Final Report

RESEARCH
ON
FAILURE FREE SYSTEMS

Westinghouse Electric Corporation
Defense and Space Center
Surface Division

P.O. Box 1897 Baltimore, Maryland 21203

Final Report

RESEARCH ON
FAILURE FREE SYSTEMS

Contract No. NASw-572
Reference No. WGD-38521

December 1966

Westinghouse Electric Corporation
Defense and Space Center
Surface Division

P. O. Box 1897

Baltimore, Md. 21203
MDE 1425

PURPOSE

This final report is prepared in accordance with the requirements of contract NASw-572, modification No. 6, between the National Aeronautics and Space Administration and the Westinghouse Electric Corporation (reference WGD-38521). The general objective of this contract is the advancement of the state-of-the-art in the design of highly reliable electronic systems associated with the national space effort. The scope of this objective includes the development of techniques for constructing electronic systems which are invulnerable to the effects of relatively large numbers of internal component failures. The research reported herein has as its objective the development of techniques for efficiently allocating a limited number of test points within modularly redundant digital systems, and the estimation of system reliability based on the results obtained from limited testing.

TABLE OF CONTENTS

LIST OF ILLLUSTRATIONS

LIST OF TABLES

I. INTRODUCTION

The steadily increasing sophistication of space missions has been reflected in an increased complexity of spaceborne electronic data processing and control systems. This increase in complexity tends to lower the reliability of systems which normally operate in an environment where the cost of system failure is extremely high. In many cases, this cost may include the loss of human life in addition to the loss of a space vehicle and an aborted mission.

Several investigators have shown that the reliability of electronic systems can be greatly increased through the proper use of redundant equipment. By far the largest portion of the analytical work in this area has been concentrated on the development of synthesis techniques and the estimation of the initial reliability of redundant systems. Relatively little work has been done on the development of procedures for testing redundant systems and for estimating their reliability when one or more system components may be failed at the time of the estimation.

Pre-launch testing of spaceborne electronic systems is becoming more and more difficult as systems increase in complexity while decreasing in physical size. The testing problem will soon become worse as in-flight tests are used to determine the choice of alternative actions in deep space probes. The nature of redundant systems further complicates this problem because component failures which are undetectable by in-system operational exercises often increase the vulnerability of systems to future component failures. The combination of the above factors will tend to severely limit the usefulness of complex redundant spaceborne systems unless suitable methods of prelaunch and in-flight testing and test evaluation can be developed. The development of these methods should provide a sound basis upon which space vehicle launch or mid-course decisions can be made.

The user of a spaceborne electronic equipment has three different situations in which he may wish to test the equipment. The first situation exists when the equipment is being examined in a shop environment prior to being mounted in the space vehicle. In this situation, time is usually not of the essence and exhaustive testing is desirable to the limit permitted by the physical design of the equipment. The second situation exists when the equipment has been mounted in the vehicle, and a test is to be made just prior to launch. In this case, time is of the essence and does not permit an exhaustive test of an entire redundant data processing

system. The third situation exists only during long term, multi-phase space missions where a test is made near the end of a phase to determine which of the possible alternatives should be followed during the next phase. In most of the latter cases the decision made simply determines whether to continue or terminate the mission depending on the probability of successfully completing the next phase. In this case, both time and the complexity of the required test equipment are of vital interest.

In the latter two situations, there exists an obvious need for a technique to facilitate making an accurate estimate of the probability of successfully completing the mission based on information gained from testing only part of the system before or during the mission. Even in the shop environment, a similar need often exists because the use of tightly packaged microminiature circuitry may severely limit the amount of individual subsystem testing which can be performed. This is true regardless of the time permitted for the test or the availability of sophisticated test equipment.

The problem may be more precisely stated in the following manner. At some time, t_1, the user of a redundant digital system desires to estimate, as accurately as possible, the probability that a system will operate continuously until some later time, t_2. The user must make the estimate in some reasonably short period of time, using a limited amount of test equipment and a limited number of accessible test points. The general problem is to develop a test philosophy and a compatible statistical analysis procedure which will permit this user to confidently make decisions based on his estimate of the probability of successfully completing the mission. The accuracy of his estimate must reflect the cost associated with a wrong decision.

