4)
(M Applications Per Node)

Fault Tree Module for Actuation System Rule 1 (Sheet 2 of 4)

Fault Tree Module for Actuation System Rule 1 (Sheet 3 of 4)

LOGICTYPE =
 COMPIDLG =
 COMPIDCL =
 VLT =
 COMPIDBS =

COMPIDFS =
 COMPIDCB =
 XO =
 COMPIDPS =

I142

Fault Tree Module for Actuation System Rule 1 (Sheet 4 of 4)

LOGICTYPE =		LOGIC UNIT COMPIDEN =		FLT =	
OUTPUT SIGNAL PATHS		INPUT SIGNAL PATHS			
GENERIC PATH ID	SUBSTITUTION FOR: SGPT	GENERIC PATH ID	SUBSTITUTIONS FOR:		
			SGPT	ACTUATION SYS FAULT INNI (TO INMN) OCCURS	FLI TO FLN
OUT1 OUT2 OUT3 OUT4 OUT5 OUT6 OUT7 OUT8 OUT9 OUT10 OUT11 OUT12 OUT13 OUT14 OUT15 OUT16 OUT17 OUT18 OUT19 OUT20		INN1 INN2 INN3 INN4 INN5 INN6 INN7 INN8 INN9 INN10 INN11 INN12 INN13 INN14 INN15 INN16 INN17 INN18 INN19 INN20			
		BOOLEAN			

C-134

1143

Fault Tree Module for Actuation System Rules 2 and 3
(M Applications Per Node)

ACTUATION SYS FAULT OUT1 OCCURS =
ASY =
FLT =
INN1 =
OUT1 =
:
OUTM =

1144

Fault Tree Module for Actuation System Rule 4
(One Application Per Node)

Fault Tree Module for a Actuation System Signal Path

```

      ASY =
      SGPT =
      COMPIDVS =
      XO =
      COMPIDMS =
      YO =

```

```

COMPIDCA =
COMPIDCD =
COMPIDCU =
COMPIDBT =
      ZO =

```

Fault Tree Module for an Actuation System Signal Path (Sheet 2 of 3)
(Fails as an Open Circuit)

ASY =
SGTP =
COMPIDSR =
COMPIDSO =
COMPIDST =

COMPIDSF =
COMPIDSL =
COMPIDSP =
COMPIDSX =

Fault Tree Module for an Actuation System Signal Path (Sheet 3 of 3)
(Fails as an Open Circuit)

ASY =
SGPT =
COMPIDCL =
VLT =
COMPIDBS =
COMPIDFS =
COMPIDCB =
NO =
COMPIDPS =

I149

Fault Tree Module for an Actuation System Signal Path
(Fails as a Complete Circuit)

ASY =
SGPT =
COMPIDVS =
XC =
COMPIDBT =
ZC =
COMPIDMS =
YC =

1151

Fault Tree Module for Operator Error

Fault Tree Module for Component Unavailability Due to Test or Maintenance

1153

Fault Tree Module for Control Circuit Unavailability Due to Test or Maintenance

Fault Tree Module for Actuation System Signal Path Unavailability Due to Test or Maintenance

Fault Tree Module for Component Unavailability Due to Unscheduled Maintenance,
Scheduled Maintenance, or Test (Sheet 1 of 3)

COMPIDEN -
TYP -
COMPIDEA -
:
COMPIDEB -
SPR1 -
SPRX -
:
SPRN -
SPRY -
-T -

S1 -
:
SN -
COMP -
-N -
-A -
-B -

Fault Tree Module for Component Unavailability Due to Unscheduled Maintenance,
Scheduled Maintenance, or Test (Sheet 2 of 3)

Fault Tree Module for Component Unavailability Due to Unscheduled Maintenance,
Scheduled Maintenance, or Test (Sheet 3 of 3)

APPENDIX D

RELAY ACTUATION AND COMPONENT CONTROL MODULES

D.1 Introduction

The Relay Actuation and Component Control (RACC) modules presented at the end of this Appendix (following the Figures used in the example problem) were designed for modeling component control or actuation circuits which utilize relays and relay contact pairs. They are intended to allow the user to model the circuit directly from the component control or actuation electrical schematics by identifying the electrical component faults that would lead to the mechanical component fault state contained in the top event of the RACC modules. The component control modules (CC1 through CC5) give the user as much freedom as possible in modeling a component control circuit and identify for the user the typical control devices seen in component control circuits. Provisions are also made for the user to define and describe a non-typical control device where encountered. The component control modules lead the user to the basic events contributing to the control circuit failure or allow the user to transfer into a set of actuation system modules (CC6 through CC11). These actuation system modules are then used to model the basic failures contributing to the component failure. It should be noted that, in many cases, system initiation or actuation circuits whose purpose is to automatically initiate the system when a predefined set of plant parameters have exceeded limiting values, may actuate more than one of the system components. Because of this, the user is cautioned to comply with the naming scheme derived for the module events to ensure that individual component failures resulting from the independent failure of a given contact pair on a master system actuation relay are identified as well as failure of all of the components actuated by the relay should the relay itself fail. Provisions also exist for the user to identify component or system failures resulting from a measured parameter, affecting the component control, being out of tolerance due to a failure in a different system. The utilization of each of the modules will be explained in greater detail later in this text.

