AVSYS, A Computer Program for Fusion Systems Availability Calculations

Z. Musicki and C.W. Maynard

February 1984

UWFDM-531

FUSION TECHNOLOGY INSTITUTE

UNIVERSITY OF WISCONSIN

MADISON WISCONSIN

AVSYS, A Computer Program for Fusion Systems Availability Calculations

Z. Musicki and C.W. Maynard

Fusion Technology Institute
University of Wisconsin
1500 Engineering Drive
Madison, WI 53706

http://fti.neep.wisc.edu

February 1984

UWFDM-531

AVSYS, A COMPUTER PROGRAM FOR FUSION SYSTEMS AVAILABILITY CALCULATIONS

Z. Musicki and C.W. Maynard

Fusion Engineering Program
Nuclear Engineering Department
University of Wisconsin-Madison
Madison, Wisconsin 53706

January 1984

UWFDM-531

Introduction

The computer code AVSYS has been developed to study the availability of systems, specifically fusion power plants. We decided to employ the Monte Carlo simulation method in the program. This will enable us to better model the operation of a real system, experimenting with it as it were, and to add useful features for accurate representation. The program has been designed in a modular fashion, so as to facilitate future additions.

The Monte Carlo method has its drawbacks, such as lengthy computing time and the need for biasing (but these are offset by being able to model scenarios that are too complicated for a deterministic approach). Also, a statistical error is introduced into the calculations, which needs to be reckoned with. A careful choice in the number of time steps and histories, biasing constants and random number generator constants is therefore necessary.

At this point, AVSYS can represent a power plant consisting of up to 100 different subsystems. Each subsystem can comprise 20 identical units (these numbers are easy to revise, provided the computer memory permits it). These subystems can be interconnected in the logic diagram (or the success tree) of the system, by using up to 100 logic gates (AND,OR).

It should also be mentioned that the code can be used to model individual subsystems. In this capacity, it might be employed as a tool for developing strategies to improve the subsystem design from a reliability standpoint.

The program can take into account the following options: multiple units of a subsystem, per demand and per hour failure rates, redundancy (active and passive, i.e., spares) in any subsystem, deferred repair option for any subsystem (i.e., wait for repair until the system is down, or until the scheduled outage time arrives, whichever comes first; this is useful for inaccessible or radioactive components), specified scheduled maintenance period of the plant.

Before utilizing the code, the user analyzes the system and draws the success tree of it, labeling the subsystems and the logic gates by successive numbers. The gate that describes the system performance, i.e. the output gate, will be the one to look for in the computer printout. A decision is made on the various parameters employed by the code (e.g., the random number generator parameters, biasing parameters). The data is written in the appropriate input files, consisting of failure data, system configuration (logic gates), subsystem options (redundancies, deferred repair, etc). This is described in greater detail below.

The program first reads the subsystem data (such as failure rates, repair times, redundancies, etc.). Then the success tree of the system is read in so that the connections of each subsystem and logic gate are known.

The next part of the program does the calculations for each history and each time step. Every subsystem's reliability is compared to a random number which will decide if the subsystem fails in the particular time step. This is repeated for every operating unit of every subsystem. Also the decision is made if a particular unit is repaired at the end of this time step. This information is used to determine the status of the logic gates in the success tree of the system. Any appropriate data from every time step is saved for future use. In the end, an error analysis is made and pertinent information (availabilities, uptimes, downtimes, number of failures) is printed out.

Code Structure

The code structure is presented in the flowchart, Appendix 1. A short description of each subroutine follows. The flowchart of each subroutine is also appended. The code listing with comments is given in Appendix 2.

MAIN

This is the routine that reads the data and leads to the rest of the computer program. The data read are:

1) Number of subsystems, time step size (in hours), number of time steps per history, number of histories.
2) Number of identical units normally operating for each subsystem.
3) Failure rate (or mean time between failures), repair time, number of standby units, deferred repair option, number of online redundant units, for each subsystem.
4) Number of AND gates, number of OR gates in the success diagram of the system.
5) For each AND gate: number of subsystems, number of AND gates and number of OR gates connected to it.
6) For each AND gate: ID number of subsystems connected to it; ID number of AND gates connected to it; ID number of OR gates connected to it (ID numbers denote the order in which the subsystems and particular kind of logic gates were read in).
7) For each OR gate: number of subsystems, number of AND gates and number of OR gates connected to it.
8) For each OR gate: ID number of subsystems connected to it; ID number of AND gates connected to it; ID number of OR gates connected to it.
9) Number of subsystems with "on demand" failure rates (instead of "per hour" failure rates).
10) ID numbers of such subsystems (then the failure rates read in 3 will be on demand failure rates for such subsystems which will be taken into

account when calculating their reliability).

11) Scheduled shutdown period (in hours) per year.

The input data are also displayed as part of the output.

MAIN then calls subroutine ORDGAT which will save in an array the correct order in which the logic gates have to be calculated. Upon return, it updates the history number and calls MAIN2, from where the main calculations proceed.

The final results are also printed by MAIN. These are: the number of failures, total uptime, total downtime, and the average availability per history of each subsystem; average availability per history of each gate, and time dependent availability of the top gate.

ORDGAT

This routine looks at the input data describing the logical representation of the system, and decides the order in which gates will have to be calculated for each time step. Gates whose input consists entirely of subsystems are to be calculated first. Next are the gates whose input consists of subsystems and the gates already accounted for. This information is stored in an array until all the gates have been thus processed. The subroutine ORDGAT is called only once. The information stored in the array is saved and used at each time step when the status of each gate is calculated.

MAIN2

This routine keeps track of elapsed time and calls other routines in each time step. First, routine SCHED is called to determine if a scheduled maintenance period has been entered; if it has, the routine REPAIR is called; if not the routine FAIL will be called first. FAIL will do the Monte Carlo simulation to determine if a unit of a subsystem has failed. REPAIR will determine if a failed unit is repaired in this time step. Next, routine STATE is called to sort all this information and see if individual subsystems are up or down, based on such consideration as failure and repair of individual units, redundancy, arrival of scheduled downtime period, etc. Subroutine GATE is called next to determine the status of each gate, according to the order laid down in ORDGAT.

After all the time steps in the current history, MAIN2 calculates the cumulative up and down times and subsystem availabilities.

FAIL

Subroutine FAIL determines if a particular unit of a subsystem fails in the current time step. First, it tests to see if the unit is operating at this time (i.e., is up and is not a spare). Then it compares its instantaneous reliability (subroutine RELY) to a random number (subroutine RANDNO). If the former is smaller than the latter, the unit enters the failed state at this time (if it was operational before) and proper flags are set.

REPAIR

This routine will determine if a failed unit will be repaired in this time step. First it checks to see that the unit is in the failed state and if it is that it is an "immediate repair" unit. If these conditions are satisfied, it compares the unit's elapsed downtime to the repair time (from the subroutine REPTIM). This comparison determines the status of the unit at the end of the time period.

STATE

This routine accepts input from the FAIL and REPAIR subroutines and combines it to make a decision on the subsystem status in the system. For failed units, the flag to denote them non-operational is set, and the converse is true for the repaired units. The number of failed units and the number of repaired units of a particular subsystem are combined with the number of redundant units to make a decision on whether the whole subsystem is up or down. The subsystem's up- or downtime is updated depending on the outcome, as is the number of redundant units.

SCHED

Subroutine SCHED will determine if the time has arrived for a scheduled maintenance outage. If the attained time is between the "year end - scheduled downtime" and the year's end, then the system is in the scheduled downtime period. A flag will be set to prevent calling FAIL at this time (this can be changed to allow for residual failures while shutdown).

GATE

This subroutine will determine the status of each gate, once the status of all the subsystems is known. It will access the array produced by the subroutine ORDGAT for the gate calculation ordering. Then it will check inputs of gates in the order that they appear in that array and calculate the outputs (i.e., status) of such gates according to the truth tables for each kind of gate. This will ensure that no backtracking is needed as ORDGAT calculations guarantee that inputs for a particular gate calculation will be ready from the preceding gate calculations. The gate clocks will be updated in accordance with the newly calculated status of the gates. The final gate, whose output denotes success or failure of the system, will then indicate if the system was available in this particular time step.

RANDNO

This subroutine produces a pseudorandom number via the method of recursive congruential generation. The parameters of this generation are such that long periodicity of the random number sequence is ensured. In this method, each random number is used to generate the next random number. The initial parameters can be either chosen by the user or set internally (with the possibility of either starting from a "random" set of parameters or reproducing the same sequence any time the program is run).

The initial parameters are $\log_2(m)$, a and x_o, such that:

$$x_n = \text{mod}_m(x_o * a)$$

$$N = \text{decimal fraction part of } (x_n/m*a)$$

for next number: $x_o = x_n$, where N is the nth random number, and x_n becomes the x_o for the next random number generation. The internal parameters in the program are:

$$\log_2 m = 20, \text{ i.e. } m = 2^{20}$$

$$a = 2^{10+1}$$

$$x_o = 566387 \; .$$

For a "random" starting point, the program saves the xn value from a previous run in an input file and uses it as the x_o for the current run.

RELY

This routine calculates the reliability of a particular unit at the end of the current time step, i.e. the probability of non-failure during this small period, $(e^{-\lambda\Delta t})$. Currently, this reliability will not change over time, so this routine need be called only once to store reliabilities of constant failure rates. In the future we may want to have different reliabilities after certain types of failure (e.g., degraded performance), or during subsystem shutdown (i.e., residual failures). In the future we may incorporate some kind of bathtub curve approximation in our time-dependent failure rates to account for defective components and wearout. This will probably be implemented in the analysis of individual subsystems rather than the whole system.

REPTIM

This routine returns the appropriate repair time to REPAIR. Currently, just the MTTR from input is returned; however, in the future, one may want to incorporate different repair times for different failure modes and even according to the availability of maintenance equipment and personnel, as well as maintenance strategies (for instance if a decision is made to have an early scheduled maintenance shutdown following some types of failures and under certain conditions).

Input to Program AVSYS

The input will be described below; additional data may be required in the future, as the code is made more sophisticated. The proper format for the particular data will be given in parentheses. The data is described in the order it is to be entered (the time unit is the hour):

1) Number of subsystems, time step length, number of time steps, number of histories (i3, f5.1, i3, i6).
2) Subsystem ID number, its mean time to failure, mean time to repair (i3, e10.4, f6.1).
3) Subsystem ID number, number of units designed to operate (i.e., minimum number + the number of online redundant units), number of spares, immediate repair option (0 for deferred repair, 1 for immediate repair), number of online redundant units (i3, i3, i3, i2, i3).

4) Number of AND gates, number of OR gates (2i3).

5) For each AND gate: AND gate ID number, number of subsystems, number of AND gates, number of OR gates at the inputs of the AND gate (4i3).

6) For each AND gate: AND gate ID number, identification number of each subsystem connected to the AND gate (21i3).

7) For each AND gate: AND gate ID number, identification number of each AND gate connected to the AND gate (21i3).

8) For each AND gate: AND gate ID number, identification number of each OR gate connected to the AND gate (21i3).

9) For each OR gate: OR gate ID number, number of subsystems, number of AND gates, number of OR gates at the inputs of the OR gate (4i3).

10) For each OR gate: OR gate ID number, identification number of each subsystem connected to the OR gate (21i3).

11) For each OR gate: OR gate ID number, identification number of each AND gate connected to the OR gate (21i3).

12) For each OR gate: OR gate ID number, identification number of each OR gate connected to the OR gate (21i3).

13) Number of subsystems with per demand failure rates (rather than the per hour failure rates) (i3).

14) Identification number of each such subsystem (i3).

15) Duration of scheduled maintenance shutdown period in hours (f6.1).

16) Random number generator input options; 1 for set starting point with internally fixed parameters, 2 for starting with the last random number generated and 3 for user supplied parameters (i2).

17) If the option in (17) has been set equal to 3, the user supplies the 3 input parameters here: $\log_2(m)$, a and x_0 (i3, f10.1, f9.1).

Output of Program AVSYS

The code will print the input data first: number of subsystems, time step size, number of time steps per history and number of histories; subsystem number, its mean time to failure and its mean time to repair; subsystem number, design number of operating units, number of spares, immediate repair option (1 for immediate repair, 0 for deferred repair), and number of active redundant units; number and type of logic gates; subsystem and gates connected to each logic gate. In addition the program-determined order of calculation of each logic gate is included. Finally, the subsystems with per demand failure rates are listed, as is the scheduled maintenance downtime per year.

The output of the code (which follows) consists of: cumulative number of failures, uptime, downtime and average availability of each subsystem; top event gate number and type; time dependent behavior (averaged over the histories) of the top event gate, including: time step number, number of failures in it, downtime in it, availability without and with scheduled maintenance for that time step; average availabilities without and with scheduled plant outage for each logic gate.

Sample Input/Output of the Code

A MARS configuration is presented below in Table 1, followed by the sample input and output for this case. Figure 1 is the success tree of the system.

FIGURE 1

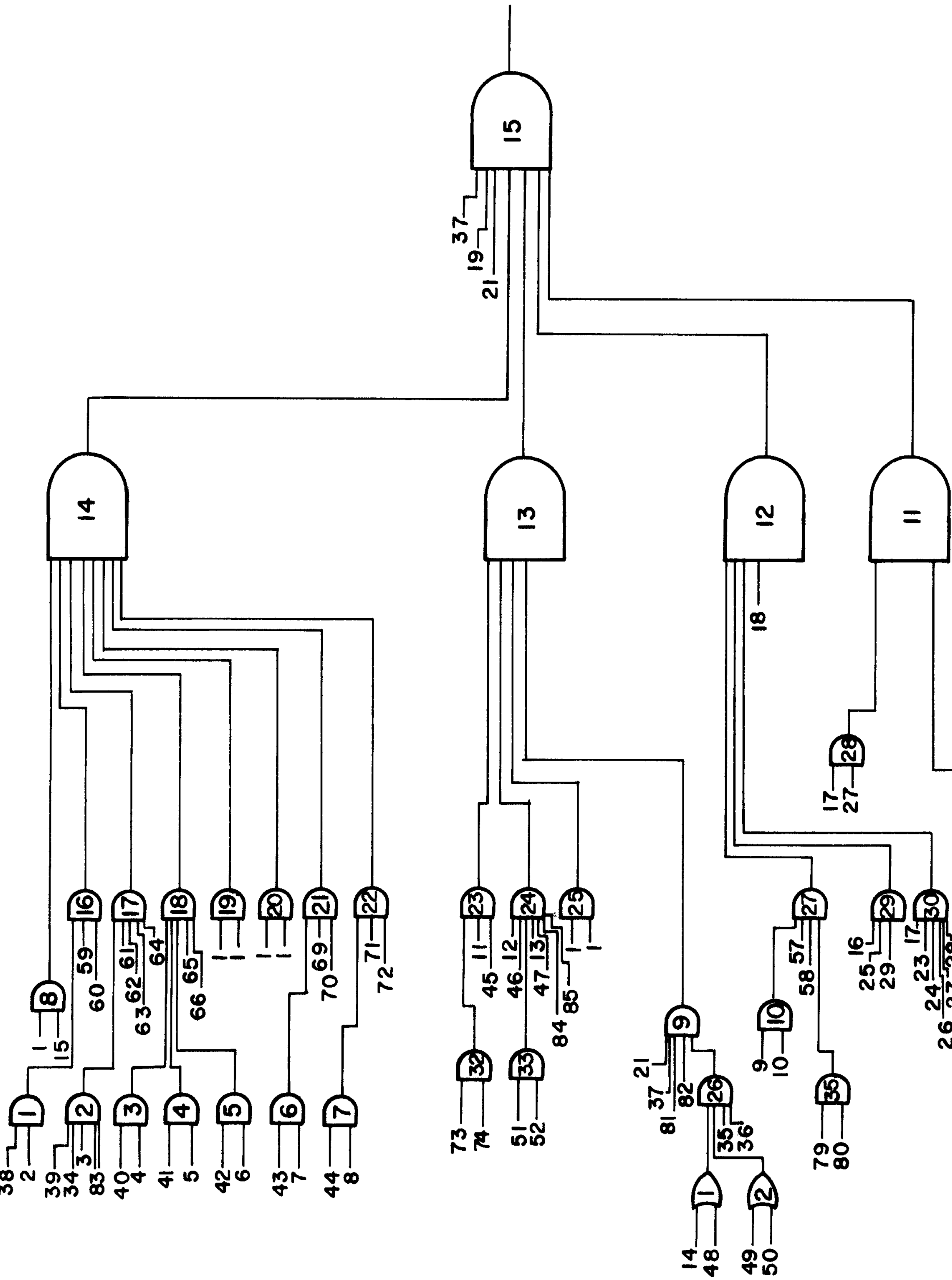


Table 1. Sample MARS Configuration

Subsystem	flrt(/hr)	mttr(hr)	Number Units Op.	Redundancy Active	Redundancy Offline
CC magnet coil	4.5E-6	720.	21/side	0	0
Choke magnet, superconducting	4.5E-6	720.	1/side	0	0
Choke magnet, normal	1.1E-5	240.	1/side	0	0
Transition magnet	4.5E-6	720.	1/side	0	0
Anchor magnet	4.5E-6	720.	2/side	0	0
Plug magnet	4.5E-6	720.	2/side	0	0
Recircularizing magnet	4.5E-6	720.	1/side	0	0
Recir. mag. C coils	4.5E-6	720.	1/side	0	0
Drift pump magnet	4.5E-6	720.	2/side	0	0
Direct convertor	5.7E-6	72.	1/side	0	0
ICRH	2.5E-3	4.	1/side	0	0
ECRH, low power	2.5E-3	4.	3/side	1	0
ECRH, high power, 2 launchers/side	2.5E-3	4.	9/launcher 18/side	1	0
Neutral beams	2.5E-3	4.	1/side	0	0
Vacuum pumps	4.6E-5	96.	5	1	0
Shield	2.4E-5	168.	1	0	0
Blanket, LiPb leak	3.8E-5	288.	1	0	0
Blanket, He leak	1.1E-5	120.	1	0	0
Reflector	7.6E-6	960.	1	0	0
Bellows seal	2.3E-5	240.	1	0	0
Service station	1.1E-5	168.	1	0	0
LiPb pump	1.6E-5	336.	7	2	0
Reflector water pump	1.6E-5	336.	2	1	0
End tank H_2O pump	1.6E-5	336.	2	1	0
Balance of plant	2.5E-4	240.	1	0	0
Control and instrumentation	2.0E-4	48.	1	0	1
Fueling - rail gun	2.3E-5	16.	1	0	1
Fuel: T_2 extraction	5.7E-6	24.	1	0	0
Fuel: preparation	1.1E-5	2.	1	0	1

Fuel: pellet fabrication	2.3E-5	8.	1	0	1
Cryogenic system, compressors	3.8E-5	52.	6(4K)+6(1.8K)	1	0
Cryogenic system, turboexpanders	3.8E-5	52.	3+3	1	0
Power supply, SCR	1.1E-5	48.	1/equipment serving	0	0
Power supply, transformer	4.6E-6	48.	same	0	0

Sample Input

Comments (not part of input).