Within the general problem area, a natural dichotomy exists between the test design problem and the statistical estimation problem. Although these two problems are intimately related, they represent two separable points of emphasis, and they may be associated with slightly different short-range goals. The goal of a given test program is the development of a test procedure which will provide the most failure state information at a fixed cost or, conversely, a fixed amount of information of a minimal cost. In this context, time, number of system test points and test equipment all may be assumed to have an associated cost per unit. On the other hand, the goal of the statistical reliability estimation problem is the development of a technique for using the test results to provide the most accurate obtainable system reliability estimate. The recognition of this division is not meant to imply, however, that the two problems should be considered independently.

This report describes the results of a study program to develop a practical solution to this general problem. Section II describes a procedure for estimating the probability that a

redundant system will successfully complete a mission, using the information obtained from limited testing within the system. Section III describes a technique for optimally allocating the llmited number of test points within a redundant digital system.

Both the test point allocation technique and the compatible reliability analysis procedure have been programmed in the FORTRAN IV language. Descriptions of the two computer programs appear in appendices A and B. The programs have been documented in sufficient detail to enable a redundant system designer or user to perform an analysis of the system using the programs.

As an illustration of the applicability of the computer programs to practical systems, they have been applied to a specific redundant system design configuration-the modularly redundant Mariner C Spacecraft Sequencer. The details and results of this application are described in section IV.

A set of assumptions has been made to precisely define the problem boundaries. This set of assumptions was intended to focus the study effort on the members of a class of systems which are most likely to be used in the field in the relatively near future. These assumptions are enumerated below.

1. Only order-three redundant systems are considered.
2. Systems are so complex that exhaustive testing of each subsystem is not feasible at the time of interest.
3. The individual subsystems fail at a constant rate; hence, they have exponential reliability functions.
4. The design failure rates of system components are assumed to be true.
5. The tests can be made rapidly enough so that no failures will occur during the test period.
6. The system can be exercised sufficiently to assure that it is functionally operational at the time of test.

II. RELIABILITY ANALYSIS PROCEDURE

The exact reliability analysis of the redundant system is a difficult process which is not easily applied to the general system. There are two generalized reliability approximation techniques which have been considered as a base for a procedure to estimate, after limited testing, the probability of mission success of the general redundant system. The first technique is the minimal cuts analysis procedure, developed by Esary and Proschan. The second technique is the block model analysis procedure developed by D. K. Rubin, of JPL. Both of these methods produce a lower bound estimate of the true reliability of the general complex redundant system.

The definition of system reliability for both the minimal cuts technique and the block model procedure, is the probability that the system is successful at time t, given that all its components were operational at time zero and that there is no repair.

Both of the analysis procedures are based on the concept of coherent systems. This term refers to the effect of subsystem failures on the operation of the system. A coherent system is defined by four conditions:

1. If when a group of circuits in the system is failed causing the system to be failed, the occurrence of any additional failure or failures will not return the system to a successful condition;
2. If when a group of circuits in the system are successful and the system is successful, the system will not fail if some of the failed components are returned to the successful condition;
3. When all the circuits in the system are successful the system will be successful and,
4. When all the circuits in the system are failed the system is failed.

The assumption of coherence produces an inherent pessimism in both analysis procedures, even for simple series-connected systems.

The block model technique was chosen as the basis for the reliability analysis procedure in the present study. There are a number of advantages gained from the selection of this procedure over the minimal cuts technique.

Mr. Rubin's study has included a comparison of the two procedures. He has proved that for series-connected triple modularly redundant systems, the block model produces a greater lower-bound than does the minimal cuts. No exact error analysis has been performed to determine exactly the closeness of either of the two lower-bounds to the actual reliability of the general complex modularly redundant network. However, sample calculations for specific systems of some complexity have indicated that the block model is indeed superior to the minimal cuts model.

The block model procedure is inherently simpler in concept than the minimal cuts technique. Computer implementation of the block model is correspondingly simpler and produces a program which is shorter in both length and computer running time.

The reliability analysis program produced during the present study is required to be written in FORTRAN IV, to be highly machine-independent. The minimal cuts procedure has been previously programmed by Westinghouse, as a part of a larger computer program designed to synthesize modularly redundant systems. The program, written in the FAP language, is not documented, however. The amount of effort, therefore, to extract the minimal cuts protion of the FAP program and convert it to FORTRAN IV would be much greater than that necessary to program the block model procedure. The effort required to completely reprogram the minimal cuts procedure would also be greater than that needed to program the block model procedure. For these reasons, the block model has been chosen as the basis for the reliabillty analysis procedure.

The following section describes briefly the minimal cuts technique. This is followed by a discussion of the block model procedure used in the reliability analysis program.