D.2 General Precautions

Every effort has been made to prevent the user from creating unnecessary errors while developing a fault model of a control or actuation circuit; however, there are some areas in which the responsibility must be left to the user. This includes a thorough review of this appendix to gain an overall understanding

of the use of the modules before attempting to create a fault tree.

Before attempting to model system control or actuation circuits, first review all of the system and component control schematics to obtain a general understanding of the system operation and identify all of the control components which are common to more than one system. For example, one water level instrument may provide initiation signals for more than one system. Another example could be a valve position indicator in one system may interlock a pump or valve in another system. In the system being modeled, each of the interfaces must be identified to ensure consistent and correct models. Having completed this step, the various analysts should then compare their list of interfacing or interdependent components with the analyst modeling the other systems. This will help identify all system dependencies and interfaces. When this is completed, decide upon a consistent naming scheme to be utilized by all analysts. The descriptions for utilization of each of the modules are quite specific; however, if this step is not accomplished, component naming errors will continue to plague the effort throughout the entire PRA process. When the identification scheme has been decided upon, enter the code name for each component, system, etc., on the list for future reference during the fault tree construction.

In developing an identification or naming scheme, it is generally helpful to utilize the component identifiers utilized in the control and actuation schematics. This assists in ensuring consistency and also makes it easier for non-PRA experienced personnel to understand the fault trees and the cut sets derived from the PRA.

D.3 Overview of Use

To utilize these modules correctly, the analyst must first determine the failure modes of the system and system components he is to model. After developing the system fault model, he then collects the control and actuation schematics for these components and systems. Working directly from the schematic diagrams, the analyst then identifies the component control circuits that can cause the specific component failure modes determined earlier. It is important to model the component control circuitry before attempting to model the system actuation circuitry. This allows the analyst to determine which components have common actuation circuits, lessening the potential for inconsistent naming of the same component and duplicate modeling of the same actuation circuitry. The component control circuitry is modeled by first dividing the component control circuitry into functional networks. For example, a motor driven pump control circuit would be divided into a start network and a stop network.

These networks are then further subdivided into signal paths that constitute the particular network being modeled. This can be accomplished by redrawing the individual networks or by highlighting the networks of concern on the schematic diagram. All of the control networks associated with each component in the system are then sequentially numbered. Each network is then subdivided into signal paths. These signal paths are then sequentially numbered within each network. Finally, the components within each signal path are assigned a unique identification code. It is suggested that the identification code utilized be as similar as possible (identical, when possible) to the plant identification code on the schematic diagram. This approach makes the fault tree model much easier to review and reduces the effort involved in associating fault tree events with the component being modeled. This is particularly helpful if corrections to the fault tree are required or if additional analysis (seismic, fire, etc.) are to be accomplished later. During this control component identification process, it is also advised that all system actuation components common to more than one component control circuit be identified and uniquely named. This will prevent naming errors from occurring when the system actuation circuitry is modeled.

After the mechanical component control networks, signal paths, and electrical components have been uniquely identified, the control circuits may now be modeled. This is accomplished for each component by using the modular fault tree that interfaces with the component (Module CCl). The events on the module not applicable to the component being modeled are deleted. The remaining events now provide an as yet unnamed logic structure of the component control networks being modeled and the signal paths associated with each network. The module at this state is generic and must now be associated with the specific component being modeled. This is accomplished by making substitutions for the generic character strings associated with each event remaining on the module. All character strings requiring substitution are listed on the module. The analyst first deletes those character strings which exist only within the events that were deleted in the first step. Character strings that are common to several events should only be deleted if all events containing that string are deleted. The character string substitutions may now be made. A list containing guidelines for determining acceptable substitutions for each of the character strings on the module are contained in Table D.1. The appropriate substitutions from Table D.1 are entered on the module completing CCl of the component control circuit modules. The completed module should now uniquely identify the component, its control circuit networks and the signal paths associated with each network. If a power failure to only one of the control networks is applicable to the component failure mode, the control

circuit network power failure module (CC2) is next to be completed. The appropriate substitutions are made as with module CC1. These two modules should now provide a completed model of the component control logic and control power failures applicable to the system mechanical failure. Next, the electrical component failures that specifically contribute to the control circuit failure are modeled. A separate electrical component failure module (CC3) is utilized for each network signal path identified on module CC1. Again, the non-applicable component failures are deleted and the appropriate substitutions are made as described above. Module CC3 identifies components normally encountered in control circuits and allows defining or describing other components that may be in the circuit.