Input	Comments
85 40. 100 100	Number of subsystems, time step (hr), number of time steps, number of histories.
1 1.0E+08 1.0	Subsystem number, its MTTF and MTTR (hr)
2 2.2E+05 720.0	
3 9.1E+04 240.0	
4 2.2E+05 720.0	
5 2.2E+05 720.0	
6 2.2E+05 720.0	
7 2.2E+05 720.0	
8 2.2E+05 720.0	
9 9.1E+04 240.0	
10 1.8E+05 72.0	
11 4.0E+02 4.0	
12 4.0E+02 4.0	
13 4.0E+02 4.0	
14 4.0E+02 4.0	
15 2.2E+04 96.0	
16 4.2E+04 168.0	
17 2.6E+04 288.0	
18 4.0E+03 240.0	
19 5.0E+03 48.0	
20 4.3E+04 16.0	
21 2.6E+04 52.0	
22 5.0E-02 10.0	
23 1.3E+05 960.0	
24 4.3E+04 240.0	
25 9.1E+04 168.0	
26 9.1E+04 120.0	
27 6.2E+04 330.0	
28 6.2E+04 330.0	
29 6.2E+04 330.0	
30 1.8E+05 24.0	
31 9.1E+04 2.0	
32 1.0E+08 1.0	
33 4.3E+04 8.0	

34	2.2E+05	720.0
35	9.1E+04	48.0
36	2.2E+05	48.0
37	2.6E+04	52.0
38	2.2E+05	720.0
39	9.1E+04	240.0
40	2.2E+05	720.0
41	2.2E+05	720.0
42	2.2E+05	720.0
43	2.2E+05	720.0
44	2.2E+05	720.0
45	4.0E+02	4.0
46	4.0E+02	4.0
47	4.0E+02	4.0
48	4.0E+02	4.0
49	4.0E+02	4.0
50	4.0E+02	4.0
51	9.1E+04	48.0
52	2.2E+05	48.0
53	1.0E+08	1.0
54	1.0E+08	1.0
55	1.0E+08	1.0
56	1.0E+08	1.0
57	9.1E+04	48.0
58	2.2E+05	48.0
59	9.1E+04	48.0
60	2.2E+05	48.0
61	9.1E+04	48.0
62	2.2E+05	48.0
63	9.1E+04	48.0
64	2.2E+05	48.0
65	9.1E+04	48.0
66	2.2E+05	48.0
67	1.0E+08	1.0
68	1.0E+08	1.0
69	9.1E+04	48.0
70	2.2E+05	48.0
71	9.1E+04	48.0
72	2.2E+05	48.0
73	9.1E+04	48.0
74	2.2E+05	48.0
75	1.0E+08	1.0
76	1.0E+08	1.0
77	1.0E+08	1.0
78	1.0E+08	1.0
79	9.1E+04	240.0
80	1.8E+05	72.0
81	2.6E+04	52.0
82	2.6E+04	52.0
83	2.2E+05	720.0
84	4.0E+02	4.0
85	4.0E+02	4.0

1	1	0	1	0
2	21	0	1	0
3	1	0	1	0
4	1	0	1	0
5	1	0	1	0
6	2	0	1	0
7	1	0	1	0
8	1	0	1	0
9	2	0	1	0
10	4	0	1	0
11	1	0	1	0
12	3	0	1	1
13	9	0	1	1
14	1	0	1	0
15	5	0	1	1
16	1	0	1	0
17	1	0	1	0
18	1	0	1	0
19	1	1	1	0
20	1	1	1	0
21	6	0	1	1
22	1	0	1	0
23	1	0	1	0
24	1	0	1	0
25	1	0	1	0
26	1	0	1	0
27	7	0	0	2
28	2	0	1	1
29	2	0	1	1
30	1	0	1	0
31	1	1	1	0
32	1	0	1	0
33	1	1	1	0
34	1	0	1	0
35	1	0	1	0
36	1	0	1	0
37	3	0	1	1
38	21	0	1	0
39	1	0	1	0
40	1	0	1	0
41	1	0	1	0
42	2	0	1	0
43	1	0	1	0
44	1	0	1	0
45	1	0	1	0
46	3	0	1	1
47	9	0	1	1
48	1	0	1	0
49	1	0	1	0
50	1	0	1	0
51	1	0	1	0
52	1	0	1	0

Subsystem number, units operating, spares, immediate repair option, active redundancy.

53	1	0	1	0
54	1	0	1	0
55	1	0	1	0
56	1	0	1	0
57	1	0	1	0
58	1	0	1	0
59	1	0	1	0
60	1	0	1	0
61	1	0	1	0
62	1	0	1	0
63	1	0	1	0
64	1	0	1	0
65	1	0	1	0
66	1	0	1	0
67	1	0	1	0
68	1	0	1	0
69	1	0	1	0
70	1	0	1	0
71	1	0	1	0
72	1	0	1	0
73	1	0	1	0
74	1	0	1	0
75	1	0	1	0
76	1	0	1	0
77	1	0	1	0
78	1	0	1	0
79	2	0	1	0
80	4	0	1	0
81	6	0	1	1
82	3	0	1	1
83	1	0	1	0
84	9	0	1	1
85	9	0	1	1

Number of AND gates, number of OR gates, AND gate ID, number of subsystems, AND gates, OR gates at its inputs

39	2		
1	2	0	0
2	4	0	0
3	2	0	0
4	2	0	0
5	2	0	0
6	2	0	0
7	2	0	0
8	2	0	0
9	4	1	0
10	2	0	0
11	0	2	0
12	1	3	0
13	0	4	0
14	0	8	0
15	3	4	0
16	2	1	0

17	4	1	0
18	2	3	0
19	2	0	0
20	2	0	0
21	2	1	0
22	2	1	0
23	2	1	0
24	6	1	0
25	2	0	0
26	2	0	2
27	2	2	0
28	2	0	0
29	3	0	0
30	6	0	0
31	4	0	0
32	2	0	0
33	2	0	0
34	2	0	0
35	2	0	0
36	2	0	0
37	2	0	0
38	2	0	0
39	2	0	0

AND gate ID, ID numbers of subsystems, AND gates and OR gates at its inputs

1	38	2						
2	39	3	34	83				
3	40	4						
4	41	5						
5	42	6						
6	43	7						
7	44	8						
8	1	15						
9	21	37	81	82				
9	26							
10	9	10						
11	28	31						
12	18							
12	29	27	30					
13	23	24	25	9				
14	16	17	18	19	20	21	22	8
15	21	19	37					
15	14	13	12	11				
16	59	60						
16	1							
17	61	62	63	64				
17	2							
18	65	66						
18	3	4	5					
19	1	1						
20	1	1						
21	69	70						

21	6					
22	71	72				
22	7					
23	11	45				
23	32					
24	12	46	13	47	84	85
24	33					
25	1	1				
26	35	36				
26	1	2				
27	57	58				
27	10	39				
28	17	27				
29	16	25	29			
30	17	23	24	26	27	28
31	20	30	31	33		
32	73	74				
33	51	52				
34	1	1				
35	1	1				
36	1	1				
37	1	1				
38	1	1				
39	79	80				

1	2	0	0	OR gate ID, number of subsystems, AND gates,
2	2	0	0	OR gates at its inputs
1	14	48		OR gate ID, ID numbers of subsystems, AND and OR
2	49	50		gates at its inputs
0				Number of subsystems with per demand failure rates
672.				Scheduled outage duration in hr/year
1				Random number generator option (set parameters)

<u>Sample Output</u>

Input Data

Subsystem Number	Mean Time to Failure, hr or Mean Number of Demands to Failure	Mean Time to Repair, hr
1	1.0000E+08	1.0000E+00
2	2.2222E+05	7.2000E+02
3	9.0909E+04	2.4000E+02
4	2.2222E+05	7.2000E+02
5	2.2222E+05	7.2000E+02
6	2.2222E+05	7.2000E+02
7	2.2222E+05	7.2000E+02
8	2.2222E+05	7.2000E+02

9	9.0909E+04	2.4000E+02
10	1.7544E+05	7.2000E+01
11	4.0000E+02	4.0000E+00
12	4.0000E+02	4.0000E+00
13	4.0000E+02	4.0000E+00
14	4.0000E+02	4.0000E+00
15	2.1739E+04	9.6000E+01
16	4.1667E+04	1.6800E+02
17	2.6316E+04	2.8800E+02
18	4.0000E+03	2.4000E+02
19	5.0000E+03	4.8000E+01
20	4.3478E+04	1.6000E+01
21	2.6316E+04	5.2000E+01
22	5.0000E-02	1.0000E+01
23	1.3158E+05	9.6000E+02
24	4.3478E+04	2.4000E+02
25	9.0909E+04	1.6800E+02
26	9.0909E+04	1.2000E+02
27	6.2500E+04	3.3600E+02
28	6.2500E+04	3.3600E+02
29	6.2500E+04	3.3600E+02
30	1.7544E+05	2.4000E+01
31	9.0909E+04	2.0000E+00
32	1.0000E+08	1.0000E+00
33	4.3478E+04	8.0000E+00
34	2.2222E+05	7.2000E+02
35	9.0909E+04	4.8000E+01
36	2.1739E+05	4.8000E+01
37	2.6316E+04	5.2000E+01
38	2.2222E+05	7.2000E+02
39	9.0909E+04	2.4000E+02
40	2.2222E+05	7.2000E+02
41	2.2222E+05	7.2000E+02
42	2.2222E+05	7.2000E+02
43	2.2222E+05	7.2000E+02
44	2.2222E+05	7.2000E+02
45	4.0000E+02	4.0000E+00
46	4.0000E+02	4.0000E+00
47	4.0000E+02	4.0000E+00
48	4.0000E+02	4.0000E+00
49	4.0000E+02	4.0000E+00
50	4.0000E+02	4.0000E+00
51	9.0909E+04	4.8000E+01
52	2.1739E+05	4.8000E+01
53	1.0000E+08	1.0000E+00
54	1.0000E+08	1.0000E+00
55	1.0000E+08	1.0000E+00
56	1.0000E+08	1.0000E+00
57	9.0909E+04	4.8000E+01
58	2.1739E+05	4.8000E+01
59	9.0909E+04	4.8000E+01
60	2.1739E+05	4.8000E+01

61	9.0909E+04	4.8000E+01
62	2.1739E+05	4.8000E+01
63	9.0909E+04	4.8000E+01
64	2.1739E+05	4.8000E+01
65	9.0909E+04	4.8000E+01
66	2.1739E+05	4.8000E+01
67	1.0000E+08	1.0000E+00
68	1.0000E+08	1.0000E+00
69	9.0909E+04	4.8000E+01
70	2.1739E+05	4.8000E+01
71	9.0909E+04	4.8000E+01
72	2.1739E+05	4.8000E+01
73	9.0909E+04	4.8000E+01
74	2.1739E+05	4.8000E+01
75	1.0000E+08	1.0000E+00
76	1.0000E+08	1.0000E+00
77	1.0000E+08	1.0000E+00
78	1.0000E+08	1.0000E+00
79	9.0909E+04	2.4000E+02
80	1.7544E+05	7.2000E+01
81	2.6316E+04	5.2000E+01
82	2.6316E+04	5.2000E+01
83	2.2222E+05	7.2000E+02
84	4.0000E+02	4.0000E+00
85	4.0000E+02	4.0000E+00

Systs	Delt	Total No. of Time Steps	Number of Histories
85	4.0000E+01	100	100

Subsystem Number	Design Number of Units Operating	Number of Spares	Immediate Repair Option	Number of Units Actively Redundant
1	1	0	1	0
2	21	0	1	0
3	1	0	1	0
4	1	0	1	0
5	1	0	1	0
6	2	0	1	0
7	1	0	1	0
8	1	0	1	0
9	2	0	1	0
10	4	0	1	0
11	1	0	1	0
12	3	0	1	1
13	9	0	1	1
14	1	0	1	0
15	5	0	1	1
16	1	0	1	0
17	1	0	1	0

18	1	0	1	0
19	1	1	1	0
20	1	1	1	0
21	6	0	1	1
22	1	0	1	0
23	1	0	1	0
24	1	0	1	0
25	1	0	1	0
26	1	0	1	0
27	7	0	0	2
28	2	0	1	1
29	2	0	1	1
30	1	0	1	0
31	1	1	1	0
32	1	0	1	0
33	1	1	1	0
34	1	0	1	0
35	1	0	1	0
36	1	0	1	0
37	3	0	1	1
38	21	0	1	0
39	1	0	1	0
40	1	0	1	0
41	1	0	1	0
42	2	0	1	0
43	1	0	1	0
44	1	0	1	0
45	1	0	1	0
46	3	0	1	1
47	9	0	1	1
48	1	0	1	0
49	1	0	1	0
50	1	0	1	0
51	1	0	1	0
52	1	0	1	0
53	1	0	1	0
54	1	0	1	0
55	1	0	1	0
56	1	0	1	0
57	1	0	1	0
58	1	0	1	0
59	1	0	1	0
60	1	0	1	0
61	1	0	1	0
62	1	0	1	0
63	1	0	1	0
64	1	0	1	0
65	1	0	1	0
66	1	0	1	0
67	1	0	1	0
68	1	0	1	0
69	1	0	1	0

70	1	0	1	0
71	1	0	1	0
72	1	0	1	0
73	1	0	1	0
74	1	0	1	0
75	1	0	1	0
76	1	0	1	0
77	1	0	1	0
78	1	0	1	0
79	2	0	1	0
80	4	0	1	0
81	6	0	1	1
82	3	0	1	1
83	1	0	1	0
84	9	0	1	1
85	9	0	1	1

number of and gates = 39   number of or gates = 2

logic gate interconnections
AND gate number

1	subsystems	38	2						
2	subsystems	39	3	34	83				
3	subsystems	40	4						
4	subsystems	41	5						
5	subsystems	42	6						
6	subsystems	43	7						
7	subsystems	44	8						
8	subsystems	1	15						
9	subsystems	21	37	81	82				
9	and gates	26							
10	subsystems	9	10						
11	and gates	28	31						
12	subsystems	18							
12	and gates	29	27	30					
13	and gates	23	24	25	9				
14	and gates	16	17	18	19	20	21	22	8
15	subsystems	21	19	37					
15	and gates	14	13	12	11				

16	subsystems	59	60				
16	and gates	1					
17	subsystems	61	62	63	64		
17	and gates	2					
18	subsystems	65	66				
18	and gates	3	4	5			
19	subsystems	1	1				
20	subsystems	1	1				
21	subsystems	69	70				
21	and gates	6					
22	subsystems	71	72				
22	and gates	7					
23	subsystems	11	45				
23	and gates	32					
24	subsystems	12	46	13	47	84	85
24	and gates	33					
25	subsystems	1	1				
26	subsystems	35	36				
26	or gates	1	2				
27	subsystems	57	58				
27	and gates	10	39				
28	subsystems	17	27				
29	subsystems	16	25	29			
30	subsystems	17	23	24	26	27	28
31	subsystems	20	30	31	33		
32	subsystems	73	74				
33	subsystems	51	52				
34	subsystems	1	1				
35	subsystems	1	1				
36	subsystems	1	1				
37	subsystems	1	1				
38	subsystems	1	1				

39 subsystems 79 80

OR gate number

1 subsystems 14 48

2 subsystems 49 50

subsystems with per demand failures
none

duration (in hours per year) of scheduled outage 6.7200E+02

Order of Gate Calculation	Gate ID Number	Gate Type (2 = and, 3 = or)
1	1	2
2	2	2
3	3	2
4	4	2
5	5	2
6	6	2
7	7	2
8	8	2
9	10	2
10	19	2
11	20	2
12	25	2
13	28	2
14	29	2
15	30	2
16	31	2
17	32	2
18	33	2
19	34	2
20	35	2
21	36	2
22	37	2
23	38	2
24	39	2
25	1	3
26	2	3
27	11	2
28	16	2
29	17	2
30	18	2
31	21	2
32	22	2
33	23	2
34	24	2
35	26	2
36	27	2

37	9	2
38	12	2
39	13	2
40	14	2
41	15	2

*******OUTPUT*******

System Number	Total Failures	Up Time	Down Time	Availability
1	0.	4.0000E+05	0.	1.0000E+00
2	3.2000E+01	3.7872E+05	2.1280E+04	9.4680E-01
3	6.0000E+00	3.9864E+05	1.3600E+03	9.9660E-01
4	3.0000E+00	3.9844E+05	1.5600E+03	9.9610E-01
5	0.	4.0000E+05	0.	1.0000E+00
6	2.0000E+00	3.9920E+05	8.0000E+02	9.9800E-01
7	0.	4.0000E+05	0.	1.0000E+00
8	0.	4.0000E+05	0.	1.0000E+00
9	9.0000E+00	3.9784E+05	2.1600E+03	9.9460E-01
10	1.3000E+01	3.9900E+05	1.0000E+03	9.9750E-01
11	1.0020E+03	3.5992E+05	4.0080E+04	8.9980E-01
12	2.9080E+03	3.8880E+05	1.1200E+04	9.7200E-01
13	8.6630E+03	2.9376E+05	1.0624E+05	7.3440E-01
14	1.0210E+03	3.5916E+05	4.0840E+04	8.9790E-01
15	9.2000E+01	3.9988E+05	1.2000E+02	9.9970E-01
16	1.3000E+01	3.9740E+05	2.6000E+03	9.9350E-01
17	1.5000E+01	3.9544E+05	4.5600E+03	9.8860E-01
18	9.9000E+01	3.7724E+05	2.2760E+04	9.4310E-01
19	6.9000E+01	4.0000E+05	0.	1.0000E+00
20	1.2000E+01	4.0000E+05	0.	1.0000E+00
21	1.0500E+02	4.0000E+05	0.	1.0000E+00
22	1.0000E+04	0.	4.0000E+05	0.
23	1.0000E+00	3.9980E+05	2.0000E+02	9.9950E-01
24	1.5000E+01	3.9640E+05	3.6000E+03	9.9100E-01
25	1.0000E+00	3.9980E+05	2.0000E+02	9.9950E-01
26	7.0000E+00	3.9916E+05	8.4000E+02	9.9790E-01
27	4.7000E+01	4.0000E+05	0.	1.0000E+00
28	1.1000E+01	4.0000E+05	0.	1.0000E+00
29	1.4000E+01	4.0000E+05	0.	1.0000E+00
30	1.0000E+00	3.9996E+05	4.0000E+01	9.9990E-01
31	1.1000E+01	4.0000E+05	0.	1.0000E+00
32	0.	4.0000E+05	0.	1.0000E+00
33	8.0000E+00	4.0000E+05	0.	1.0000E+00
34	1.0000E+00	3.9960E+05	4.0000E+02	9.9900E-01
35	3.0000E+00	3.9976E+05	2.4000E+02	9.9940E-01
36	1.0000E+00	3.9992E+05	8.0000E+01	9.9980E-01
37	5.7000E+01	4.0000E+05	0.	1.0000E+00
38	3.4000E+01	3.7640E+05	2.3600E+04	9.4100E-01
39	3.0000E+00	3.9948E+05	5.2000E+02	9.9870E-01
40	2.0000E+00	3.9856E+05	1.4400E+03	9.9640E-01