A. MINIMAL CUTS TECHNIQUE

A cut is a set of circuits such that if they fail the system will have failed regardless of the condition of the other circuits in the system. The system may have a large number of cuts and a particular circuit may be in more than one of them. An example of a coherent system is shown in figure 1. As long as any path through successful circuits exists between the two terminals of the system, the system is said to be successful. A circuit failure opens the path between the two terminals of the circuit.

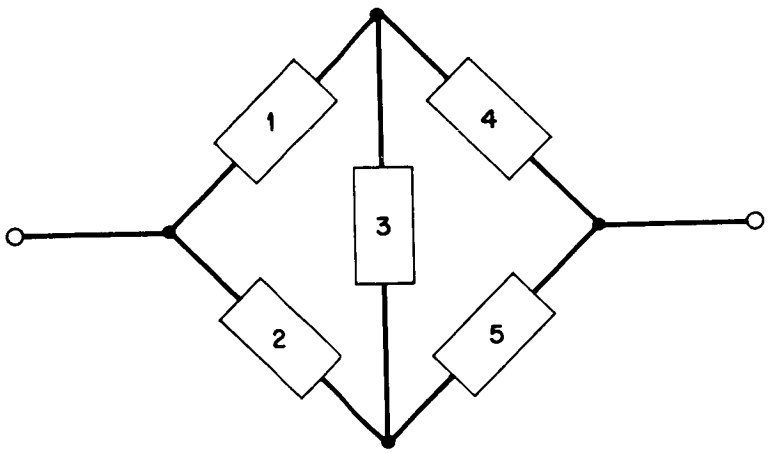


Figure 1. Example of a Coherent Network

Table 1. The Cuts of the Network in Figure 1.

Cut No.	Circuits in the Cut
1.	1, 2
2.	1, 2, 3
3.	1, 2, 4
4.	1, 2, 5
5.	1, 2, 3, 4
6.	1, 2, 3, 5
7.	1, 2, 4, 5
8.	1, 2, 3, 4, 5
9.	4, 5
10.	3, 4, 5
11.	2, 4, 5
12.	1, 4, 5
13.	2, 3, 4, 5
14.	1, 3, 4, 5
15.	1, 3, 5
16.	2, 3, 4

The cuts of this system are listed in table 1. The numbers of the circuits describe the cuts. The failure of any of the cuts will cause the network of figure 1 to fail.

A minimal cut is defined as a cut in which there is no subset of circuits whose failure alone will cause the system to fail. From table 1 the minimal cuts of the network in figure 1 can easily be recognized. They are listed in table 2 along with their probabilities of occurrence. Since the symbol p_i represents the probability of success, of the ith circuit, then $(1-p_i)$ is the probability that the ith circuit fails.

Table 2. The Minimal Cuts of the Network in Figure 1.

Min. Cut No.	Circuits in the Min. Cut	Probability of Failure of the Min. Cut
1.	1, 2	$(1-p_1)\ (1-p_2)$
2.	4, 5	$(1-p_4)\ (1-p_5)$
3.	1, 3, 5	$(1-p_1)\ (1-p_3)\ (1-p_5)$
4.	2, 3, 4	$(1-p_2)\ (1-p_3)\ (1-p_4)$

The lower bound approximation to reliability depends on the identification of all the minimal cuts in the network. Esary and Proschan found that a lower bound to system reliability is the probability that none of the system's minimal cuts fails. For the example, this lower bound, R_{LB} is:

$$R_{LB} = \left[1 - (1-p_1)\ (1-p_2)\right] \quad \left[1 - (1-p_4)\ (1-p_5)\right] \quad \left[1 - (1-p_1)\ (1-p_3)\ (1-p_5)\right] \left[1 - (1-p_2)\ (1-p_3)\ (1-p_4)\right] \tag{II-1}$$

This relationship can be written for general systems if the jth minimal cut is denoted by the set S_j. The members of the jth minimal cut are given by $i \in S_j$. The probability of failure of the jth minimal cut is:

$$\prod_{i \in S_j} (1-p_i) \tag{II-2}$$

The lower bound to the system reliability is the probability that none of the system's minimal cuts fail or:

$$R_{LB} = \prod_{\text{all}j} \left[1 - \prod_{i \in S_j} (1-p_i)\right] \tag{II-3}$$

The minimal cuts of a multiple-line redundant network have three characteristics which are sufficient to establish their identity. These are listed below. For these characteristics, m denotes the order of redundancy and specifies the minimum number of correct circuits required in each stage.

1. All the members of the minimal cut are circuits in a restored function or in the functions or restorers that are the error-linked sources of that restored function.