Module CC4 allows the user to model various instrumentation failures identified on CC3 that could contribute to the component failure. Module CC5 allows for modeling the various process switches identified on CC3 that could also contribute to the component failure.

In modules CC3 and CC4, there are events that allow the user to transfer from the component control circuit to a component actuation or initiation logic circuit. Modules CC6 through CC13 provide the generic modules for these types of circuits.

Utilizing the relay actuation system modules is very similar to the control circuit modules. The actuation circuit is first divided into networks and then the networks are further divided into signal paths. Each network, signal path, and signal path component is uniquely named, then the appropriate network, signal path and component modules are used to construct a fault tree of the system actuation circuitry. The aspects of the relay actuation system modules that differ from the control circuit modules are:

1. The modules are designed to allow the analyst to use the same set of generic modules to transfer to another actuation system network. This allows the analyst to utilize the same generic modules over and over until each network in the relay actuation system is uniquely modeled.
2. Network logic structures commonly encountered (1 of 2 Twice, AND, OR) are already constructed. This eliminates the necessity for the analyst to construct these modules from signal path modules.

D.4 Module CCl

The first component control module to be used by the analysts is the CCl module of this Appendix. The top event of this module is designed to link with the mechanical component failure modules for motor operated valves (MOVs), safety relief valves (SRVs), motor driven pumps (MDPs), turbine driven pumps (TDPs), etc. This module then allows the user to develop a fault model for the component control circuit. First, the control circuit schematic is subdivided into networks. Each network should be a part of the control circuit designed to accomplish a specific purpose in controlling the component. For example, a motor driven pump control circuit would have both a start network and a stop network. A remotely operated MOV would have an opening network and a closing network. Each network identified by the analyst is then evaluated to determine if a fault in the network can contribute to the mechanical fault identified in the top event of the module. Continuing with our example of an MOV, suppose the fault is "fails closed" for a normally closed valve that is supposed to automatically open and remain open in response to a given transient. In this case, both a fault in the MOV opening network and a fault in the MOV closing network could cause the valve to fail closed (i.e., fail in its original state). Therefore, both networks should be included in the fault model. If the valve is a normally open valve, that is supposed to remain open during the transient, then only a fault in the close network could contribute to the valve failing closed. In this case, only one network should be included in the fault model. Once each network is identified, then each network is sequentially numbered. Assuming a system with 1 pump and 3 MOVs, where both start and stop networks for the pump, both the open and close networks for two of the valves, and only the close network for the remaining valve are identified as potential contributors to the system fault, there would be 7 networks numbered 01 through 07. Each of these networks is then broken down into signal paths. For example, the close network for one of the MOVs may have separate local, remote, and automatic actuation paths. If this was network 07, then the signal paths for the 07 network would be numbered 0701, 0702, and 0703. A failure which would complete any one of these paths, energizing the closing coil contacts, would cause the valve to close.

The analyst must also determine if a ventilation system failure could cause the control circuit to malfunction and what test or maintenance actions could contribute to the valve unavailability. Finally, the analyst must evaluate a loss of power to the control circuit to determine whether or not this could contribute to the component failure. Once all of the fault potentials have been identified, the analyst then deletes the non-applicable events from the module.

The second step is to substitute plant specific names and descriptions into the remaining events. Each of the character strings that must be substituted for are listed on the module. First, those character strings which are no longer applicable should be deleted from the list. These would be event specific character strings that are inappropriate because the event they correspond to has been deleted from the module. The remaining character strings are then substituted for using the guidelines for this module from Table D.1 of this appendix. For example, returning to a MOV designated F016, suppose the fault was a normally open valve that failed to remain open with three separate signal paths capable of contributing to the fault. All of the events except for failures in the closing network would be deleted leaving only the following substitutions to make:

SYST	TYP	COMPIDEN	NTWK1PRPS	NX	NP
FLST	XF1	XF2	XF3	MECFALTSTATDE	

From Table 1, the following substitutions should be made. Assuming the model was of the high pressure injection system (HPI):

SYST - HPI
TYP - MOV
COMPIDEN - F016
NTWK1PRPS - Closing
MECFALTSTATDES - To Remain Open
FLST - PTR0
NX - 07
NP - CL
XF1 - CMP
XF2 - CMP
XF3 - CMP

In addition, gates COMPIDEN-NPNX-SP and COMPIDEN-NPNY-SP must be designated as AND or OR gates.