41	5.0000E+00	3.9640E+05	3.6000E+03	9.9100E-01
42	2.0000E+00	3.9856E+05	1.4400E+03	9.9640E-01
43	2.0000E+00	3.9856E+05	1.4400E+03	9.9640E-01
44	1.0000E+00	3.9928E+05	7.2000E+02	9.9820E-01
45	9.4700E+02	3.6212E+05	3.7880E+04	9.0530E-01
46	2.8180E+03	3.9016E+05	9.8400E+03	9.7540E-01
47	8.6300E+03	2.9300E+05	1.0700E+05	7.3250E-01
48	1.0470E+03	3.5812E+05	4.1880E+04	8.9530E-01
49	9.5300E+02	3.6188E+05	3.8120E+04	9.0470E-01
50	1.0170E+03	3.5932E+05	4.0680E+04	8.9830E-01
51	5.0000E+00	3.9960E+05	4.0000E+02	9.9900E-01
52	4.0000E+00	3.9968E+05	3.2000E+02	9.9920E-01
53	0.	4.0000E+05	0.	1.0000E+00
54	0.	4.0000E+05	0.	1.0000E+00
55	0.	4.0000E+05	0.	1.0000E+00
56	0.	4.0000E+05	0.	1.0000E+00
57	4.0000E+00	3.9968E+05	3.2000E+02	9.9920E-01
58	2.0000E+00	3.9984E+05	1.6000E+02	9.9960E-01
59	9.0000E+00	3.9928E+05	7.2000E+02	9.9820E-01
60	4.0000E+00	3.9968E+05	3.2000E+02	9.9920E-01
61	7.0000E+00	3.9944E+05	5.6000E+02	9.9860E-01
62	3.0000E+00	3.9976E+05	2.4000E+02	9.9940E-01
63	4.0000E+00	3.9968E+05	3.2000E+02	9.9920E-01
64	0.	4.0000E+05	0.	1.0000E+00
65	5.0000E+00	3.9960E+05	4.0000E+02	9.9900E-01
66	5.0000E+00	3.9960E+05	4.0000E+02	9.9900E-01
67	0.	4.0000E+05	0.	1.0000E+00
68	0.	4.0000E+05	0.	1.0000E+00
69	3.0000E+00	3.9976E+05	2.4000E+02	9.9940E-01
70	2.0000E+00	3.9984E+05	1.6000E+02	9.9960E-01
71	4.0000E+00	3.9968E+05	3.2000E+02	9.9920E-01
72	3.0000E+00	3.9976E+05	2.4000E+02	9.9940E-01
73	6.0000E+00	3.9952E+05	4.8000E+02	9.9880E-01
74	0.	4.0000E+05	0.	1.0000E+00
75	0.	4.0000E+05	0.	1.0000E+00
76	0.	4.0000E+05	0.	1.0000E+00
77	0.	4.0000E+05	0.	1.0000E+00
78	0.	4.0000E+05	0.	1.0000E+00
79	8.0000E+00	3.9808E+05	1.9200E+03	9.9520E-01
80	3.0000E+00	3.9976E+05	2.4000E+02	9.9940E-01
81	8.1000E+01	3.9996E+05	4.0000E+01	9.9990E-01
82	3.6000E+01	4.0000E+05	0.	1.0000E+00
83	0.	4.0000E+05	0.	1.0000E+00
84	8.3700E+03	2.9796E+05	1.0204E+05	7.4490E-01
85	8.6170E+03	2.9312E+05	1.0688E+05	7.3280E-01

top event is and gate number 15

time dependent availability, averaged over 100 histories of top event gate

Time Step Number	Number of Failures	Downtime	Availability
1	5.0000E-02	2.4000E+00	9.4000E-01
2	2.0000E-02	3.2000E+00	9.2000E-01
3	4.0000E-02	4.8000E+00	8.8000E-01
4	0.	4.0000E+00	9.0000E-01
5	2.0000E-02	4.8000E+00	8.8000E-01
6	2.0000E-02	5.6000E+00	8.6000E-01
7	3.0000E-02	6.0000E+00	8.5000E-01
8	5.0000E-02	7.2000E+00	8.2000E-01
9	4.0000E-02	8.8000E+00	7.8000E-01
10	4.0000E-02	1.0000E+01	7.5000E-01
11	2.0000E-02	9.6000E+00	7.6000E-01
12	3.0000E-02	9.6000E+00	7.6000E-01
13	2.0000E-02	1.0000E+01	7.5000E-01
14	0.	9.2000E+00	7.7000E-01
15	6.0000E-02	1.0800E+01	7.3000E-01
16	0.	1.0800E+01	7.3000E-01
17	3.0000E-02	1.1200E+01	7.2000E-01
18	1.0000E-02	1.0400E+01	7.4000E-01
19	0.	1.0000E+01	7.5000E-01
20	3.0000E-02	1.1200E+01	7.2000E-01
21	4.0000E-02	1.0400E+01	7.4000E-01
22	0.	1.0000E+01	7.5000E-01
23	3.0000E-02	1.0000E+01	7.5000E-01
24	1.0000E-02	9.2000E+00	7.7000E-01
25	1.0000E-02	9.2000E+00	7.7000E-01
26	2.0000E-02	8.0000E+00	8.0000E-01
27	3.0000E-02	8.4000E+00	7.9000E-01
28	4.0000E-02	8.8000E+00	7.8000E-01
29	1.0000E-02	9.2000E+00	7.7000E-01
30	1.0000E-02	8.8000E+00	7.8000E-01
31	3.0000E-02	1.0000E+01	7.5000E-01
32	1.0000E-02	9.6000E+00	7.6000E-01
33	3.0000E-02	1.0000E+01	7.5000E-01
34	4.0000E-02	1.0400E+01	7.4000E-01
35	5.0000E-02	1.1600E+01	7.1000E-01
36	1.0000E-02	1.1200E+01	7.2000E-01
37	2.0000E-02	1.0000E+01	7.5000E-01
38	2.0000E-02	1.0400E+01	7.4000E-01
39	2.0000E-02	1.0000E+01	7.5000E-01
40	4.0000E-02	1.0400E+01	7.4000E-01
41	5.0000E-02	1.1200E+01	7.2000E-01
42	3.0000E-02	1.1200E+01	7.2000E-01
43	3.0000E-02	1.0800E+01	7.3000E-01
44	1.0000E-02	1.0400E+01	7.4000E-01
45	4.0000E-02	9.6000E+00	7.6000E-01
46	3.0000E-02	8.4000E+00	7.9000E-01
47	0.	6.8000E+00	8.3000E-01
48	3.0000E-02	6.8000E+00	8.3000E-01
49	2.0000E-02	7.2000E+00	8.2000E-01

50	2.0000E-02	8.0000E+00	8.0000E-01
51	2.0000E-02	8.4000E+00	7.9000E-01
52	0.	7.2000E+00	8.2000E-01
53	2.0000E-02	7.2000E+00	8.2000E-01
54	0.	6.8000E+00	8.3000E-01
55	2.0000E-02	6.8000E+00	8.3000E-01
56	4.0000E-02	7.6000E+00	8.1000E-01
57	6.0000E-02	9.6000E+00	7.6000E-01
58	2.0000E-02	9.6000E+00	7.6000E-01
59	3.0000E-02	8.4000E+00	7.9000E-01
60	2.0000E-02	9.2000E+00	7.7000E-01
61	2.0000E-02	9.2000E+00	7.7000E-01
62	3.0000E-02	9.2000E+00	7.7000E-01
63	4.0000E-02	9.2000E+00	7.7000E-01
64	2.0000E-02	9.6000E+00	7.6000E-01
65	2.0000E-02	9.6000E+00	7.6000E-01
66	6.0000E-02	1.1600E+01	7.1000E-01
67	2.0000E-02	1.0800E+01	7.3000E-01
68	2.0000E-02	1.0000E+01	7.5000E-01
69	3.0000E-02	1.1200E+01	7.2000E-01
70	5.0000E-02	1.2400E+01	6.9000E-01
71	4.0000E-02	1.2800E+01	6.8000E-01
72	2.0000E-02	1.1600E+01	7.1000E-01
73	2.0000E-02	1.1200E+01	7.2000E-01
74	1.0000E-02	1.1600E+01	7.1000E-01
75	6.0000E-02	1.1600E+01	7.1000E-01
76	1.0000E-02	1.0000E+01	7.5000E-01
77	3.0000E-02	8.4000E+00	7.9000E-01
78	2.0000E-02	8.8000E+00	7.8000E-01
79	2.0000E-02	8.4000E+00	7.9000E-01
80	4.0000E-02	8.4000E+00	7.9000E-01
81	0.	4.0000E+01	0.
82	0.	4.0000E+01	0.
83	0.	4.0000E+01	0.
84	0.	4.0000E+01	0.
85	0.	4.0400E+01	0.
86	0.	4.0000E+01	0.
87	0.	4.0000E+01	0.
88	2.0000E-02	8.8000E+00	7.8000E-01
89	2.0000E-02	8.4000E+00	7.9000E-01
90	5.0000E-02	8.4000E+00	7.9000E-01
91	3.0000E-02	8.8000E+00	7.8000E-01
92	3.0000E-02	1.0000E+01	7.5000E-01
93	2.0000E-02	9.6000E+00	7.6000E-01
94	2.0000E-02	1.0000E+01	7.5000E-01
95	0.	8.4000E+00	7.9000E-01
96	4.0000E-02	8.4000E+00	7.9000E-01
97	3.0000E-02	9.2000E+00	7.7000E-01
98	3.0000E-02	8.8000E+00	7.8000E-01
99	3.0000E-02	9.6000E+00	7.6000E-01
100	2.0000E-02	9.6000E+00	7.6000E-01

Standard deviation of top gate time dependent availability

First time step sigma	Last time step sigma
2.3749e-02	4.2708e-02

And Gate Number	Availability
1	8.9170E-01
2	9.9440E-01
3	9.9250E-01
4	9.9100E-01
5	9.9440E-01
6	9.9640E-01
7	9.9820E-01
8	9.9970E-01
9	9.9910E-01
10	9.9210E-01
11	9.8860E-01
12	9.0180E-01
13	9.9610E-01
14	8.5580E-01
15	7.7150E-01
16	8.8990E-01
17	9.9160E-01
18	9.7590E-01
19	1.0000E+00
20	1.0000E+00
21	9.9540E-01
22	9.9680E-01
23	9.9880E-01
24	9.9820E-01
25	1.0000E+00
26	9.9920E-01
27	9.8550E-01
28	9.8860E-01
29	9.9300E-01
30	9.7700E-01
31	1.0000E+00
32	9.9880E-01
33	9.9820E-01
34	1.0000E+00
35	1.0000E+00
36	1.0000E+00
37	1.0000E+00
38	1.0000E+00
39	9.9460E-01

Or Gate Number	Availability
1	1.0000E+00
2	1.0000E+00

Description of Variables

a = random number generator constant

aclk(i,1) = average uptime of subsystem i, per history

alck(i,2) = average downtime of subsystem i, per history

alck(i,3) = average number of failures of subsystem i, per history

ano = random number generated

avail(i) = availability of subsystem i for current history

avav(i) = average availability of subsystem i

avgava(i) = average gate availability of AND gate i, with scheduled outage excluded

avgavo(i) = average gate availability of OR gate i, with scheduled outage excluded

clk(i,1) = uptime of subsystem i for current history

clk(i,2) = downtime of subsystem i for current history

clk(i,3) = number of failures of subsystem i for current history

delt = time step size, in hours

flrt(j) = failure rate (per hr or per demand) of subsystem j

hist = current history number

gavand(i) = availability of AND gate i, for current history, without scheduled outage

gavor(i) = availability of OR gate i for current history, without scheduled outage

gclk(i,1,1) = uptime of logic gate i

gclk(i,1,2) = downtime of logic gate i

gclk(i,2,1) = ID number of logic gate (i.e., AND gate ID number or OR gate ID number)

gclk(i,2,2) = type of logic gate i; 2 for AND gate, 3 for OR gate

iaand(i,l) = ID number of the AND gate connected to the lth AND gate input of the ith AND gate

ian(i,1) = number of subsystems connected to the inputs of AND gate i

ian(i,2) = number of AND gates connected to the inputs of AND gate i

ian(i,3) = number of OR gates connected to the inputs of AND gate i

ianacc(ijk) = ID number of AND gate that is the ijkth AND gate to be accounted for in subroutine ORDGAT

iaor(i,l) = ID number of the OR gate connected to the lth OR gate input of the ith AND gate

iasub(i,l) = ID number of the subsystem connected to the lth subsystem input of the ith AND gate

icand(i) = status of AND gate i, 1 for up, 0 for down

icor(i) = status of OR gate i, 1 for up, 0 for down

ifail(i) = status of subsystem i, 1 for up, 0 for down

ifailu(i,j) = 1 for incipient failure of unit j of subsystem i, 0 otherwise

igord(i,1) = ID number of a logic gate (AND or OR) that is to be the ith gate calculated

igord(i,2) = type of such gate; 2 for an AND gate, 3 for an OR gate

ijktot = total number of logic gates (AND and OR) ordered so far in subroutine ORDGAT

imrpr(j) = immediate repair option for subsystem j; 1 for immediate repair, 0 for deferred repair index = number of histories

ioand(i,l) = ID number of the AND gate connected to the lth AND gate input of the ith OR gate

ioor(i,l) = ID number of the OR gate connected to the lth OR gate input of the ith OR gate

iop(i,j) = 1 when unit j of subsystem i becomes operational, 0 otherwise

ior(i,1) = number of subsystems connected to the inputs of OR gate i

ior(i,2) = number of AND gates connected to the inputs of OR gate i

ior(i,3) = number of OR gates connected to the inputs of OR gate i

ioracc(ijk) = ID number of OR gate that is the ijkth OR gate to be accounted for

iosub(i,l) = ID number of the subsystem connected to the lth subsystem input of the ith OR gate

ipd(i) = demand failure option; 1 for subsystems with demand failure rate, 0 otherwise

irep(i,j) = 1 when unit j of subsystem i becomes repaired, 0 otherwise

iscflg = scheduled maintenance flag; if 1 the system is within the scheduled outage period, if 0 outside that period

it = ordinary number of current history

itot = total number of AND gates accounted for so far

jtot = total number of OR gates accounted for so far

kk = type of gate to be presently calculated, 2 for AND, 3 for OR

mm = ID number of the gate (AND or OR) to be presently calculated

mmnnyy = random number generator constant

mofn(j) = number of actively redundant units of subsystem j

mttr(j) = mean time to repair of subsystem j ndfr = number of subsystems with demand failure rates

ntot = total number of logic gates

numand = number of AND gates

numb(j) = number of identical subsystems of kind j designed to normally operate (includes the minimum number necessary for operation + the number of actively redundant units)

numor = number of OR gates

redun(j,1) = number of spares (i.e., passively redundant units) of subsystem j

redun(j,2) = total number of redundant units (active and passive) of subsystem j

rel(j) = reliability of subsystem j for time step delt

syssta(i,ich,1) = time of change number ich in the status of subsystem i

syssta(i,ich,2) = type of the above change: 0 for failure, 1 for repair, 2 for operation

systs = number of subsystems

timtrl = number of time steps

trials = number of histories

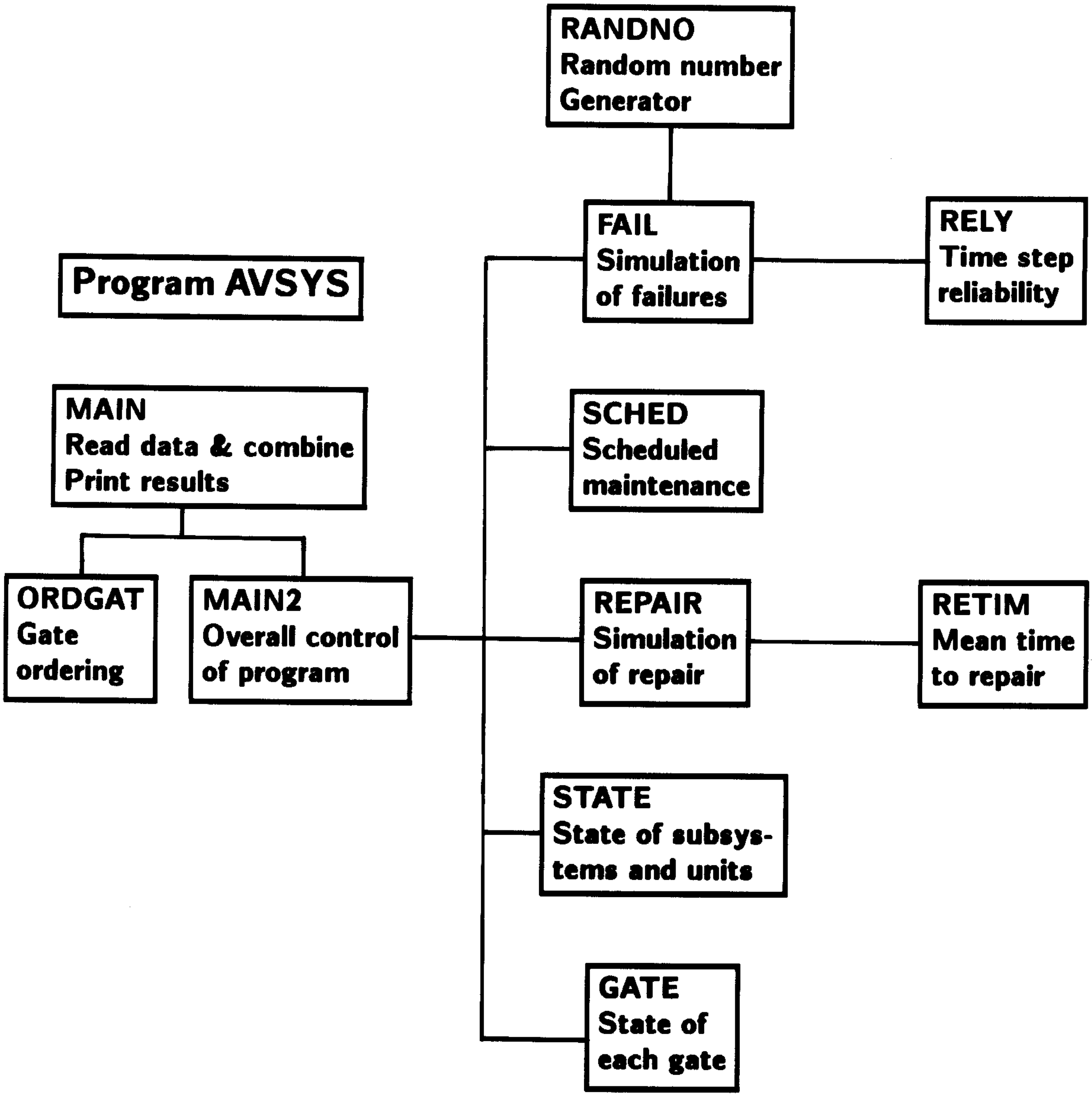
tsdy = number of hours per year scheduled for plant outage

xn = starting number xo for the next random number generation

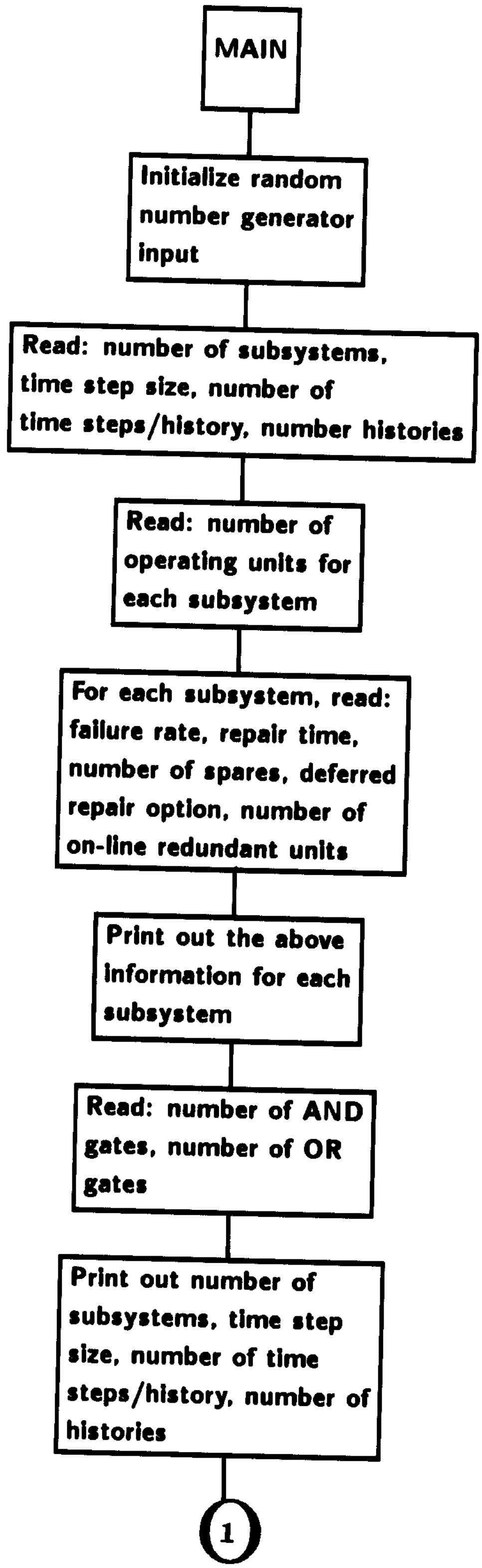
xo = random number generator constant

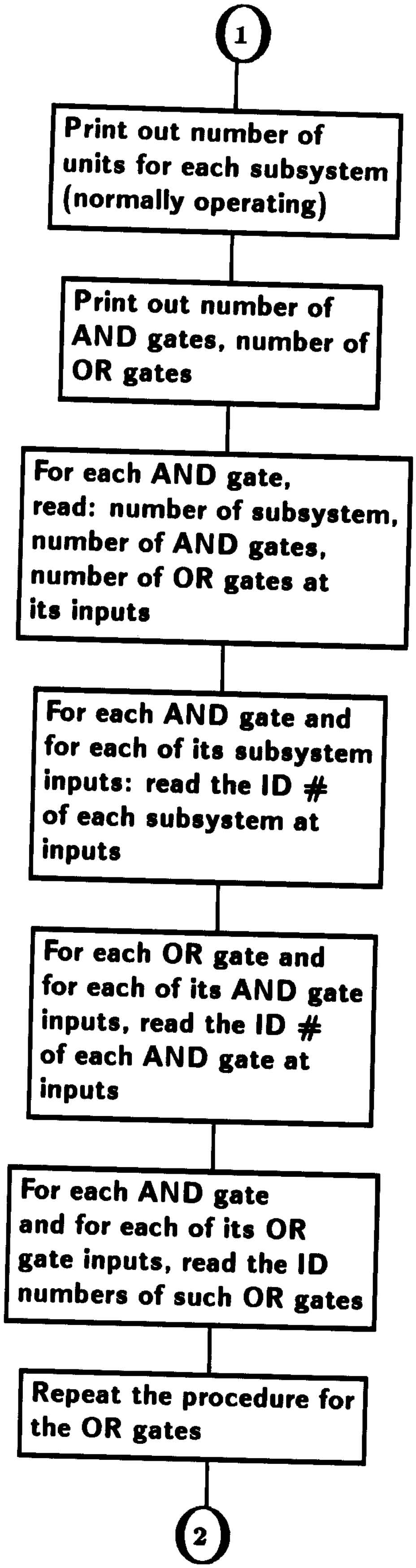
Acknowledgment

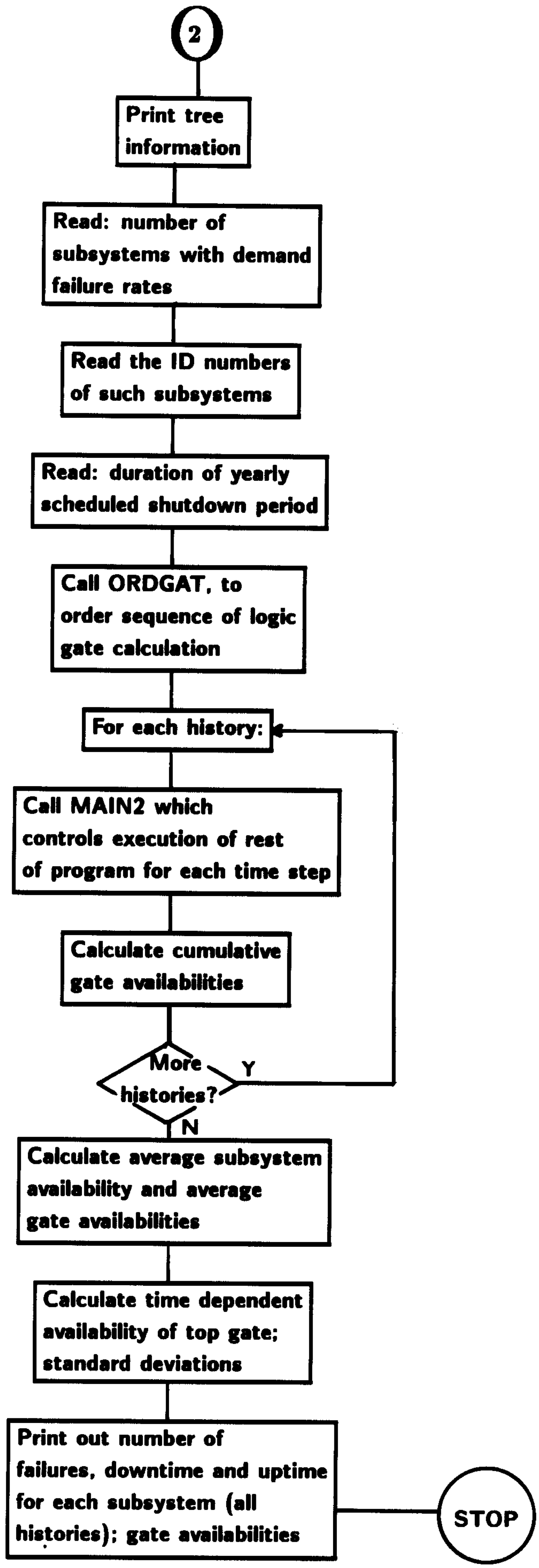
Support for this work has been provided by the U.S. Department of Energy and by the Wisconsin Electric Utilities Research Foundation (WEURF).

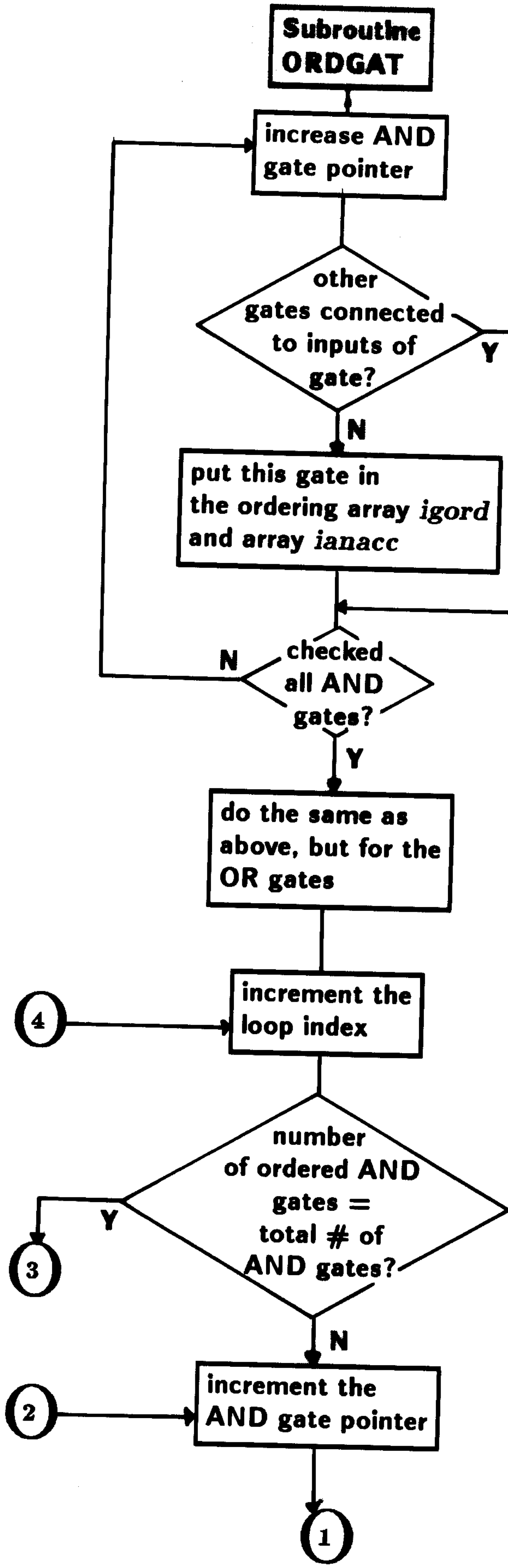


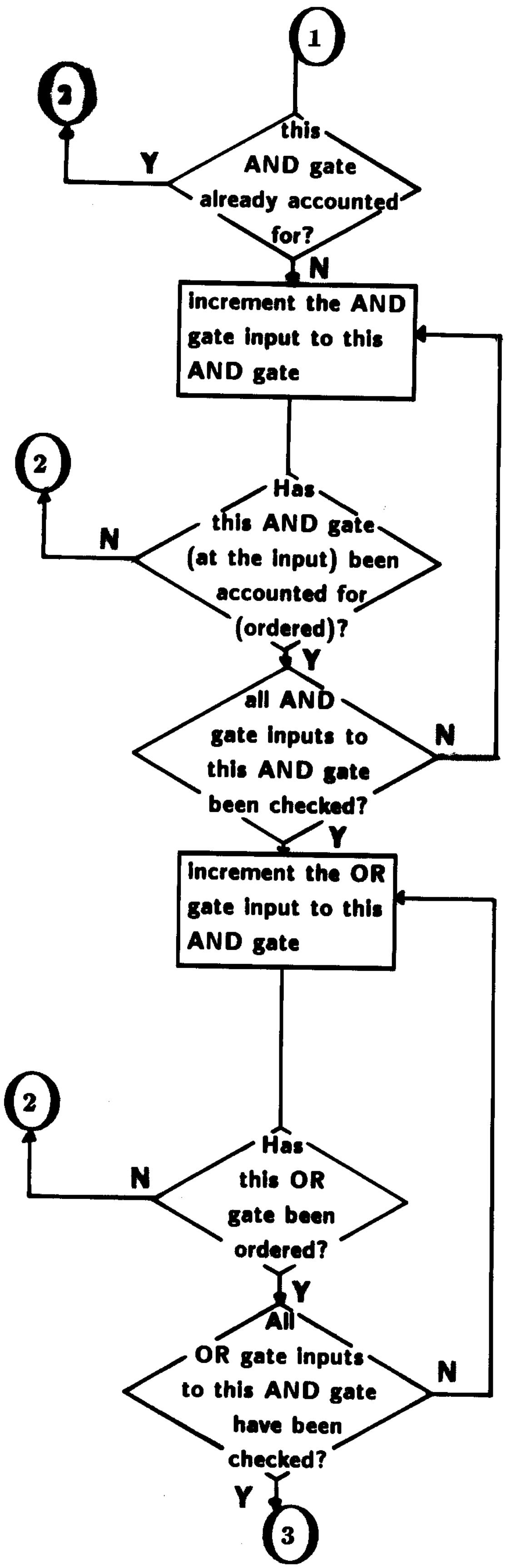
PROGRAM FLOWCHART

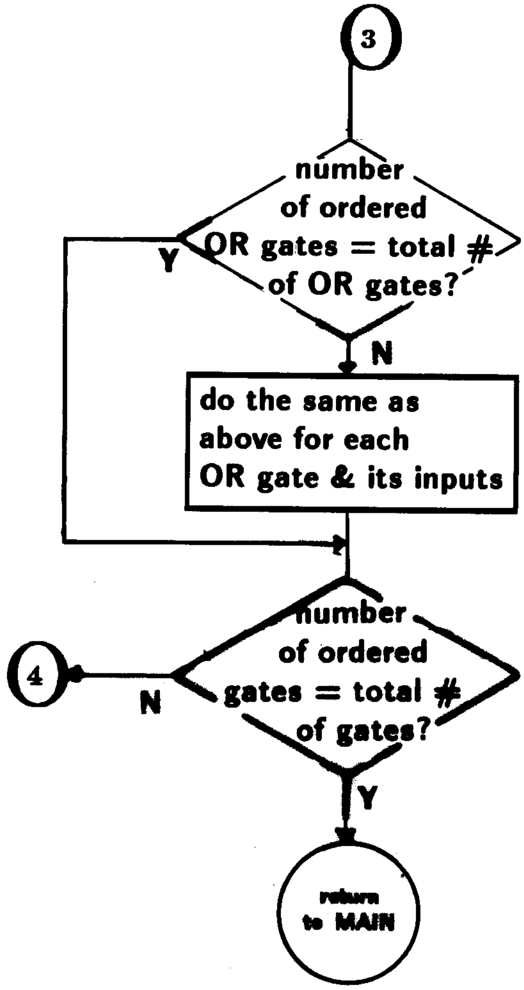

MAIN
Initialize random number generator input
Read: number of subsystems, time step size, number of time steps/history, number histories
Read: number of operating units for each subsystem
For each subsystem, read: failure rate, repair time, number of spares, deferred repair option, number of on-line redundant units
Print out the above information for each subsystem
Read: number of AND gates, number of OR gates
Print out number of subsystems, time step size, number of time steps/history, number of histories
1

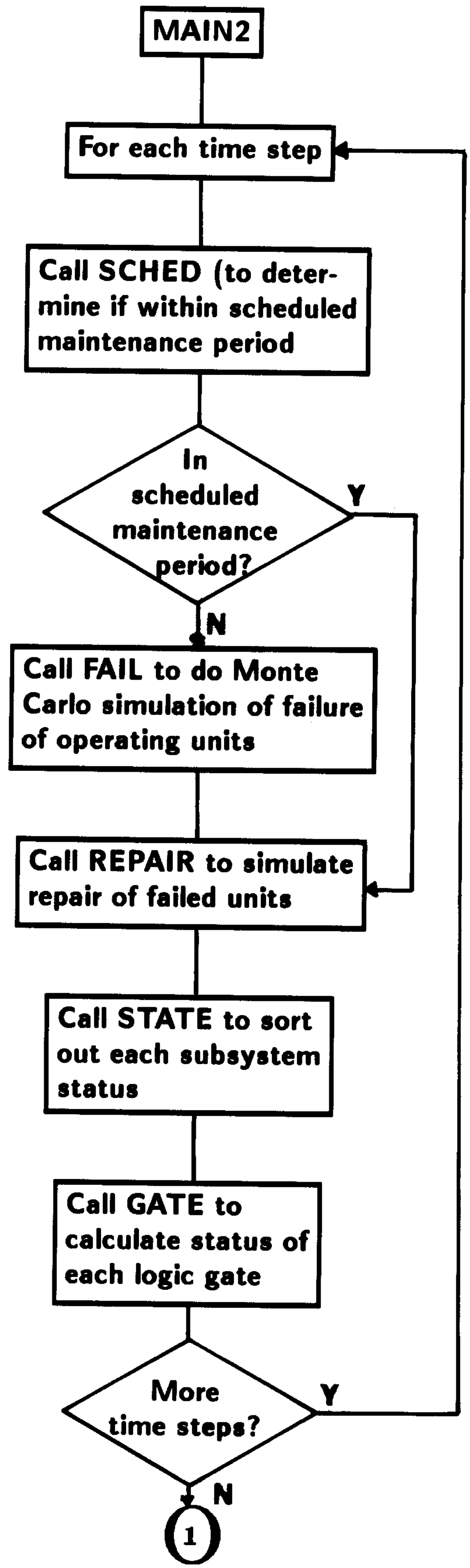

1
Print out number of units for each subsystem (normally operating)
Print out number of AND gates, number of OR gates
For each AND gate, read: number of subsystem, number of AND gates, number of OR gates at its inputs
For each AND gate and for each of its subsystem inputs: read the ID # of each subsystem at inputs
For each OR gate and for each of its AND gate inputs, read the ID # of each AND gate at inputs
For each AND gate and for each of its OR gate inputs, read the ID numbers of such OR gates
Repeat the procedure for the OR gates
2

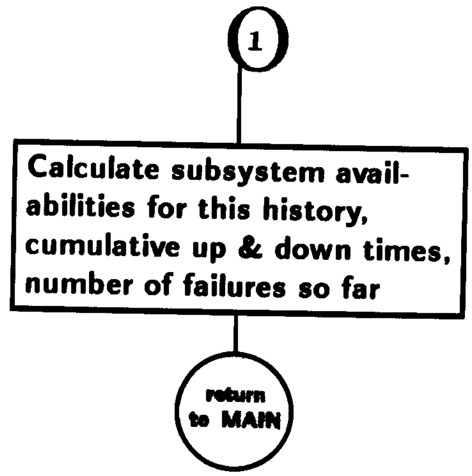

2
Print tree information
Read: number of subsystems with demand failure rates
Read the ID numbers of such subsystems
Read: duration of yearly scheduled shutdown period
Call ORDGAT, to order sequence of logic gate calculation
For each history:
Call MAIN2 which controls execution of rest of program for each time step
Calculate cumulative gate availabilities
More histories?
Y
N
Calculate average subsystem availability and average gate availabilities
Calculate time dependent availability of top gate; standard deviations
Print out number of failures, downtime and uptime for each subsystem (all histories); gate availabilities
STOP

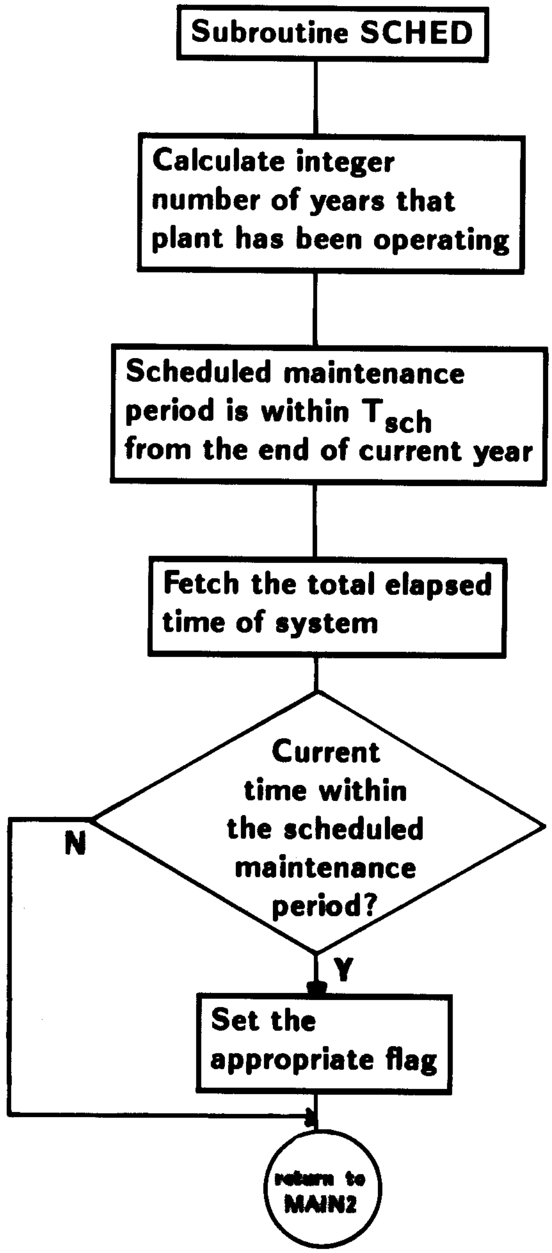

Subroutine ORDGAT
increase AND gate pointer
other gates connected to inputs of gate?
Y
N
put this gate in the ordering array igord and array ianacc
checked all AND gates?
N
Y
do the same as above, but for the OR gates
4
increment the loop index
number of ordered AND gates = total # of AND gates?
Y
3
N
2
increment the AND gate pointer
1

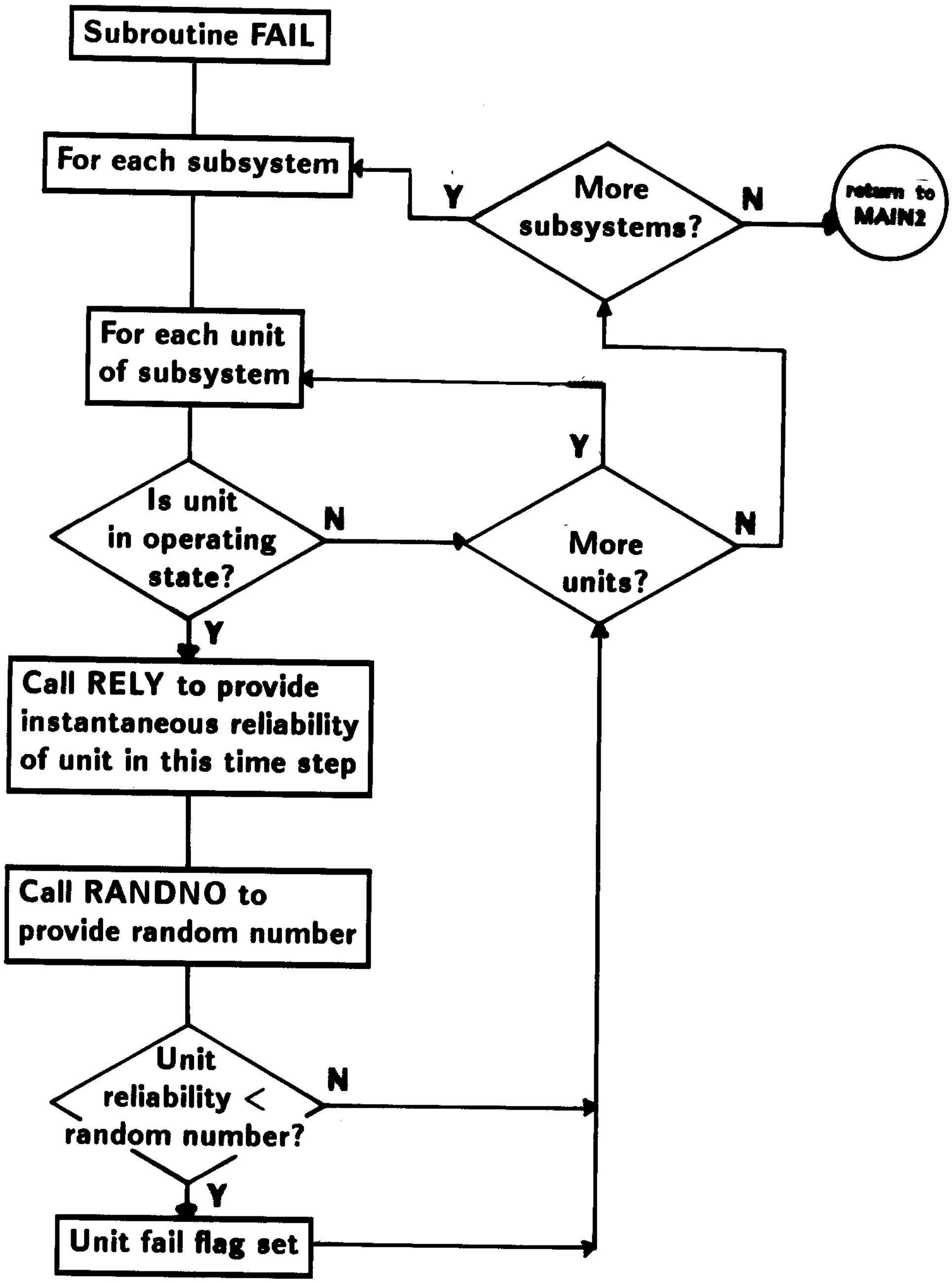

1
2
Y
this AND gate already accounted for?
N
increment the AND gate input to this AND gate
2
N
Has this AND gate (at the input) been accounted for (ordered)?
Y
all AND gate inputs to this AND gate been checked?
N
Y
increment the OR gate input to this AND gate
2
N
Has this OR gate been ordered?
Y
All OR gate inputs to this AND gate have been checked?
N
Y
3