D.5 Module CC2

The second control circuit module is used to identify electrical component failures that could contribute to a loss of power to a given control circuit network. This particular module would be used in cases where different control circuit networks have independent power supplies or when a power failure contributes to a fault in only one network. In these cases, the electrical component faults contributing to a power failure to the network are identified, the non-applicable events are crossed out and then the remaining event character strings are substituted for using the guidelines in Table D.1. The analyst must take care to ensure the top event of this module exactly matches the developed

event in Module CC1 that it is suppose to link to. If this is not accomplished, an error will occur when an attempt is made to link the modules.

D.6 Module CC3

Component Control Module 3 is used to identify electrical component failures in the control circuit which could lead to the mechanical component fault described in the top event. A separate CC3 module must be used for each signal path in each component control network. In each case, the analyst must ensure that the top event in CC3 identically matches, both in name and description, the developed event it is suppose to link to in Module CC1. The user must, in addition to identifying the component's faults that could contribute to the mechanical component fault, define several gates as AND or OR gates. These gates are noted on the generic modules by squares (special gates) instead of AND or OR Boolean logic symbols. These gates are the top gates: gate AI-COMPIDEN-NNON, and gate PS-COMPIDEN-NNON.

On this sheet, many electrical components normally encountered in a component control circuit have been identified; however, there are instances where a component is used for which no corresponding event can be found on Module CC3. In this case, modules CD-CMOTIDNM-NNON and TY-INTYIDTTT-TF are provided. The analyst must name and describe these events using the guidelines in Table D.1.

In this module, there is a differentiation between an analog instrument and a process switch. This is to allow the analyst to easily differentiate between a contact pair, in a component control circuit, that is driven by an instrumentation channel and a contact pair that is an integral part of a process sensor. The process switch branch of the module is used in cases where a detector, such as a level detector, contains a set of contacts that are integral to the detector where the contact pair is directly in the control circuit signal path. In cases where the contact pair is driven by another relay or off of a bistable or recorder in instrumentation channel, either the relay or analog instrument branch of the module would be used. All three branches of course may be used simultaneously depending upon the circuit configuration.

The relay contact pair fault branch contains events that allow the user to transfer from the individual component control circuit to a system actuation circuit. This branch would be fully developed when a contact pair in a component control circuit was actuated by a relay outside of the control circuit. This would be the case where a pump is designed to start and a valve to open on a low level signal. Both components are

controlled by the same process parameter; however, different contact pairs on a master relay are used to initiate each component. Users are cautioned to ensure that the SYST-RANN-FLT event is correctly substituted for in these cases so that all contact pairs driven from a master relay link to the appropriate actuation system module developed with Module CC6.

D.7 Module CC4

Module CC4 is used to model analog instrument channel faults contributing to a control circuit failure. As before, the analyst must ensure the top event in the module is correctly substituted for to ensure it identically matches the developed event it is supposed to link to in Module CC3. In this module, the word NAME is substituted to describe the instrument channel type such as flux, temp, or rad. These are already on Module CC3. In cases where the instrument channel is a type other than one listed above, appropriate description of the instrument type would be entered such as PRES for pressure, LVL for level, etc., on both Module CC3 (for TYPE) and Module CC4 (for NAME). A thorough set of substitution guidelines for each case is provided in Table D.1. As before, the analyst deletes the non-applicable events from the module and makes the appropriate substitution for the remaining character strings.

D.8 Module CC5

Module CC5 allows development of the various process switch failures identified in Module CC3. A separate CC5 module must be developed for each switch identified on Module CC3. As before, the user is cautioned to ensure the top event on Module CC5 is correctly modified to ensure it identically matches its corresponding developed event on Module CC3.

D.9 Module CC6

Module CC6 marks the beginning of the system actuation modules. All modules from this one through CC12 are used to model system actuation circuits. Module CC6 provides the analyst with a set of events capable of modeling the various logic structures normally encountered as well as a loss of power to the actuation circuitry. In addition, an event is provided to allow the analyst to model a logic structure not listed above. The analyst must define the top gate of this module as AND or OR. After the appropriate events are deleted from the module, careful substitutions must be made to ensure that this module's top event exactly matches its corresponding developed event on Module CC3 or Module CC4.

D.10 Module CC7

Module CC7 provides the analyst with a preconstructed 1 of 2 twice logic fault module. None of the gates on this module should be deleted; only character string substitutions are necessary. The signal paths in the network have been pre-numbered as shown in the following diagram.