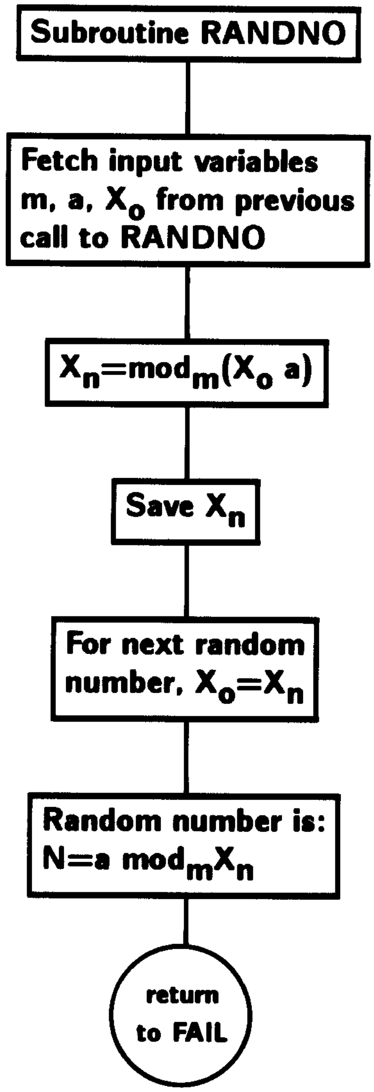

3
number of ordered OR gates = total # of OR gates?
Y
N
do the same as above for each OR gate & its inputs
number of ordered gates = total # of gates?
4
N
Y
return to MAIN

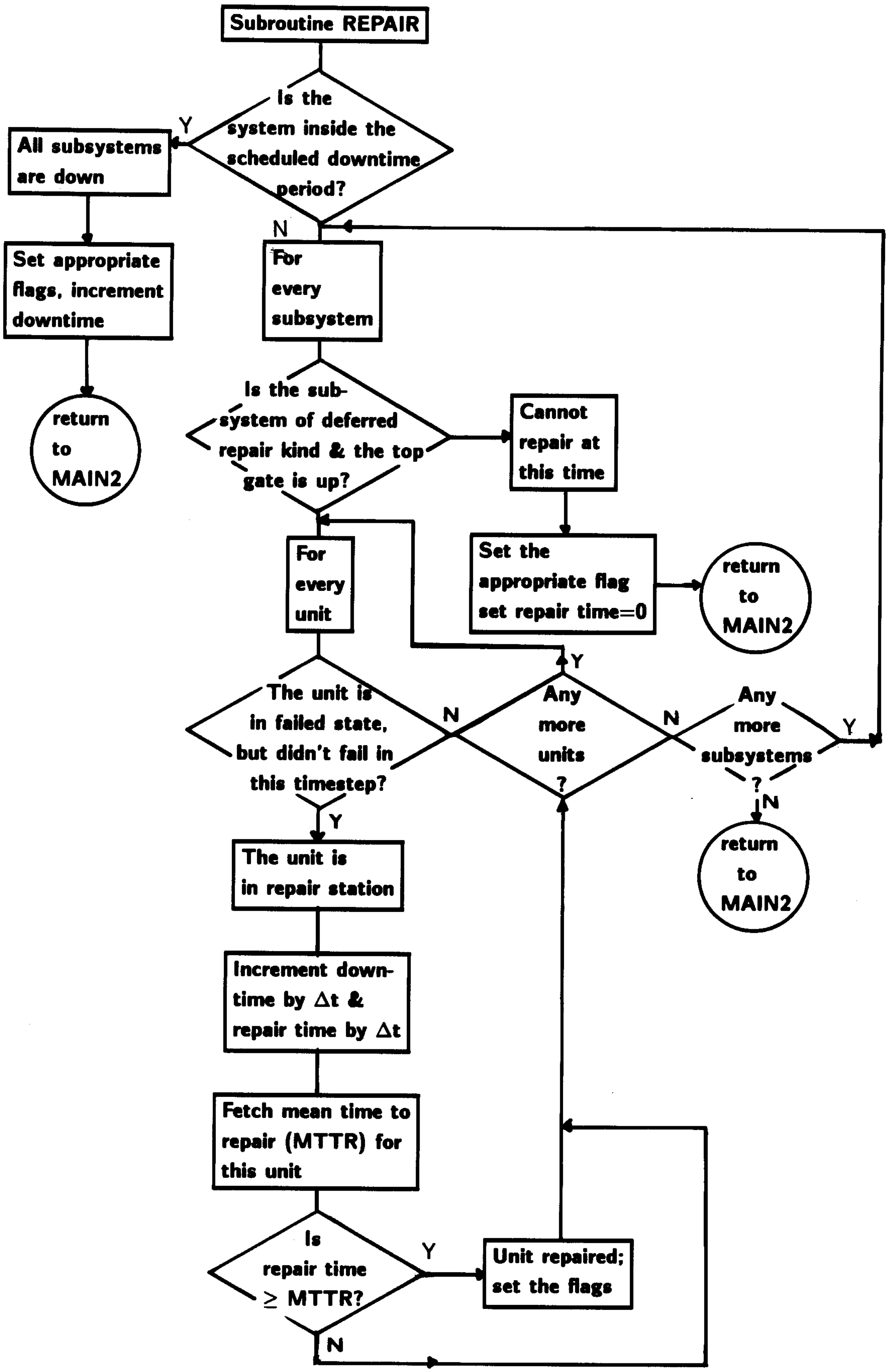

MAIN2
For each time step
Call SCHED (to determine if within scheduled maintenance period
In scheduled maintenance period?
Y
N
Call FAIL to do Monte Carlo simulation of failure of operating units
Call REPAIR to simulate repair of failed units
Call STATE to sort out each subsystem status
Call GATE to calculate status of each logic gate
More time steps?
Y
N
1


1
Calculate subsystem availabilities for this history, cumulative up & down times, number of failures so far
return to MAIN

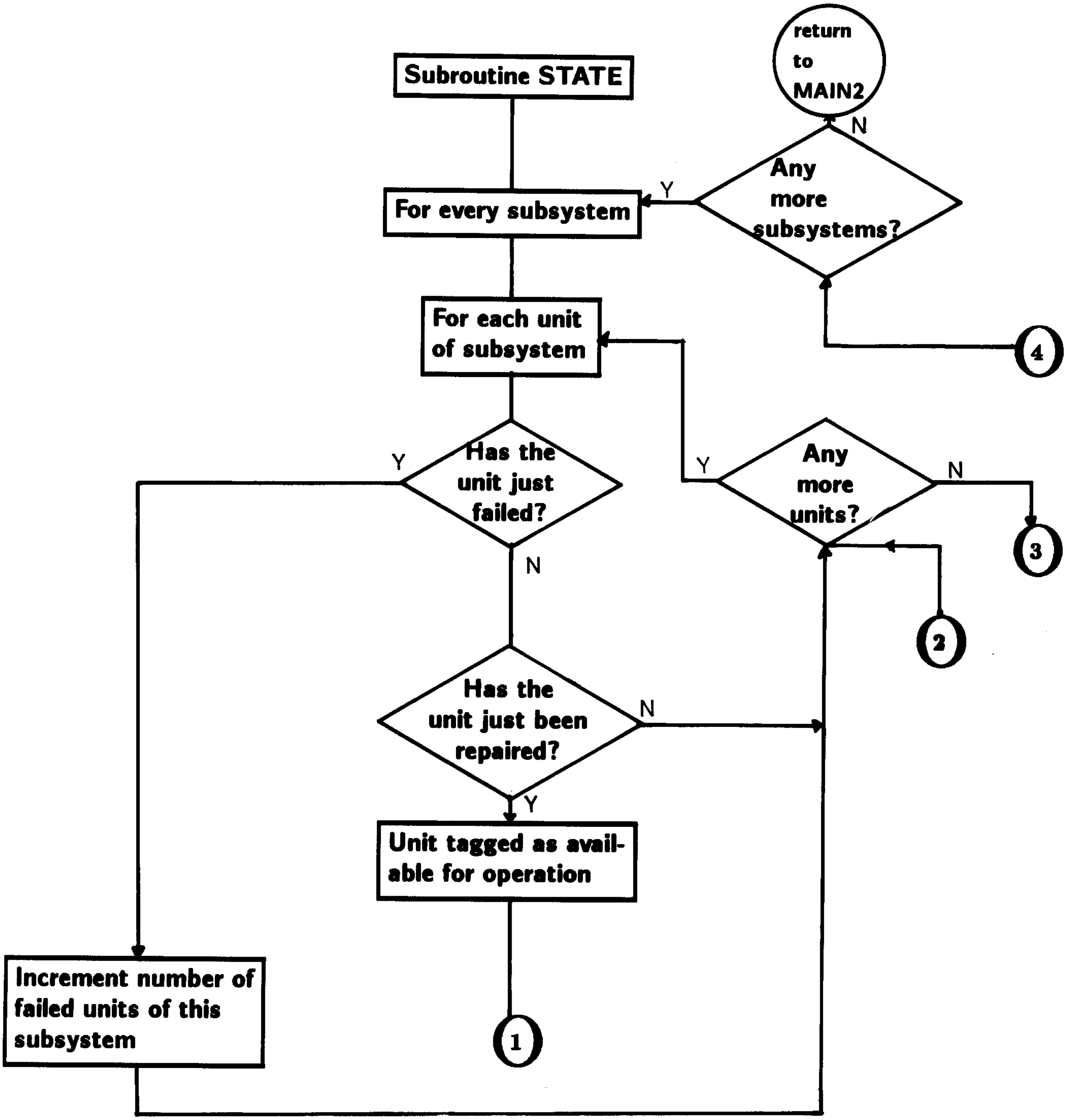

Subroutine SCHED
Calculate integer number of years that plant has been operating
Scheduled maintenance period is within T_{sch} from the end of current year
Fetch the total elapsed time of system
Current time within the scheduled maintenance period?
N
Y
Set the appropriate flag
return to MAIN2

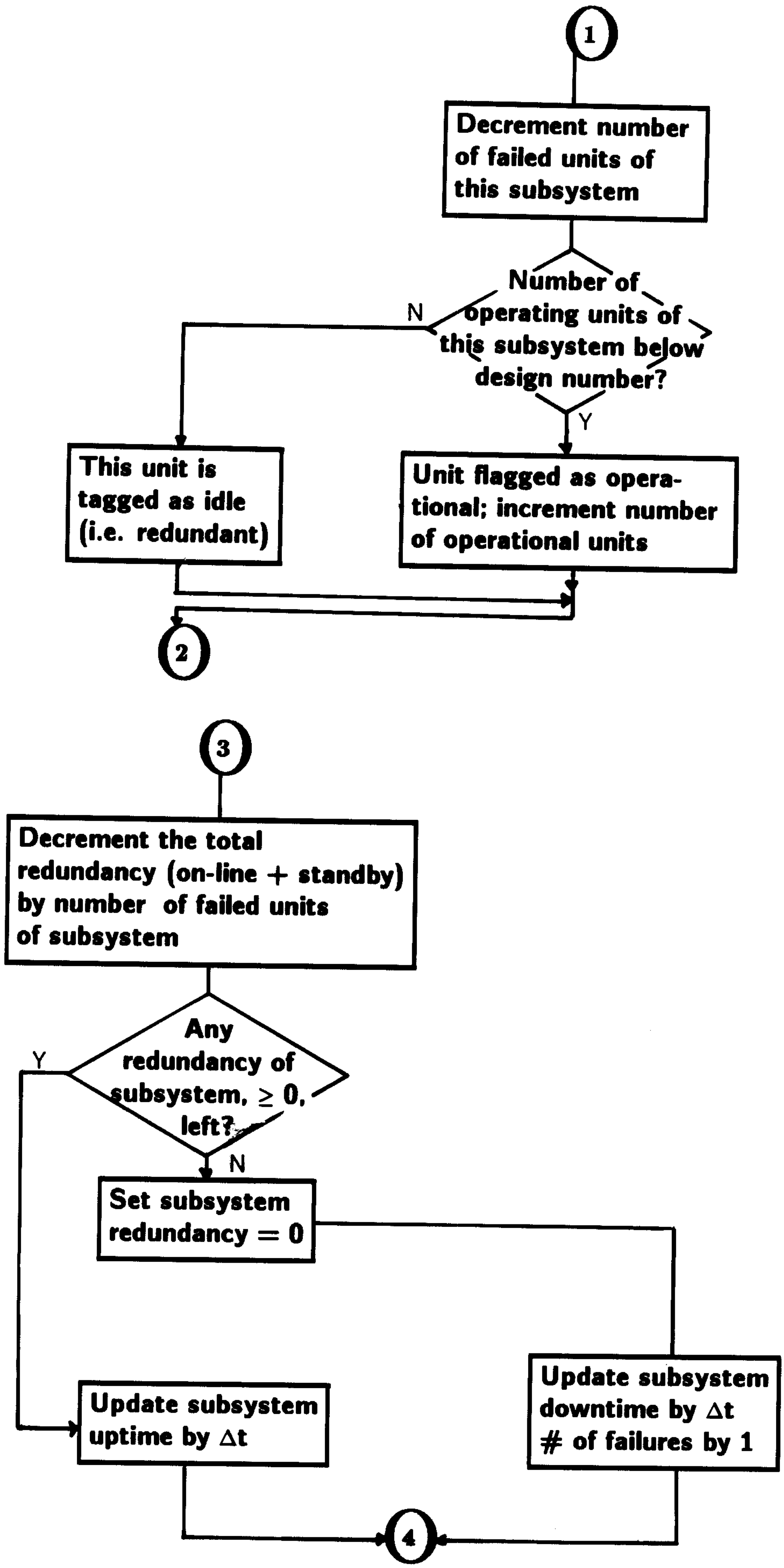

Subroutine FAIL
For each subsystem
More subsystems?
Y
N
return to MAIN2
For each unit of subsystem
Is unit in operating state?
N
Y
More units?
Y
N
Call RELY to provide instantaneous reliability of unit in this time step
Call RANDNO to provide random number
Unit reliability < random number?
N
Y
Unit fail flag set

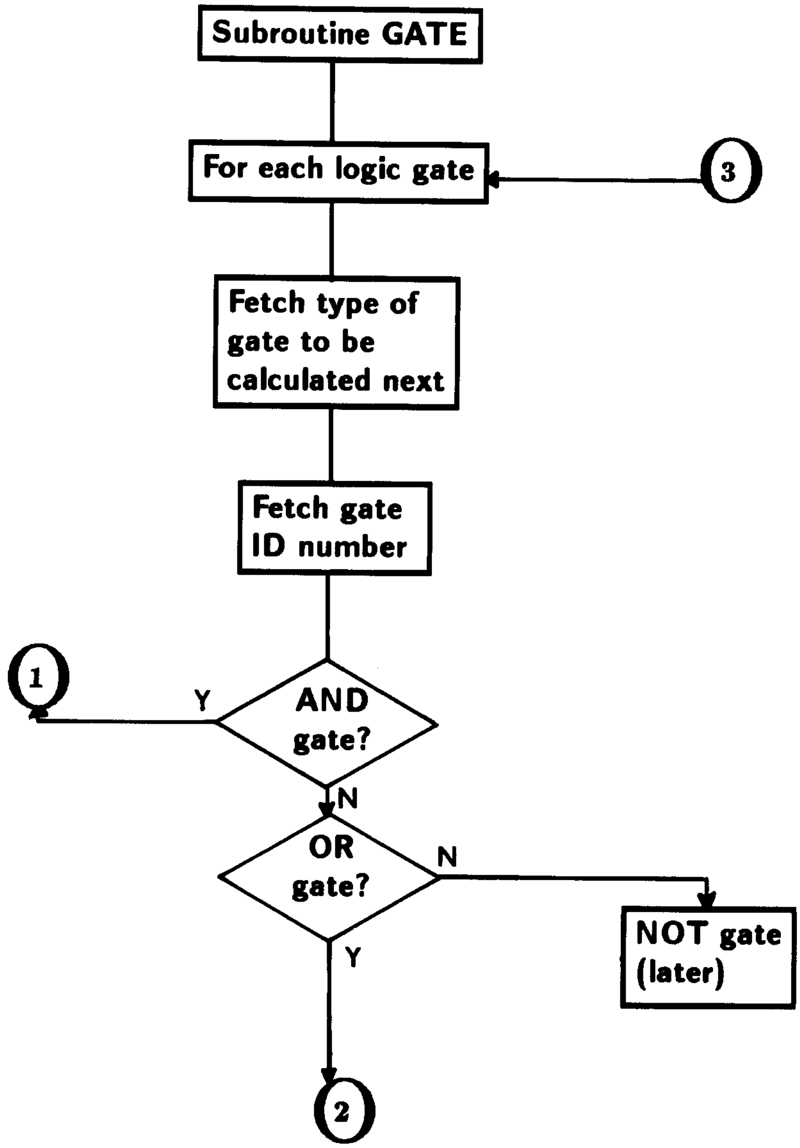

Subroutine RANDNO
Fetch input variables m, a, X_o from previous call to RANDNO
$X_n = mod_m(X_o\ a)$
Save X_n
For next random number, $X_o = X_n$
Random number is: $N = a\ mod_m X_n$
return to FAIL

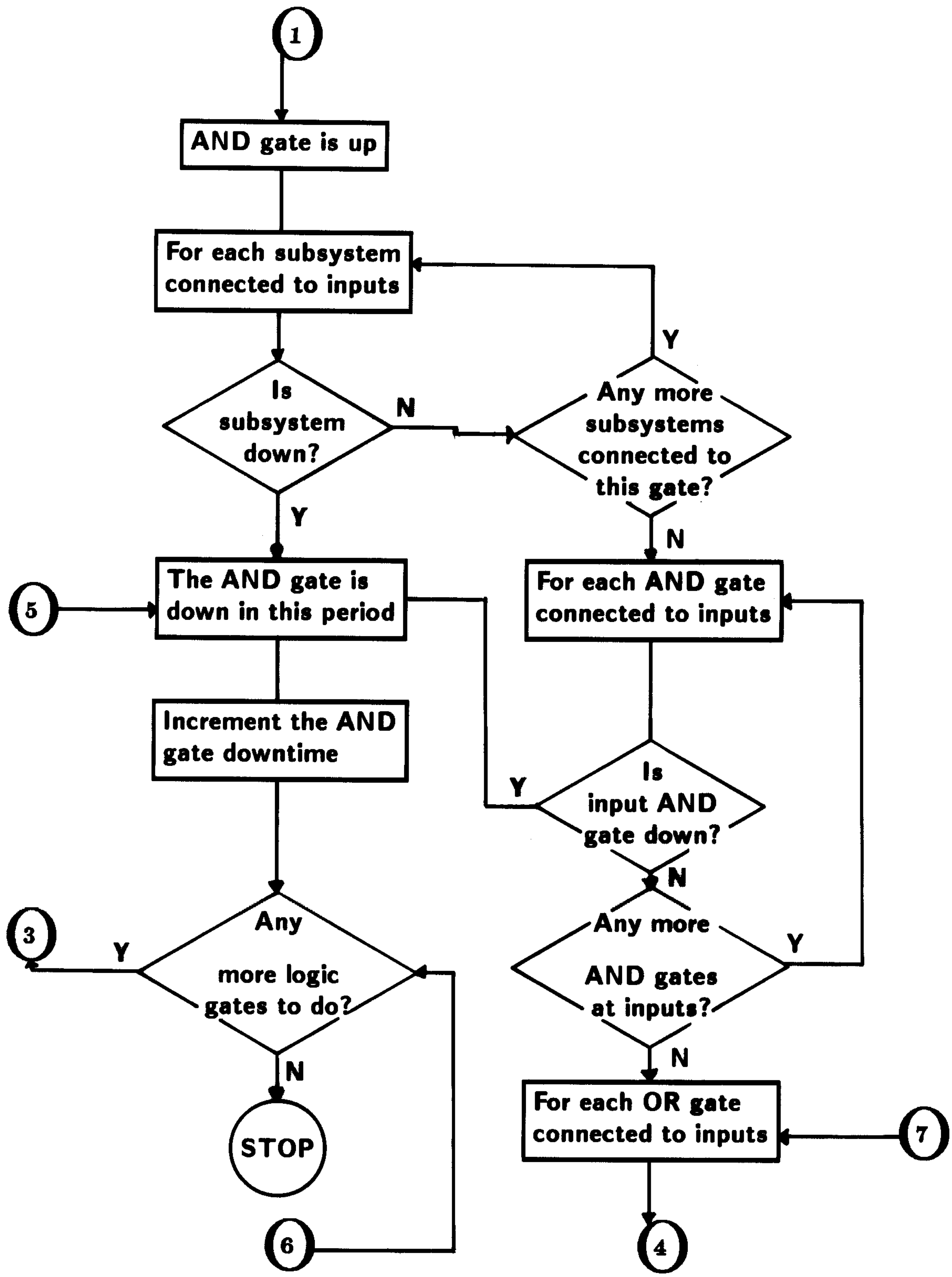


Subroutine REPAIR
Is the system inside the scheduled downtime period?
Y
N
All subsystems are down
Set appropriate flags, increment downtime
return to MAIN2
For every subsystem
Is the sub-system of deferred repair kind & the top gate is up?
Cannot repair at this time
Set the appropriate flag set repair time=0
return to MAIN2
For every unit
The unit is in failed state, but didn't fail in this timestep?
N
Y
Any more units ?
Y
N
Any more subsystems ?
Y
N
return to MAIN2
The unit is in repair station
Increment down-time by Δt & repair time by Δt
Fetch mean time to repair (MTTR) for this unit
Is repair time ≥ MTTR?
Y
N
Unit repaired; set the flags

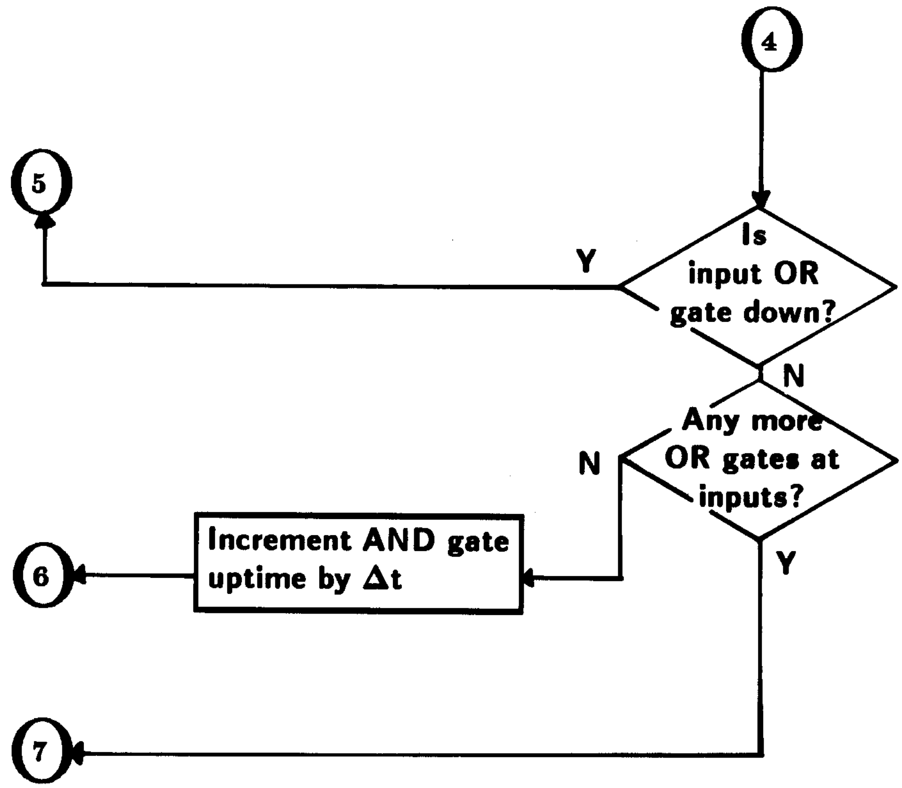

Subroutine STATE
return to MAIN2
N
Any more subsystems?
Y
For every subsystem
For each unit of subsystem
4
Has the unit just failed?
Y
N
Any more units?
Y
N
3
2
Has the unit just been repaired?
N
Y
Unit tagged as available for operation
Increment number of failed units of this subsystem
1

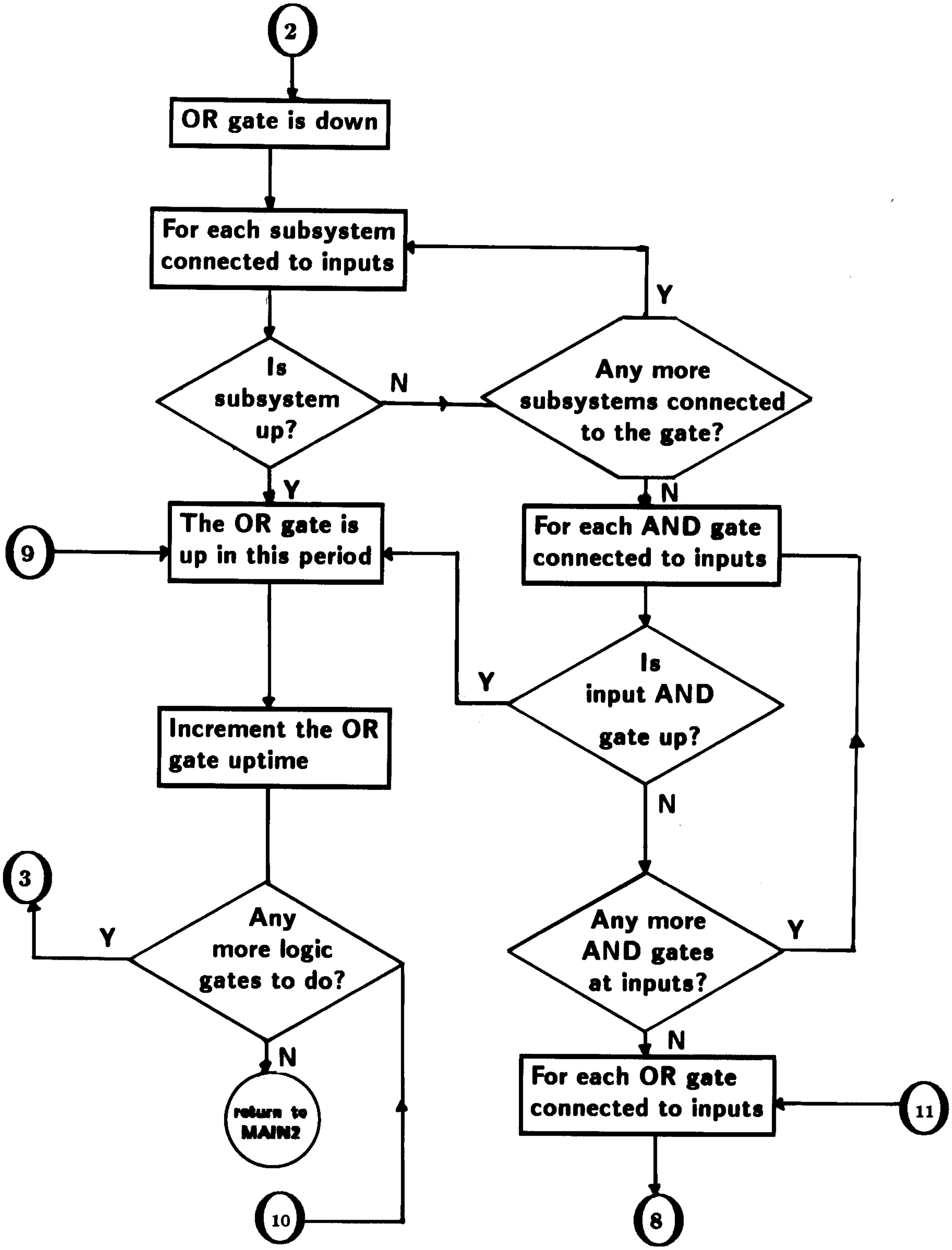

1
Decrement number of failed units of this subsystem
Number of operating units of this subsystem below design number?
N
Y
This unit is tagged as idle (i.e. redundant)
Unit flagged as operational; increment number of operational units
2
3
Decrement the total redundancy (on-line + standby) by number of failed units of subsystem
Any redundancy of subsystem, ≥ 0, left?
Y
N
Set subsystem redundancy = 0
Update subsystem uptime by Δt
Update subsystem downtime by Δt # of failures by 1
4

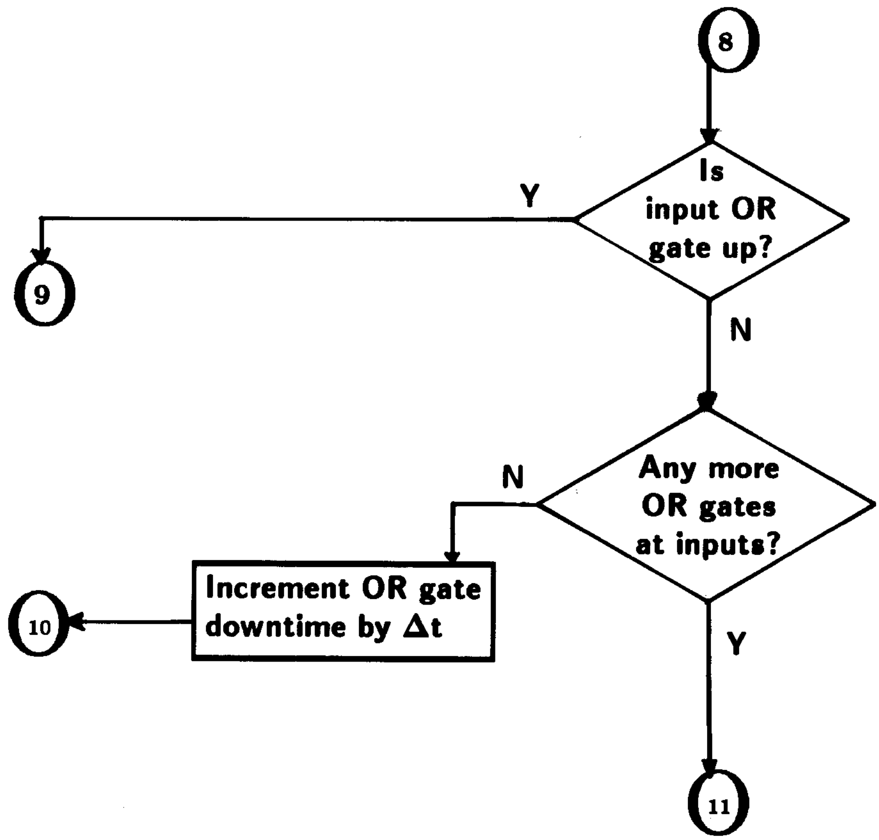

Subroutine GATE
For each logic gate
3
Fetch type of gate to be calculated next
Fetch gate ID number
1
Y
AND gate?
N
OR gate?
N
NOT gate (later)
Y
2



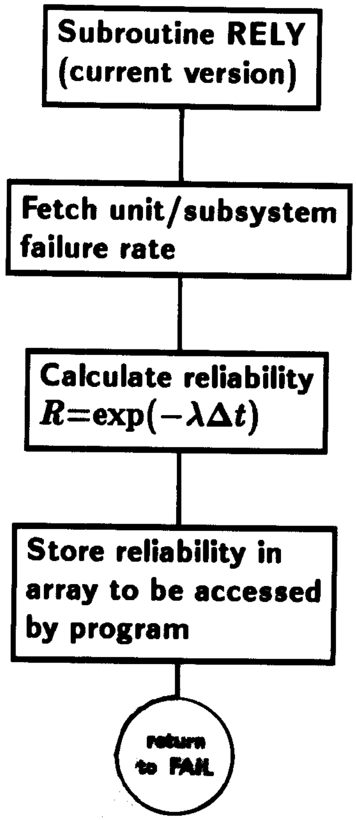
1
AND gate is up
For each subsystem connected to inputs
Is subsystem down?
N
Y
Any more subsystems connected to this gate?
Y
N
5
The AND gate is down in this period
For each AND gate connected to inputs
Increment the AND gate downtime
Y
Is input AND gate down?
N
3
Y
Any more logic gates to do?
Any more AND gates at inputs?
Y
N
N
STOP
For each OR gate connected to inputs
7
6
4

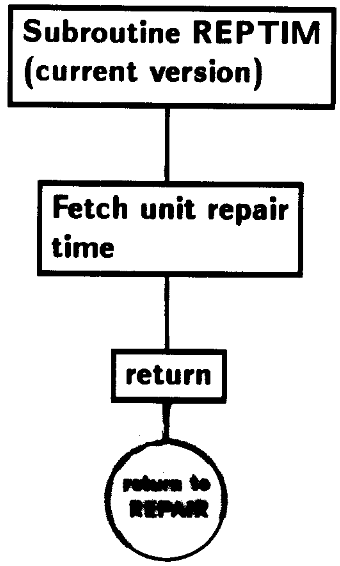

4
5
Y
Is
input OR
gate down?
N
Any more
OR gates at
inputs?
N
Increment AND gate
uptime by Δt
6
Y
7


2
OR gate is down
For each subsystem connected to inputs
Is subsystem up?
N
Y
Any more subsystems connected to the gate?
Y
N
9
The OR gate is up in this period
For each AND gate connected to inputs
Y
Is input AND gate up?
N
Increment the OR gate uptime
3
Y
Any more logic gates to do?
Any more AND gates at inputs?
Y
N
N
return to MAIN2
For each OR gate connected to inputs
11
10
8



8
Is
input OR
gate up?
Y
9
N
Any more
OR gates
at inputs?
N
Increment OR gate
downtime by Δt
10
Y
11


Subroutine RELY (current version)
Fetch unit/subsystem failure rate
Calculate reliability $R=\exp(-\lambda\Delta t)$
Store reliability in array to be accessed by program
return to FAIL


Subroutine REPTIM (current version)
Fetch unit repair time
return
return to REPAIR