1 of 2
Twice
Numbering
Scheme

D.11 Module CC8

Module CC8 provides the analyst with a logic structure for developing several network signal paths in an AND or OR logic structure. The non-applicable number of signal paths are deleted by the analyst and the top gate defined as AND or OR. The appropriate substitutions are made as before. In making the substitutions either AND/ or /OR should be deleted for clarity.

D.12 Module CC9

Module CC9 is the system actuation circuitry equivalent of Module CC3 for the control circuits. It identifies the various electrical components likely to be encountered in an actuation network. It should be used the same as CC3.

D.13 Module CC10

Module CC10 is the actuation system equivalent of CC4 in the control circuit modules. It is developed and utilized the same as CC4.

D.14 Module CC11

Module CC11 is the actuation system equivalent of CC5 for the control circuit modules. It is developed and used the same as Module CC5.

D.15 Module CC12

Module CC12 is provided to allow control circuit malfunctions resulting from a loss of cooling to be developed.

D.16 Example

The following discussion demonstrates how the RACC modules are used in modeling a system component controls and actuation. For purposes of illustration, assume the existence of a Hypothetical Injection System (HIS). The HIS is shown in Figure D-1. It consists of a motor driven pump (C001) and a motor operated valve (MOV) designated as F010. The system numeric identification code is 96. This system provides for an automatic restoration of system 75 tank level should it decrease below 200 inches.

The control and actuation circuits (Figures D-2 through D-5) for this example are simplified representations of typical circuits found in existing plants. They are not developed to the same level of detail normally seen in a plant schematic. These drawings, however, contain all of the information necessary to understand the application and use of the modules.

The HIS normal lineup is with MOV F010 closed and C001 in standby. If the system 75 tank level decreases below 200 inches, pump C001 is to auto start and valve F010 is to auto open. When the tank level is restored above 220 inches, MOV F010 is to auto close and pump C001 is to automatically trip on low flow. The control circuit for C001 is shown in Figure D-2. There is a local and remote operating station for the pump. The controlling station is selected by S1. S2 and S3 are the remote and local manual control switches, respectively. The K2 relay contacts (T1 and T2) in the CC network provide the auto start signal for C001. The TC network stops the pump. As with the starting network, there are provisions for local or remote manual control. In addition, there are two pump protection trips. These are a low suction pressure trip (K4) and a low flow trip (K10) which automatically secures the pump if a system low flow is indicated in conjunction with a normal pump running discharge pressure.

The control circuit for F010 is shown in Figure D-3. The valve may be locally or remotely operated. It is automatically opened by the K2 relay and will automatically shut when level is restored in the system 75 tank through the actuation of the K6 contact pair. Torque switches in the open and close networks deenergize these circuits when the valve has reached full travel.

The system actuation logic is shown in Figure D-4. The system is automatically initiated when the 1 out of 2 twice low water level circuit senses a low water level in the tank. This actuates K2

which automatically starts C001 and opens F010. The pump is protected from a low pump suction pressure by three pressure switches arranged in a 2 of 3 logic network. If this circuit is actuated, the K4 relay will trip the pump. The F010 valve will automatically close when the tank level is restored. This signal is provided from the system 75 tank level instrument channels A and C. Block diagrams of these instruments are shown in Figure D-5. The channel A contact pair is driven by a bistable amplifier. The channel C contact pair is switched by recorder indicator. System flow is monitored by two flow switches 96N005 and 96N006. This is a normally energized circuit that deenergizes to operate. This signal also seals in and must be manually reset by pushing the S10 reset switch. The K8 relay in conjunction with the N004 pressure switch contact pair indicates that the pump is running against a shutoff head and causes the pump to stop by actuating the K10 relay.

Assuming the system fault being modeled is failure of the HIS to restore level in the system 75 tank, then the faults associated with each of these components are:

Pump C001	Fail to start Fail to continue to run	Fails deenergized
Valve F010	Fail to open Fail to remain open	Fails closed

The steps in modeling these control and actuation circuits are:

1. Separate each of these circuits into networks,
2. Examine each network for its potential for contributing to the mechanical component fault state,
3. Identify the signal paths in each network that could cause the network to malfunction, and
4. Develop the component faults contributing to the signal path fault.

Throughout this example, the failure of room cooling and test and maintenance events will be left in where applicable, but will not be developed or discussed further as their use is explained in Reference [1].

Steps 1 through 3 can be accomplished with a highlighting pen on the schematic diagrams by marking those network signal paths that will contribute to the component fault. This may also be accomplished by redrawing each of the circuits showing only those networks and signal paths that will be modeled. For this

example, each of the networks have been redrawn in Figures D-6 through D-14. These figures consist of the pump start and stop networks, the valve open and close networks and each of the system actuation networks making a total of 9 separate networks. These networks are numbered 01-09. Each signal path within each network is then numbered from NN01-NNXX. Where NN is the network number. Examples of this numbering system are shown in Figures D-6 through D-14.

In redrawing or highlighting these circuits, only those networks and portions of a network that can contribute to the mechanical fault should be identified. This is shown in Figure D-6 where the local and remote manual control stations have been eliminated from the redrawn pump start network. This was done assuming no fault in either of these signal paths could fail the automatic actuation signal path (0101). A short circuit in either of these paths is assumed to be included in the resulting fuse failure associated with the short. This assumption is debatable, but will be used in this example for illustration. It should also be noted that throughout the modeling process, any relay that must be energized to contribute to a fault is assumed to be incapable of self energization. Therefore, these relays are not included in the redrawn circuits.

Beginning with the pump, there are two faults to consider, failure to start and failure to continue to run. This involves the start network, the power to the start network and the stop network. The first two of these contribute to the failure to continue to run. The control circuit is modeled first starting with module CC1. Both networks could contribute so each network branch is left intact on the module. The power failure only affects the start network so the loss of power fault branch is deleted. In network 01, only 1 signal path is assumed to contribute, therefore, two of the signal paths are deleted below the NTKW1PRPS gate. The loss of power to the network is left intact. In the NTKW2PRPS branch, the power failure is deleted and an additional signal path is added to account for each of the potential signal path faults that could trip the pump. For this example, both the room cooling and test and maintenance events are included but are not discussed further. Finally, the appropriate character string substitutions are made. A completed module is shown in Figure D-15.

Next, the power failure to the network is modeled. This is done using Module CC2. The components on CC2 that could contribute to a power failure to the network are:

1. the transformer (CT on Figure D-3)

2. the control power wiring (designated WIR01 for this example)
3. the control power fuses (F1 and F2 on Figure D-3).

There is no local power supply or circuit breaker so these are deleted. A power failure at the bus is coincident with a power failure to the component and should have been modeled as a loss of power to the component. It is also deleted. An additional fuse event is added to account for F2. Appropriate character string substitutions from Table 1 are then entered. The completed module is shown in Figure D-16.

The individual component faults in each signal path are now modeled using Module CC3. A separate CC3 module must be used for each signal path modeled. In network 01, there is only 1 signal path 0101. The component faults in this signal path are:

K2 contact pair T1 and T2	Fails open
Relay CC	Fails deenergized

The appropriate component faults are deleted from Module CC3 leaving only the relay coil event and the relay contact pair event. The contact pair can fail from either a local fault or from an input signal fault to the K2 relay. These contributors are left intact on CC3 for linking to the actuation circuit modules. The completed module including the appropriate character string substitutions is shown in Figure D-17.

Figures D-18 through D-21 show the completed CC3 modules for each of the stop network (02) signal paths. Referring to Figure D-7, it should be noted that the signal path containing coil TC is not included because it is assumed that the coil cannot energize itself. In the modules for signal paths 0201 and 0202, there are only manual switch events. Note that both the manual switch event and human error event for S1 are repeated in each of these modules because it is a common switch. The individual contact pairs, however, are different. The manual switch events on both of these modules must be reproduced because there are two switch contact pairs in each signal path. In the module for signal paths 0203 and 0204, there is only the relay contact pair event and both of these include input signal faults from the actuation circuitry.

In preparing each of these modules, caution should be exercised in the substitutions to ensure the top event in each module is identical to the module it is supposed to link to in Figure D-15. This includes ensuring both the event description and the event name are identical in both modules.

This completes the modeling of the pump control circuit. The next step is to prepare the valve (F010) control circuit model.

The analyst begins, as with the pump, with Module CC1. The events on CC1 applicable to the valve faults include:

1. open faults in the valve opening network
2. loss of power faults to the valve opening network
3. faults that would complete the closing network when the valve is open

To model these faults, NTWK1PRPS network is arbitrarily designated as the open network and the NTWK2PRSP network is designated as the close network. Power failures apply only to the open network, therefore, control circuit power failure events are deleted. The open network has only 1 signal path; thus, the NN02 and NN03 signal path events are deleted below the NTWK1PRPS event. A power failure does not apply to the close network, therefore, the NTWK2RPS power failure event is deleted. The character strings associated with the remaining events are substituted for at this point. A completed module is shown in Figure D-22.