PROGRAM LISTING

```
      subroutine main
c   This is the availability program AVSYS.  On the MFE network it is
c   stored in .test1 refus6 of user 1244.  The execution controlee is xrefus6
c   in the same directory and is also accessible.  The source code version
c   with comments is .test1 refus6c.
       parameter (ng=100,ni=20,ns=100,ng3=300,nts=1000,nh=10000)
      dimension gavand(ng),gavor(ng)
       dimension angtdt(ng),orgtdt(ng),angttf(ng),orgttf(ng)
      dimension aclk(ns,3)
      dimension ipd(ns)
      dimension ian(ng,3),ior(ng,3)
      dimension cand(ng),cor(ng),iasub(ng,ni),iaand(ng,ni)
      dimension iaor(ng,ni),iosub(ng,ni),ioand(ng,ni),ioor(ng,ni)
      dimension ifail(ns),mttr(ns)
      dimension redun(ns,3),mofn(ns),imrpr(ns),flrt(ns)
      dimension avail(ns)
      dimension gclk(ng3,2,3),igord(ng3,2)
      dimension numb(ns)
      dimension avgava(ng),avgavo(ng)
       dimension uptm15(nts),av15(nts),avdt15(nts),avnf15(nts)
       dimension dntm15(nts),totf15(nts),sigca1(nh),
     $ sigca2(nh)
       dimension iprdem(ns),igate(ng3),itype(ng3)
      common/mra/mmnnyy,a,xo
       common/mm2/avail,trials,timtrl,avav(ns),aclk
     $,iitx,dntm15,totf15,sigca1,sigca2
      common/mg/iasub,iosub
      common/mm2g/gclk
      common/mog/iaand,iaor,ioand,ioor,ian,ior
       common/mm2og/numand,numor
       common/mfrs/hist
       common/mm2frs/systs,numb,redun
       common/mr/imrpr
       common/mfsgrr/delt
       common/mm2sr/mofn
       common/mgs/ifail
       common/mrl/flrt
       common/mrt/mttr
       common/msc/tsdy
      integer systs,timtrl,trials,redun
      integer a
       integer hist
      real mttr,mttf(ns)
       call link("unit2=(rangen,access=rw),unit5=(input,open),
     $ unit6=(output,create,text)//")
c   This call establishes the link between units and files; rangen is
c   a one record file (gets overwritten at the end of the run), which
c   contains the 'random' starting point (parameter xo) from the last
c   random number generated from the last run.  The user should put
c   something in this file (format f9.1) the first time around, even
c   if no read from file rangen is desired, otherwise an error message
c   results.  The program automatically writes the last xo used into
c   rangen at the end of the run.  File input contains the input data
```

```
c    and file output will contain the output.
        mmnnyy=2**20
        a=2**10 + 3
        xo=566387.
c    mmnnyy,a and x0 are the internally specified random number seed parameters
         do 1050  i=1,ng
         angtdt(i)=0.
         angttf(i)=0.
         orgtdt(i)=0.
 1050    orgttf(i)=0.
c    initialize various matrices
        read(5,11010)systs,delt,timtrl,trials
c read number of subsystems, time step size, number of time steps, number
c of histories
         do 2111 k=1,systs
 2111    read(5,11030) j,flrt(j),mttr(j)
c for each subsystem, read the subsystem ID number, its mean time to failure,
c mean time to repair
         do 2112  k=1,systs
 2112    read(5,11020) j,numb(j),redun(j,1),imrpr(j),mofn(j)
c    For each subsystem, read the subsystem ID number, number of identical
c    units designed to operate, number of spares, immediate repair option
c    (1 for immediate repair, 0 for deferred repair), and number of online
c    redundant units.
        write(6,10)
         do 4391  i=1,systs
 4391    mttf(i)=1./flrt(i)
         write(6,20)(j,mttf(j),mttr(j),j=1,systs)
c    output subsystem ID number, its mean time to failure, mean time to repair,
c    for every subsystem.
        write(6,5310)
        read(5,11040)numand,numor
c    number of and gates, number of or gates
        if(numand .eq. 0)goto 3351
        do 1935  i=1,numand
 1935   avgava(i)=0.
 3351   if(numor .eq. 0)goto 3359
        do 1936  i=1,numor
 1936   avgavo(i)=0.
3359    do 1937  i=1,systs
        aclk(i,1)=0.
        aclk(i,2)=0.
        aclk(i,3)=0.
 1937   avav(i)=0.
         do 19357  i=1,timtrl
         totf15(i)=0.
19357    dntm15(i)=0.
c    initialize some matrices
        write(6,53)systs,delt,timtrl,trials
c    echo print the number of subsystems, time step size, number of
c    time steps, number of histories.
        write(6,5310)
        write(6,60)
```

```
      write(6,65)(j,numb(j),redun(j,1),imrpr(j),mofn(j),j=1,systs)
c   for each subsystem, echo print its ID number, number of identical units
c   designed to operate, number of spares, immediate repair option, number
c   of units actively redundant.
      write(6,5310)
      write(6,70)numand,numor
c   print number of and gates, number of or gates in the system.
       write(6,11290)
      if(numand .eq. 0)goto 101
       write(6,11300)
       do 5197  k=1,numand
 5197  read(5,11050) i,(ian(i,j),j=1,3)
c    for each and gate, read its sequential ID number, number of
c    subsystems, number of and gates and number of or gates connected
c    to it.
      do 11   i=1,numand
      ll=ian(i,1)
      if(ll .eq. 0)goto 201
      read(5,11060)(iasub(i,l),l=1,ll)
c   for each and gate, read the ID number of each subsystem connected
c   to it.
       write(6,11320)i,(iasub(i,l),l=1,ll)
c   write out the ID number of the AND gate and the ID numbers of
c   all the subsystems connected to it.
 201  mm=ian(i,2)
      if(mm .eq. 0)goto 202
      read(5,11070)(iaand(i,l),l=1,mm)
c   read the ID number of each and gate connected to this and gate.
       write(6,11330)i,(iaand(i,l),l=1,ll)
c   print out the and gate ID number and the ID numbers of the and gates
c   connected to it.
 202  nn=ian(i,3)
      if(nn .eq. 0)goto 11
      read(5,11080)(iaor(i,l),l=1,nn)
c   read the ID number of each or gate connected to this and gate
       write(6,11340)i,(iaor(i,l),l=1,nn)
c   print out the and gate ID number, the ID numbers of all the or gates
c   connected to it.
11    continue
  101 if(numor .eq. 0)goto 1021
       write(6,11310)
       do 5198   k=1,numor
c   the same procedure as above is now repeated for the or gates;
c   input data is read in and echo printed as for the and gates.
 5198  read(5,11090) i,(ior(i,j),j=1,3)
      do 12   i=1,numor
      ll=ior(i,1)
      if(ll .eq. 0)goto 203
      read(5,11100)(iosub(i,l),l=1,ll)
       write(6,11320)i,(iosub(i,l),l=1,ll)
 203  mm=ior(i,2)
      if(mm .eq. 0)goto 204
      read(5,11110)(ioand(i,l),l=1,mm)
```

```
        write(6,11330)i,(ioand(i,l),l=1,mm)
 204   nn=ior(i,3)
       if(nn .eq. 0)goto 12
       read(5,11120)(ioor(i,l),l=1,nn)
        write(6,11340)i,(ioor(i,l),l=1,nn)
 12    continue
       write(6,5320)
       write(6,5310)
1021   do 955 i=1,systs
 955   ipd(i)=0
       read(5,10010)ndfr
c   read the number of subsystems with per demand failure rate.
       do 953 i=1,ndfr
       read(5,10010) n
c   read the ID numbers of such subsystems.
        iprdem(i)=n
 953   ipd(n)=1
       do 954  i=1,systs
 954   if(ipd(i) .eq. 1)flrt(i)=flrt(i)/delt
c   this adjusts the failure rates of subsystems with per demand failures.
        write(6,11390)
        if(ndfr .eq. 0)goto 3131
        write(6,11400)(iprdem(i),i=1,ndfr)
c   the id numbers of subsystems with per demand failure rates are
c   printed out.
 3131   write(6,11500)
       read(5,10020)tsdy
c   read the duration of the annual scheduled maintenance period.
        write(6,11410)tsdy
c   print out the duration fo the annual scheduled maintenance period.
        read(5,10030)nran
c   read the random number generator option.
        goto(1025,1022,1023),nran
 1025   goto 1024
c   if 1, the program uses the internal random number seed parameters,
c   specified above.
 1022   read(2,10040)xo
        backspace 2

c   if 2 read the random starting parameter xo (saved from the last
c   number generated in the last run)
 backspace one record in
c   file rangen, so that the next xo write overwrites the previous
c   one and the file doesn't keep growing.
        goto 1024
 1023   read(5,10050)lgm,a,xo
        m=2.**lgm
c   if 3 read the user specified seed parameters.
 1024   call ordgat(idgate,kdgate)
c   ordgat will arrange the order in which the gates will have
c   to be calculated.
 102   do 50  iitx=1,trials
c   for each history ...
```

```
       hist=iitx
       call main2(idgate,kdgate)
c   main2 is the routine in charge of computation for each history
       ntot=numand+numor
c   ntot is the total number of logic gates
       if(ntot .eq. 0)goto 20566
       do 5500  i=1,ntot
       kk1=ifix(gclk(i,2,2))-1
c   kk1 is the type of the ith gate in the array gclk
c   1 for and gate, or for or gate
       if(kk1 .eq. 0)goto 5500
       mm=ifix(gclk(i,2,1))
c   mm is the sequential ID number of the ith gate entry in
c   array gclk
       goto(19311,19312),kk1
19311 gavand(mm)=gclk(i,1,1)/(gclk(i,1,1)+gclk(i,1,2))
        angtdt(mm)=angtdt(mm)+gclk(i,1,2)
        angttf(mm)=angttf(mm)+gclk(i,1,3)
c   gavand is the availability of and gate mm, angtdt is its
c   accumulated downtime (through all the previous histories),
c   and angttf is its accumulated number of failures.
       goto 5500
19312  gavor(mm)=gclk(i,1,1)/(gclk(i,1,1)+gclk(i,1,2))
        orgtdt(mm)=orgtdt(mm)+gclk(i,1,2)
        orgttf(mm)=orgttf(mm)+gclk(i,1,3)
c   gavor is the average availability (in this history) of or gate
c   mm, orgtdt is its accumulated downtime, and orgttf is its
c   accumulated number of failures.
 5500  continue
       if(numand .eq. 0)goto 511
       do 7171   i=1,numand
 7171  avgava(i)=avgava(i)+gavand(i)
c   avgava is the average and gate availability (averaged over all
c   the histories and the time steps) of and gate mm.
 511   if(numor .eq. 0)goto 50
       do 7373  i=1,numor
 7373  avgavo(i)=avgavo(i)+gavor(i)
c   avgavo is the average or gate availability (averaged over all
c   the histories and the time steps) of or gate mm.
 50    continue
        write(2,10000)xo
        write(6,11210)
        do 3957  i=1,ntot
        igate(i)=ifix(gclk(i,2,1))
 3957   itype(i)=ifix(gclk(i,2,2))
        write(6,11220)(i,igate(i),itype(i),i=1,ntot)
c   this will print out the order of calculation i for each
c   logic gate igate of type itype.
        write(6,5310)
        write(6,5320)
       write(6,5310)
       write(6,1010)
       do 75  j=1,systs
```

```
      avav(j)=avav(j)/trials
75    write(6,2010) j,aclk(j,3),aclk(j,1),aclk(j,2),avav(j)
c  print out subsystem sequential ID number, total number of
c  failures (in all histories), total up time, total downtime
c  and the subsystem average availability.
      itopn=ifix(gclk(ntot,2,1))
      itopt=ifix(gclk(ntot,2,2))-1
c  itopn is the sequential ID number of the top event gate,
c  and itopt is its type (and or or).
      write(6,5310)
      goto(5386,5387),itopt
 5386 write(6,11150)itopn
      goto 5388
 5387 write(6,11160)itopn
 5388 write(6,5310)
      write(6,11170)trials
      do 5379  i=1,timtrl
      uptm15(i)=delt-dntm15(i)/trials
c  uptime of top event gate in time step i, averaged over the
c  histories.
      av15(i)=1.-dntm15(i)/trials/delt
c  availability of top event gate in time step i averaged over the
c  histories.
      avdt15(i)=dntm15(i)/trials
c  downtime of top event gate in time step i averaged over the
c  histories.
 5379 avnf15(i)=totf15(i)/trials
c  number of failures of top event gate in time step i averaged over
c  the histories.
      write(6,5310)
      write(6,11180)(i,avnf15(i),avdt15(i),av15(i),
     $ i=1,timtrl)
      sigma1=0.
      sigma2=0.
      do 3191  i=1,trials
      sigma1=sigma1+(av15(1)+sigca1(i)/delt-1.)**2
      sigma2=sigma2+(av15(trials)+sigca2(i)/delt-1.)**2
 3191 continue
      sigma1=sqrt(sigma1)/trials
      sigma2=sqrt(sigma2)/trials
c  sigma1 is the standard deviation of top event gate availability
c  in the first time step, sigma2 is this deviation in the last
c  time step.
      write(6,11600)sigma1,sigma2
      if(numand .eq. 0)goto 5432
      write(6,5310)
      write(6,80)
      do 4818  i=1,numand
 4818 avgava(i)=avgava(i)/trials
      write(6,40)(i,avgava(i),angtdt(i),angttf(i),i=1,numand)
 5432 if(numor .eq. 0)goto 20566
      write(6,5310)
      write(6,90)
```

```
      do 3918  i=1,numor
3918  avgavo(i)=avgavo(i)/trials
      write(6,30)(i,avgavo(i),orgtdt(i),orgttf(i),i=1,numor)
20566 stop
10000 format(f9.1)
11150   format(' ','top event is and gate number',i3)
11160   format(' ','top event is or gate number',i3)
11170   format(' ','time dependent availability, averaged over',i6,
     $3x,'histories of top event gate'//' ','time step number',t20,
     $ 'number of failures',t40,'downtime',t60,'availability')
11180   format(' ',i3,t20,e10.4,t40,e10.4,t60,e10.4)
11210    format(' ','order of gate calculation',t40,'gate ID number',
     $  t60,'gate type (2 = and, 3 = or)')
11220    format(' ',i3,t40,i3,t60,i2)
11320  format(' '///' ',i3,2x,'subsystems',2x,20(i3,2x))
11330  format(' ',i3,2x,'and gates',3x,20(i3,2x))
11340  format(' ',i3,2x,'or gates',4x,20(i3,2x))
11290  format(' '//' ','logic gates interconnections')
11300  format(' '/' ','AND gate number')
11310  format(' '/' ','OR gate number')
11390  format(' '//' ','subsystems with per demand failures')
11400  format(' ',20i3)
11410  format(' '/,' ','duration (in hours per year) of scheduled',
     $1x,'outage',2x,e10.4)
11500  format(' ','none')
11600  format(' '//' ','standard deviation of top gate time',1x,
     $'dependent availability'/' ','first time step sigma',t40,
     $'last time step sigma'/' ',e10.4,t40,e10.4)
 1010  format(' ','subsystem'/' ','id No.',t10,'total failures',t30,'up
     $time',t50,'down time',t70,'availability')
 2010  format(' ',i3,t10,e10.4,t30,e10.4,t50,e10.4,t70,e10.4)
 10    format(' ','input data'//' ','subsystem number',t30,
     $'mean time to failure, hr',t70,'mean time to repair, hr'/' ',
     $t30,'or mean number of demands to failure')
 20      format(' ',i3,t30,e10.4,t70,e10.4)
 30    format(' ',1x,i3,t25,e10.4,t55,e10.4,t85,e10.4)
 40    format(' ',1x,i3,t25,e10.4,t55,e10.4,t85,e10.4)
 53    format(' ','number subsystems',t20,'delt',t40,'total no. of time s
     $teps',t70,'number of histories'/' ',i3,t20,e10.4,t40,i7,t70,i7)
 60      format(' ','subsystem number',t20,'design number of units',t50,
     $'number of spares',t70,'immediate repair',t90,'number of units'/
     $' ',t20,'operating',t70,'option',t90,'actively redundant')
 65      format(' ',i3,t20,i3,t50,i3,t70,i2,t90,i3)
 70    format(' ','number of and gates =',i3,3x,'number of or gates =',i3
     $ /' ')
 80      format(' ','and gate number',t25,'availability',t55,
     $ 'total down time',t85,'total number of failures')
 90      format(' ','or gate number',t25,'availability',t55,
     $ 'total down time',t85,'total number of failures')
 190   format(' ','final weights for monte carlo'/' ',
     $'subsystem id number',t30,'weight')
 5310  format(' ')
 5320  format(' ',//////////,'*******output*******')
```

```
 290   format(' ',i3,t30,e10.4)
11010 format(i3,f5.1,i3,i6)
11020 format(i3,i3,i3,i2,i3)
11030 format(i3,e10.4,f6.1)
11040 format(2i3)
11050 format(4i3)
11060 format(3x,20i3)
11070 format(3x,20i3)
11080 format(3x,20i3)
11090 format(4i3)
11100 format(3x,20i3)
11110 format(3x,20i3)
11120 format(3x,20i3)
10010 format(i3)
10020 format(f6.1)
10030 format(i2)
10040 format(f9.1)
10050 format(i3,f10.1,f9.1)
       end
       subroutine main2(idgate,kdgate)
c   subroutine main2 is in charge of computations for each history;
c   it initializes various matrices, and calls the subroutines that
c   compute the system's status for every time step.
        parameter (ns=100,ng=100,nu=20,nts=1000,nh=10000,ng3=300)
        dimension redun(ns,3),mofn(ns)
        dimension avail(ns)
       dimension icand(ng),icor(ng)
        dimension clk(ns,3)
        dimension ich(ns,nu),nop(ns),nfl(ns)
        dimension numb(ns),dntime(ns,nu)
        dimension iop(ns,nu),ifailu(ns,nu),irep(ns,nu)
        dimension dntm15(nts),totf15(nts)
        dimension sigca1(nh),sigca2(nh)
        common/m2frs/iop,ifailu,irep
        common/m2r/dntime
        common/m2s/clk,ich,nop,nfl
        common/mm2/avail,trials,timtrl,avav(ns),aclk(ns,3)
     $,iitx,dntm15,totf15,sigca1,sigca2
       common/m2gr/icand,icor
       common/mm2g/gclk(ng3,2,3)
        common/m2frss/it
        common/mm2frs/systs,numb,redun
        common/mm2sr/mofn
        common/mm2og/numand,numor
       integer systs,timtrl
       integer trials
       integer redun
        ntot=numand+numor
c   ntot is the total number of logic gates, and and or.
       do 251  i=1,ntot
       gclk(i,1,1)=0.
        gclk(i,1,3)=0.
 251   gclk(i,1,2)=0.
```

```
      do 100  j=1,systs

c   for each subsystem....
      avail(j)=0.
      redun(j,2)=mofn(j)+redun(j,1)
       nfl(j)=0
       nop(j)=numb(j)
       num=numb(j)
       do 200  i=1,num
       dntime(j,i)=0.
       ich(j,i)=0
       ifailu(j,i)=0
       iop(j,i)=1
 200   irep(j,i)=0
c   various matrices are initialized above
       num1=num+1
       num2=num+redun(j,1)
       do 300  i=num1,num2
       dntime(j,i)=0.
       ich(j,i)=0
       ifailu(j,i)=0
       iop(j,i)=0
 300   irep(j,i)=0
c   various matrices initialized; the meaning of these
c   matrices will be explained where they are used.
 100  continue
      do 150  j=1,systs
      do 150  i=1,3
 150  clk(j,i)=0.0
       if(numand .eq. 0)goto 51
       do 50  i=1,numand
 50    icand(i)=1
 51    if(numor .eq. 0)goto 52
       do 60   i=1,numor
 60    icor(i)=1
c   various matrices initialized above...
 52    index=timtrl
c   index is the number of time steps
      do 400   i=1,index
c   for each time step
       it=i
       call sched(iscflg)
c   subroutine sched determines if the system has entered
c   the annual scheduled downtime period.
       call repair(iscflg,idgate,kdgate)
c   subroutine repair updates the repair times of failed components,
c   and determines if the repair can be initiated.
       if (iscflg .eq. 1)goto 101
c   iscflg=1 means that the scheduled downtime period has
c   been entered, hence no operating failures are possible.
       call fail
c   subroutine fail does the Monte Carlo simulation of failures
c   of subsystems.
```

```
 101    call state
c   subroutine state determines the status of each subsystem in
c   a given time step.
        dt151t=gclk(ntot,1,2)
c   this is the top event gate downtime from the last time step.
        tf151t=gclk(ntot,1,3)
c   this is the top event gate number of failures from the last
c   time step.
        call gate
c   subroutine gate calculates the status of each logic gate
c   at the end of the time step.
        dntm15(i)=dntm15(i)+gclk(ntot,1,2)-dt151t
c   dntm15 is the cumulative downtime (later to be averaged
c   over all the histories) of top event gate in this time step.
        if(i .eq. 1)sigca1(iitx)=gclk(ntot,1,2)-dt151t
c   sigca1 will be used in the calculation of the first time step
c   standard deviation of the top event gate (see sigma1 and sigma2 in main).
        if(i .eq. timtrl)sigca2(iitx)=gclk(ntot,1,2)-dt151t
c   sigca2 will be used in the calculation of the last time step
c   standard deviation of the top event gate (see sigma1 and sigma2 in main).
        totf15(i)=totf15(i)+gclk(ntot,1,3)-tf151t
c   totf15 is the cumulative number of failures of top event gate
c   in this time step (later averaged over the histories).
 400    continue
 401  do 75  j=1,systs
      avail(j)=clk(j,1)/(clk(j,1)+clk(j,2))
c   availability of subsystem j (clk(j,1) is its uptime,
c   clk(j,2) its downtime).
      aclk(j,1)=aclk(j,1)+clk(j,1)
c   accumulated uptime of subsystem j
      aclk(j,2)=aclk(j,2)+clk(j,2)
c   accumulated downtime of subsystem j
      aclk(j,3)=aclk(j,3)+clk(j,3)
c   accumulated number of failures of subsystem j (over all
c   the previous histories).
 75   avav(j)=avav(j)+avail(j)
c   avav(j) is the average availability of subsystem j (see main).
      return
      end
      subroutine ordgat(idgate,kdgate)
c   subroutine ordgat determines the order in which the gates
c   are to be calculated; this assures that when a gate output
c   is calculated, all of its inputs have already been calculated.
       parameter (ng3=300,ng=100,ni=20)
      dimension igord(ng3,2),ianacc(ng),ioracc(ng)