The power failure to open the network is modeled next using module CC2. The applicable faults are:

1. a fault in the power supply cable (WIR02)
2. a fault in the fuses (F1 and F2)
3. a fault in the circuit breaker
4. a loss of power at power panel PP301

The transformer and local power supply faults are deleted and the appropriate substitutions made. As before, an additional fuse failure event was added to the module to account for F2. A completed module is shown in Figure D-23.

Next, the signal path component faults are modeled. Signal path 0301 contains the following components:

1. a torque switch (TSC)
2. a manual switch (S4L)
3. a relay contact pair (K2, T3-T4)
4. the open relay

The relay contact pair (K2, T3-T4) can fail both as a local fault and due to fault in the control signal to the K2 relay so the actuation signal developed event is left intact. A completed module is given in Figure D-24.

The valve closing network (NN=04) signal path faults are developed in Figures D-25 through D-27. A completion fault through any one of these signal paths will cause the valve to fail closed. It should be noted that the torque switch (TSC) in Figure D-3 is not reproduced in Figure D-9 nor included in Figure D-28. It is assumed that any fault in this switch could effectively contribute only to the valve failing to close as the switch is normally made up when the valve is open.

This completes the modeling of the component controls for this system. The actuation circuits for the system must now be modeled. First, the preceding modules must be reviewed to identify any contact pairs that have common relays. In this example, only K2 has multiple contact pairs. These are T1-T2 in the pump start network and T3-T4 in the valve open network. The relay system actuation events in Figures D-17 and D-24 for the pump and the valve should be identical to ensure both proper linking of the actuation circuit module and to prevent modeling the same actuation circuit twice.

Before beginning to model the networks specifically, it is advised that the power supply to the actuation circuits be evaluated to determine if a common power source exists. In this example, it is true, and the power failure is modeled only once.

The actuation circuit modeling begins with network 05, the HIS initiation network shown in Figure D-10. Module CC6 is used to begin the modeling process. The applicable failures on CC6 are:

1. loss of power
2. failure of 1 out of 2 twice logic
3. fault in signal path 0506

The remaining events are deleted and the appropriate substitutions are made. The completed module is shown in Figure D-28.

The next module to be used is CC7. No deletions need to be made from this structure. It is already in its simplest Boolean form. Therefore, only character string substitutions are required. Note that each character string identified on the sheet must be substituted for. A completed module is presented in Figure D-29.

Each signal path component fault is now modeled using module CC9. This module is similar to CC3 and is used in exactly the same manner as CC3. A completed CC9 module for each of the signal paths 0501-0504 and 0506 is presented in Figures D-30 through D-34. Note that the only component fault remaining in each of these figures is the level switch event. These events are

developed with the CC11 Module. For this example, the faults identified include the contact pair fault, the sensor fault, the sensing line fault and the test and maintenance fault. Completed modules for each switch are given in Figures D-35 through D-38.

Module CC9 is also used to develop the K2 relay fault in signal path 0506. This module is shown in Figure D-34. It links to the 0506 signal path fault in Figure D-28.

The pump low suction pressure network is the next to be modeled. It links to Figure D-20. Module CC6 is again used to transfer from the component control module to the system actuation modules. This network uses a 2 of 3 logic, which is developed by using the AND/OR LOGIC event. The completed module is shown in Figure D-39.

Module CC8 is used to complete the 2 of 3 logic structure model. The top event is designated as an OR gate and three signal path events are left intact. A completed module is shown in Figure D-40. The signal path modules (CC9) are used to develop the signal path component faults (Figures D-41 through D-43). These are all process switch faults that are developed using CC11. Note that each switch must be developed twice because two different contact pairs are used for each switch. Completed modules for each of these are shown in Figures D-44 through D-49.

The F010 high level auto close network 07 is next to be modeled. A failure that completes either signal path 0701 or 0702 will cause F010 to close. The network modeling process begins again with module CC6. The events remaining on the module are the top event and the AND/OR logic event. The AND/ should be deleted from the module to make it read as OR logic. A completed module is shown in Figure D-50. This module should link with the CC3 module shown in Figure D-27.

Figure D-51 shows the completed AND/OR logic network module. All but two of the signal paths have been deleted. A signal path for the relay (K6) is not included because it is assumed that the K6 relay may not self energize. The remainder of the network modeling for the flow instruments is shown in Figures D-52 through D-55.

The next network to be modeled is the pump low flow trip network 09. This is an AND network with a single signal path. The modeling process begins with the CC6 network as before. The top gate on this module will link to the K8 contact pair actuation event in Figure D-21. Because it is a single signal path, only the signal path component fault event on the module is left intact. A completed module is shown in Figure D-56. The signal path component module CC9 is then completed (Figure D-57) for the

K8 relay contact pair and the N004 pressure sensor. The completed pressure switch module (CC11) is shown in Figure D-58. Note that sensing line faults for the pressure sensor are not included in the model. This is because it is assumed that any sensing line fault could not result in a false high pressure signal indicative of the pump running. The K8 signal faults will now be developed.