      dimension ian(ng,3),ior(ng,3),iaand(ng,ni),iaor(ng,ni),
     $ioand(ng,ni),ioor(ng,ni)
      common/og/igord
      common/mog/iaand,iaor,ioand,ioor,ian,ior
       common/mm2og/numand,numor
      ijk=0
      itot=0
      jtot=0
```

```
      ktot=0
      if(numand .eq. 0)goto 101
      do 100  i=1,numand
      if(ian(i,2) .ne. 0 .or. ian(i,3) .ne. 0)goto 100
c   gates with only subsystem inputs will be considered first
c   (ian(i,2) is the number of and gate inputs, ian(i,3) is the
c   number of or gate inputs to and gate i).
      ijk=ijk+1
      igord(ijk,1)=i
c   igord is the array that keeps track of the order of
c   gates; and gate i is the ijk-th entry in this array,
c   because it has only subsystem input.
      igord(ijk,2)=2
c   this identifies the ijk-th entry as an and gate.
      ianacc(ijk)=i
c   ith and gate is the ijk-th entry in this array reserved only
c   for and gates
 100  continue
      itot=ijk
c   the number of and gates accounted for
 101  if(numor .eq. 0)goto 301
c   the same process is now repeated for the or gates
      do 200  j=1,numor
      if(ior(j,2) .ne. 0 .or. ior(j,3) .ne. 0)goto 200
      ijk=ijk+1
      igord(ijk,1)=j
      igord(ijk,2)=3
c   the ijk-th entry in the array igord has been identified as
c   the j-th or gate
      ioracc(ijk-itot)=j
c   ioracc is the ordering array reserved only for or gates
 200  continue
      jtot=ijk-itot
c   the number of or gates accounted for
 301  ktot=ijk-itot-jtot
      do 400   iii=1,10000
      if(itot .eq. numand)goto 2
c   if number of ordered and gates equals the total number of and gates,
c   skip this section and go to the or gates' section.
      do 500  i=1,numand
c   for every and gate....
      do 501   ii=1,itot
      if(ianacc(ii) .eq. i)goto 500
c   check if and gate i has already been ordered; if yes, skip
c   to the next and gate
 501  continue
c   if and gate i has not been ordered:
      mm=ian(i,2)
c   we will check each input to this and gate consisting of an and
c   gate.
c   mm is the total number of and gate inputs to the ith and gate.
      if(mm .eq. 0)goto 502
      do 505   j=1,mm
```

```
        do 504  k=1,itot
        if(ianacc(k) .eq. iaand(i,j))goto 505
c   checking to see if the jth and gate input to the ith and gate
c   is contained in array ianacc, which means it has been ordered;
c   in that case we can proceed and check the next and gate input
c   to the ith and gate.
 504    continue
        goto 500
c   if the jth and gate input to the ith and gate input has not been
c   ordered, the ith and gate itself cannot be ordered at this time,
c   so skip to the (i+1)th and gate.
 505    continue
 502    nn=ian(i,3)
c   nn is the number of or gate inputs to the ith and gate;
c   we will now check each or gate input in the same manner as the
c   and gate inputs above.
        if(nn .eq. 0)goto 511
        do 507  j=1,nn
        do 508  k=1,jtot
        if(ioracc(k) .eq. iaor(i,j))goto 507
 508    continue
        goto 500
 507    continue
 511    ijk=ijk+1
        itot=itot+1
        igord(ijk,1)=i
        igord(ijk,2)=2
        ianacc(itot)=i
c   if all the inputs of this ith and gate have been found to be already
c   ordered, then this gate can be ordered itself and put on top of
c   arrays igord and ianacc.
 500    continue
 2      if(jtot .eq. numor)goto 401
c   the same procedure as above is now repeated for the or gates, to
c   see if some of them can be ordered and put on top of arrays
c   igord and ioracc.
        do 600   i=1,numor
        do 601  ii=1,jtot
        if(ioracc(ii) .eq. i)goto 600
 601    continue
        mm=ior(i,2)
        if(mm .eq. 0)goto 602
        do 605  j=1,mm
        do 604  k=1,itot
        if (ianacc(k) .eq. ioand(i,j))goto 605
 604    continue
        goto 600
 605    continue
 602    nn=ior(i,3)
        if(nn .eq. 0)goto 611
        do 607  j=1,nn
        do 608  k=1,jtot
        if(ioracc(k) .eq. ioor(i,j))goto 607
```

```
 608   continue
       goto 600
 607   continue
 611   ijk=ijk+1
       jtot=jtot+1
       igord(ijk,1)=i
       igord(ijk,2)=3
       ioracc(jtot)=i
 600   continue
 401   ntot=numand+numor
       ijktot=itot+jtot
       if(ntot .eq. ijktot)goto 800
c   have all the gates been ordered?
 400   continue
c   if not, repeat the process....
 800    idgate=igord(ntot,1)
        kdgate=igord(ntot,2)
c   if yes, the gate on top of array igord is the top event (i.e.
c   output) gate, with sequential ID number idgate and of kind
c   kdgate (2 for an and gate, 3 for an or gate).
       return
       end
       subroutine randno(ano)
c   subroutine randno calculates a random number employing the
c   congruential recursive method.  The user (or the program)
c   supplies the three initial (seed) parameters a, m and xo
c   (see main).
       common/mra/m,a,xo
       integer a
       xn=xo*a-m*ifix(xo*a/m)
       xo=xn
c   this is the xo for generation of next random number.
       ano=(xn/m*a)-ifix(xn/m*a)
c   random number ano is the decimal part of xn/m*a.
       return
       end
       subroutine gate
c    subroutine gate determines gate outputs once the inputs
c    are given.
        parameter (ng3=300,ng=100,ns=100,ni=20)
       dimension gclk(ng3,2,3), ifail(ns),
     $ icand(ng),icor(ng),ior(ng,3),iosub(ng,ni),
     $ ioand(ng,ni),ioor(ng,ni),ian(ng,3),iasub(ng,ni)
     $ ,iaand(ng,ni),iaor(ng,ni),igord(ng3,2)
       common/mg/iasub,iosub
       common/m2gr/icand,icor
       common/og/igord
       common/mm2g/gclk
       common/mog/iaand,iaor,ioand,ioor,ian,ior
        common/mm2og/numand,numor
        common/mfsgrr/delt
        common/mgs/ifail
 311   ntot=numand+numor
```

```
c   total number of logic gates.
       if(ntot .eq. 0)goto 20566
       do 1001  i=1,ntot
c   for each gate:
       mm=igord(i,1)
       kk=igord(i,2)
c   mm is the sequential ID number of the ith gate to be calculated.
c   kk is the kind of that gate (2 for and, 3 for or).
       kk11=kk-1
        inofl=0
       gclk(i,2,1)=float(mm)
       gclk(i,2,2)=float(kk)
       goto (1,2),kk11
 1      if(icand(mm) .eq. 1)inofl=1
c   for an and gate:
c   inofl is a parameter that indicates when a gate changes status
c   from up to down (for calculating the number of gate failures).
c   icand(mm) is the status of the mm-th and gate (1 or 0).
        icand(mm)=1
       jj=ian(mm,1)
c   jj is the number of subsystem inputs to the mm-th and gate.
       if(jj .eq. 0)goto 207
       do 333  j=1,jj
c   check each subsystem input
       n=iasub(mm,j)
c   n is the ID number of j-th subsystem input to the mm-th
c   and gate.
       if(ifail(n) .eq. 1)goto 3341
c   if subsystem n is down so is the mm-th and gate.
 333   continue
       goto 207
 3341  icand(mm)=0
       goto 100
 207   mmm=ian(mm,2)
c   number of and gate inputs.
       if(mmm .eq. 0)goto 208
       do 334  j=1,mmm
       m=iaand(mm,j)
c   ID numbe of the j-th and gate input to the mm-th and gate.
       if(icand(m) .eq. 0)goto 33411
c   if this input is down, so is the mm-th and gate.
 334   continue
       goto 208
33411 icand(mm)=0
       goto 100
 208   nn=ian(mm,3)
c   nn is the number of or gate inputs to the mm-th and gate.
       if(nn .eq. 0)goto 100
       do 335  j=1,nn
c   check each or gate input
       n=iaor(mm,j)
c   n is the ID number of the j-th or gate input to the mm-th
c   or gate.
```

```
      if(icor(n) .eq. 0)goto 3351
c   if this input is down so is the and gate.
 335  continue
      goto 100
 3351 icand(mm)=0
      goto 100
 2     if(icor(mm) .eq. 1)inofl=1
c   the same process is repeated for the or gate
      icor(mm)=0
      jj=ior(mm,1)
      if(jj .eq. 0)goto 307
      do 433  j=1,jj
      n=iosub(mm,j)
      if(ifail(n) .eq. 0)goto 4341
 433  continue
      goto 307
 4341 icor(mm)=1
c   if only one subsystem input is up, so is the or gate.
      goto 100
 307  mmm=ior(mm,2)
      if(mmm .eq. 0)goto 308
      do 434  j=1,mmm
      m=ioand(mm,j)
      if(icand(m) .eq. 1)goto 43411
c   if only one and gate input is up, so is the or gate.
 434  continue
      goto 308
43411 icor(mm)=1
      goto 100
 308  nn=ior(mm,3)
      if(nn .eq. 0)goto 100
      do 435  j=1,nn
      n=ioor(mm,j)
      if(icor(n) .eq. 1)goto 4351
c   if only one or gate input is up, so is this or gate.
 435  continue
      goto 100
 4351 icor(mm)=1
 100  goto(193,293),kk11
c   for kk11=1, an and gate is in question, 2 for an or gate.
 193  if(icand(mm) .eq. 1)goto 194
      gclk(i,1,2)=gclk(i,1,2)+delt
c   update the downtime of the i-th gate.
       if(inofl .eq. 1)gclk(i,1,3)=gclk(i,1,3)+1.
c   this signifies change of state from up to down, so update
c   the number of failures of the ith gate.
      goto 1001
 194  gclk(i,1,1)=gclk(i,1,1)+delt
c   update the uptime of the ith gate.
      goto 1001
 293  if(icor(mm) .eq. 1)goto 294
      gclk(i,1,2)=gclk(i,1,2)+delt
       if(inofl .eq. 1)gclk(i,1,3)=gclk(i,1,3)+1.
```

```
      goto 1001
 294  gclk(i,1,1)=gclk(i,1,1)+delt
 1001 continue
c   calculate the status of the next gate ((i+1)th).
20566 return
      end
       subroutine fail
c   subroutine fail does the simulation of failures by comparing a
c   subsystem's time step reliability to a random number.
       parameter (ns=100,nu=20)
       dimension numb(ns)
       dimension redun(ns,3),iop(ns,nu),ifailu(ns,nu),irep(ns,nu)
       common/m2frs/iop,ifailu,irep
       common/mfrs/hist
       common/m2frss/it
       common/mm2frs/systs,numb,redun
       common/mfsgrr/delt
       integer systs
       integer redun
       integer hist
       do 100   i=1,systs
c   for each subsystem....
       num2=numb(i)+redun(i,1)
c   total number of units of subsystem i
       do 200   j=1,num2
       if(iop(i,j) .eq. 1 .and. ifailu(i,j) .eq. 0 .and.
     $ irep(i,j) .eq. 0)goto 250
c   is the unit j of subsystem i operating?
       goto 200
 250   call rely(i,j,r)
c   calculate its reliability r for time step delt.
       call randno(ano)
c   fetch a random number ano
       if(r .lt. ano)goto 350
       ifailu(i,j)=0
c   the unit has not failed in this time step
       goto 200
 350   ifailu(i,j)=1
c   the unit has failed in time step delt.
c   the fail flag is set for this unit.
 200   continue
 100   continue
       return
       end
       subroutine repair(iscflg,idgate,kdgate)
c   subroutine repair determines if a unit in repair has been
c   repaired and updates its up or down time.
       parameter (ns=100,ng=100,nu=20)
       dimension numb(ns),imrpr(ns)
       dimension icand(ng),icor(ng)
       integer redun(ns,3)
       dimension dntime(ns,nu),iop(ns,nu),ifailu(ns,nu),irep(ns,nu)
       common/m2frs/iop,ifailu,irep
```

```
      common/m2r/dntime
      common/mfrs/hist
      common/m2frss/it
      common/mm2frs/systs,numb,redun
      common/mr/imrpr
      common/mfsgrr/delt
      common/m2gr/icand,icor
      common/mm2sr/mofn(ns)
      integer systs
      integer hist
      iokdef=0
      kk=kdgate-1
c   the type of top gate is kk (1 for and, 2 for or).
      goto(1,2)kk
 1    if(icand(idgate) .eq. 0)iokdef=1
      goto 3
 2    if(icor(idgate) .eq. 0)iokdef=1
c   is the top gate down (i.e. the system down)?
c   if yes set the flag (allowing the repair of deferred
c   repair units).
 3    if(iscflg .eq. 0)goto 197
c   are we in the scheduled maintenance period?
c   if yes update the downtime of all the units and all the
c   subsystems and set the proper flags.
     do 300  i=1,systs
      num2=numb(i)+redun(i,1)
     do 400   j=1,num2
      iop(i,j)=1
      irep(i,j)=0
      ifailu(i,j)=1
 400  dntime(i,j)=dntime(i,j)+delt
 300  continue
      goto 101
 197  do 100   i=1,systs
c   for each subsystem
      num2=numb(i)+redun(i,1)
      do 200  j=1,num2
c   for each unit..
      if(imrpr(i) .eq. 0 .and. iokdef .eq. 0)goto 199
c   no immediate repair and the system is up (then we cannot repair
c   the subsystem i if it's failed).
      if(iop(i,j) .eq. 1 .and. ifailu(i,j) .eq. 1)goto 200
c   unit has just failed?
      if(ifailu(i,j) .eq. 1)goto 250
c   unit is down, but did not fail in this time step.
      goto 200
 250  iop(i,j)=0
c   unit is not up (i.e. not operating)
      irep(i,j)=1
c   unit is in repair
      call reptim(i,j,rt)
c   fetch the unit repair time rt.
      dntime(i,j)=dntime(i,j)+delt
```

```
c   update the unit downtime
        if(dntime(i,j) .ge. rt)goto 350
c   if this condition met the unit has been repaired.
        goto 200
 350    ifailu(i,j)=0
        dntime(i,j)=0.
c   if unit repaired reset the fail flag and the downtime
        goto 200
 199    irep(i,j)=0
        dntime(i,j)=0.
c   if unit is of deferred repair kind and the system is not down,
c   the repair flag is reset as is the downtime (no repair at this
c   time).
  200 continue
  100 continue
 101    return
        end
        subroutine state
c   subroutine state determines the state of each unit and
c   each subsystem and updates the subsystem uptime, downtime
c   and number of failures accordingly.
        parameter (ns=100,nc=200,nu=20)
        dimension numb(ns)
        dimension numr(ns)
        dimension clk(ns,3),redun(ns,3),mofn(ns),ifl(ns),ifail(ns)
        dimension syssta(ns,nc,2)
        dimension iop(ns,nu),ifailu(ns,nu),irep(ns,nu),ich(ns,nu)
        dimension nop(ns),nfl(ns)
        common/mfrs/hist
        common/m2frss/it
        common/mm2frs/systs,numb,redun
        common/mfsgrr/delt
        common/m2s/clk,ich,nop,nfl
        common/mm2sr/mofn
        common/mgs/ifail
        common/m2frs/iop,ifailu,irep
        integer sigmam,redun
        integer systs
        integer hist
        do 100  i=1,systs
c   for each subsystem
        numr(i)=0
        num2=numb(i)+redun(i,1)
c   number of units of subsystem i
        ifl(i)=0
        do 200    j=1,num2
c   for each unit ...
        sigmam=4*iop(i,j)+2*ifailu(i,j)+irep(i,j)+1
c   this will distinguish among the various possible flag
c   combinations.
        goto(4,104,204,304,404,504,604),sigmam
  4     numr(i)=numr(i)+1
c   no flag for the unit has been set, since the unit is a redundant one.
```

```
      goto 200
 104  nch=ich(i,j)+1
c   the unit has just been repaired; update the number of changes in
c   the unit's status.
      syssta(i,nch,1)=it*delt
c   note the time of change
      ich(i,j)=ich(i,j)+1
      syssta(i,nch,2)=1.
c   new status of the unit: redundant unit
      irep(i,j)=0
      if(nop(i) .ge. numb(i))goto 200
c   number of operating units greater or equal to the design number.
c   in which case the just repaired unit is redundant.
      syssta(i,nch,2)=2.
c   new status of the unit: non-redundant unit
      iop(i,j)=1
c   set the operating flag of the unit.
      nop(i)=nop(i)+1
c   increment the number of operating units of subsystem i
      goto 200
 204  nfl(i)=nfl(i)+1
c   the unit is in failed state, but not in the repair station.
c   increment the number of failed units of subsystem i.
      goto 200
 304  nfl(i)=nfl(i)+1
c   the unit is in failed state and being repaired.
      goto 200
404   goto 200
c   the unit is operating.
504   goto 200
c   the unit is operating and in repair (degraded performance, not
c   implemented at this time).
 604  nch=ich(i,j)+1
c   the unit has just failed.
c   update the number of changes.
      ich(i,j)=ich(i,j)+1
      syssta(i,nch,1)=it*delt
c   note the time of change
      syssta(i,nch,2)=0.
c   the new status of unit: failed
      iop(i,j)=0
c   reset the operating flag.
      nop(i)=nop(i)-1
c   decrement the number of operating units of subsystem i.
      nfl(i)=nfl(i)+1
c   increment the number of failed units of subsystem i.
 200  continue
      clk(i,1)=clk(i,1)+delt
c   update the subsystem's uptime.
      if(nop(i)-numb(i))1,2,3
 1    do 300  j=1,num2
c   number of operating units is less than the design number, so
c   we'll search for any redundant units that may be put into service.
```

```
      if(iop(i,j) .eq. 0 .and. ifailu(i,j) .eq. 0 .and.
     $ irep(i,j) .eq. 0)goto 103
c   this is a redundant unit that can be put in operation.
      goto 300
 103  nop(i)=nop(i)+1
      numr(i)=numr(i)-1
c   update the number of operating units, decrement the number
c   of failed units.
      iop(i,j)=1
c   update the operating flag of this unit.
      if(nop(i) .eq. numb(i))goto 199
c   we have found the adequate number of units to put in operation,
c   so abandon the search now.
 300  continue
      if(nop(i) .ge. (numb(i)-mofn(i)))goto 301
c   if this condition is not satisfied, the subsystem i does not have
c   the minimum number of units necessary for operation, so it's failed.
      ifail(i)=1
      numr(i)=0
      clk(i,1)=clk(i,1)-delt
      clk(i,2)=clk(i,2)+delt
      clk(i,3)=clk(i,3)+1
c   set the fail flag for subsystem i, reset the number of redundant
c   units, adjust the uptime, increment the downtime and increment
c   the number of failures of subsystem i.
      goto 1099
 301  ifail(i)=0
c   adequate number of units are operational such that the subsystem
c   is not in the failed state; reset the subsystem fail flag.
      goto 100
 2    goto 199
c   exactly the right number of units (as designed) are operational,
c   so no adjustments are necessary.
 3    mcont=nop(i)-numb(i)
c   more than the design number of units are operational, so
c   we'll have to put some of them in the redundant category.
      ifail(i)=0
      do 400  j=1,num2
      if(iop(i,j) .eq. 0 .or. ifailu(i,j) .eq. 1 .or.
     $ irep(i,j) .eq. 1)goto 400
c   the units with these flag values are not the ones contributing
c   to the surplus of the operating units.
      iop(i,j)=0
      mcont=mcont-1
      numr(i)=numr(i)+1
c   for the operating units (provided the surplus still exists, i.e.
c   mcont > 0 ):  reset the operating status flag, update the number of
c   spares (i.e. put the unit in the category of passively redundant units,
c   or spares) and decrement the number of superfluous operating units.
      if(mcont .eq. 0)goto 199
c   if satisfied, no surplus exists anymore.
 400  continue
 199  redun(i,2)=numr(i)+mofn(i)
```

```
c   adjust the total number of redundant units (number of spares +
c   number of actively redundant units).
        ifail(i)=0
c   reset the subsystem's fail flag.
        goto 100
 1099   redun(i,2)=0
 100    continue
        return
        end
        subroutine sched(iscflg)
c   subroutine sched determines if the system has entered a scheduled
c   maintenance period.
        common/msc/tsch
        common/mfsgrr/delt
        common/m2frss/it
        k=ifix(it*delt/8766)+1
c    it is the time step number, delt is the time step size,
c    and k is the year number that the system is in.
        test1=k*8766.-tsch
c   this is the time of the start of the annual scheduled
c   maintenance period.
        test2=k*8766.
c   this is the time of the end of the annual scheduled
c   maintenance period.
        time=it*delt
        iscflg=0
        if(test1 .le. time .and. time .le. test2)iscflg=1
c   if we are in between the start and the end of the scheduled
c   maintenance period, set the flag; the system is down for
c   scheduled maintenance.
        return
        end
        subroutine rely(i,j,r)
c   subroutine rely fetches a unit's failure rate and calculates
c   its time step reliability to be used in the Monte Carlo
c   simulation.
c   this subroutine is not needed now, but would be useful in
c   a situation where the failure rate changes with time and
c   withstood stress, e.g. in case of Weibul distribution of
c   failure rates (modeling burnout and infant mortality).
        parameter (ns=100,nu=20)
        dimension lamda(ns,nu),flrt(ns)
        common/mfsgrr/delt
        common/mrl/flrt
        real lamda
        lamda(i,j)=flrt(i)
        r=exp(-lamda(i,j)*delt)
        return
        end
        subroutine reptim(i,j,rt)
c   subroutine reptim fetches a unit's repair time.
c   this subroutine is not needed at this time, but
c   would be useful if modeling different failure
```

```
c   modes (requiring different repair times) of
c   a subsystem, or modeling of repair timelines.
        dimension mttr(50)
        common/mrt/mttr
        real mttr
        rt=mttr(i)
        return
        end
```