Modeling the HIS low flow monitoring network 08 (Figure D-13) requires particular care because this circuit is a deenergized-to-operate network unlike the other actuation circuits. The development of the logic structure is done using modules CC6 and CC8. In addition, a power failure to the network must be considered. In examining a power failure, however, it is discovered that this would also affect the K10 network preventing it from operating. Thus, a power failure need not be included in the model. One additional observation concerning the power supply to this network and the K10 network involves a power restoration (or spike) to the actuation circuits during system operation. The relay timing could result in a pump trip if power to the actuation circuits was lost and then restored. The exact consequences of this type of occurrence would require extensive investigation beyond the scope of this discussion and will not be addressed further.

The modeling process begins, once again, with CC6. The events remaining on the module are the top event, the AND/OR gate, and the signal path event. The top gate is identified as an OR gate. A completed module is shown in Figure D-59.

Next, the CC8 module is modified to provide the signal path logic structure needed for signal paths 0801 and 0802 using module CC8. The top gate is noted as an AND gate, and two of the signal paths are left intact (Figure D-60).

The signal path component fault models are developed using Module CC9. These are open or deenergized faults which are different from preceding applications of the CC9 module. In all other aspects, however, the model use is the same. Completed examples are given in Figures D-61 through D-63. For two of these, process switch faults are developed using CC11. Completed modules are shown in Figures D-64 and D-65.

Table D.1
Character String Substitution Guidelines

CC1

Code	Character Limit	Meaning	Substitutions/Examples
SYST	4	System Abbreviation	RCIC, HPCI, HPS
TYP	3	Component Type	MOV - Motor Operated Valve PMP - Pump MDP - Motor Driven Pump
COMPIDEN	8	Component ID code	Valve F013 - F013 (may contain system code) i.e., HPCS code is 51 to HPCS valve F015 could be 51F015
ROOMRRR	7	Room Identification	131 for Room 131 NWCC - Northwest corner cubicle
VLT	3	Block name voltage code	480, 64k
NTWK1PRPS NTWK2PRPS	9	Control circuit network purpose description	opening closing starting stopping
NP NQ	2	Network purpose block name code	OP - open CL - closing ST - starting SP - stopping
NX NY	2	Network numerical ID code	01, 12, etc.
COMPONENT ID CODES			
XFMRID	6	Transformer ID code	
CABLID	6	Cable ID code	
FUSEID	6	Fuse ID code	
CKBRID	6	Circuit breaker ID code	
PWRSID	6	Power Supply ID code	
COMPIDBS	8	Electrical Bus ID code	

Table D.1 (Continued)

Character String Substitution Guidelines

CC1 (Concluded)

Code	Character Limit	Meaning	Substitutions/Examples
FAULT STATE CODES			
MECFALTSTATDES	14	Mechanical fault state description	open to remain open closed to remain closed start to run
FLST		COMPIDEN block name failure code	FTRO - fail to remain open FTS - fail to start FISP - fail to stop
PX	2	Network block name	OP - open circuit
FY		failure code	SH - short circuit EN - energized DE - deenergized
Component block name failure codes			
TF	2	Transformer	OP - open
WF	2	Cable/wire	SH - shorted
FF	2	Fuse	CL - closed
BF	2	Breaker	CM - completed
SF	2	Power Supply	DE - deenergized
XF1	3	Network signal path	OPN - open
XF2	3	block name failure	CMF - completed
XF3	3	codes	CLO - closed
YF1	3		DEN - deenergized
YF2	3		ENG - energized
YF3	3		

Table D.1 (Continued)

Character String Substitution Guidelines

CC2

<u>Code</u>	<u>Character Limit</u>	<u>Meaning</u>	<u>Substitutions/Examples</u>
NN	2	Network identification number used in CC1	
NP	2	Same as NP or NQ on CC1	
NTWKNPRPS	9	Network Purpose description used in module CC1	
COMPIDEN	8	Same as CC1	
COMPONENT ID CODES			
XFMRID	6	Same as CC1	
CABLID	6	Same as CC1	
FUSEID	6	Same as CC1	
CKBRID	6	Same as CC1	
PWRSID	6	Same as CC1	
VLT	3	Same as CC1	
COMPIDBS	8	Same as CC1	
COMPONENT FAULT CODES			
TF	2	Same as CC1	
WF	2	Same as CC1	
FF	2	Same as CC1	
BF	2	Same as CC1	
SF	2	Same as CC1	