2. The failure of each member of the minimal cut will cause one output line of the restored function to be in error, and each member will be in a different position, or rank.

3. The failure of a minimal cut will cause exactly m-k+1 output lines of the restored function to be in error, hence a minimal cut will have m-k+1 members.

For an order-three redundant system, these characteristics may be used to construct an expression for the minimal cuts lower bound on system reliability. For an order-three network, the minimal cut will consist of 2 subsystems, each in a different position and each either in the restored function or in its error-linked sources. There are two kinds of minimal cuts in an order-three network. The first kind includes those cuts in which both subsystems are in the same function or restorer. Every function or restorer in the network will contribute this type of cut. The probability of failure of a cut of this nature in function $\underline{x}$ is:

$$(1-p_x)^2 \tag{II-4}$$

There are three ways to choose the subsystems which are failed in a particular function or restorer. Then each will contribute three minimal cuts. The probability that failure does not occur because of failure of minimal cuts of the first kind is:

$$\prod_{\text{all } x} \left[1 - (\quad d_x \quad) (1-p_x)^2 \right]^3 \tag{II-5}$$

The symbol d_x represents a Boolean function which is 1 if stage $\underline{x}$ is present in the system, and Oif stage $\underline{x}$ is not present.

The second kind of minimal cut includes two subsystems, one in each of two function or restorer stages. Two subsystems are in the same minimal cut only if they are both error-linked sources of the same restored function. The probability of a minimal cut of the second kind, one in function $\underline{x}$ and one in function $\underline{y}$ is:

$$(1-p_x)\ (1-p_y). \tag{II-6}$$

Minimal cuts of the second kind are present in functions $\underline{x}$ and $\underline{y}$ only if the Boolean function

$$\left[d_{ix}\ d_{iy} \right]_{x \neq y} = 1 \tag{II-7}$$

is satisfied. Equation II-7 says that both $\underline{x}$ and $\underline{y}$ are error-linked sources of the same restored function, $\underline{i}$.

Then the probability that a cut of the second kind does not fail in function $\underline{x}$ and $\underline{y}$ is:

$$1-(d_{ix}\ d_{iy})_{x\neq y}\ (1-p_x)\ (1-p_y) \qquad \text{(II-8)}$$

Subsystems in the same rank cannot be in the same minimal cut. There are three ranks from which the first member of the minimal cut may be chosen from function $\underline{x}$ but once this choice is made, there are only two ranks in function $\underline{y}$ from which the second member of the cut may be chosen. There are then 6 minimal cuts which include one subsystem in function $\underline{x}$ and one subsystem in function $\underline{y}$.

With this information, the probability that a minimal cut of the second kind does not fail in a system can be written:

$$\prod_{\substack{\text{all } x\neq y \\ \text{all } i}} \left[1 - (d_{ix}\ d_{iy})_{x\neq y}\ (1-p_x)\ (1-p_y)\right]^6 \qquad \text{(II-9)}$$

Now with the results of (II-5) and (II-9) the total minimal cut reliability expression for an order-three system is written.

$$R_{LB} = \left\{\prod_{\text{all } x}\left[1 - (d_x)\ (1-p_x)^2\right]\right\}^3 \left\{\prod_{\substack{\text{all } x\neq y \\ \text{all } i}}\left[1-(d_{ix}\ d_{iy})_{x\neq y}\ (1-p_x)(1-p_y)\right]\right\}^6 \qquad \text{(II-10)}$$

This equation is valid for a system in which all subsystems in all stages have the same reliability. For the general case in which the subsystem reliabilities may differ, the cubic and sixth power terms in (II-10) must be expanded.

The minimal cuts technique is conceptually more complex than the block model procedure. In addition, the computer implementation of the minimal cuts is both more lengthy and slower than the block model implementation.

B. BLOCK MODEL ANALYSIS METHOD

The procedure for estimating the reliability of partially tested redundant systems used in this study is based on the block model analysis technique, developed by D. K. Rubin, of J. P. L. In the block model procedure, it is assumed that the failure of majority of replicas of a subsystem will cause a system failure. It is further assumed that the analysis is made at time zero, and that all units, or subsystems, are operating correctly. For the present study, these latter two assumptions have been relaxed and the procedure correspondingly modified to include information obtained from system test data. The following paragraphs describe

first the basic block analysis procedure. This is followed by a section explaining the modified procedure used in the computer program implementation.

The block model analysis procedure yields a lower-bound estimate of the reliability of triple-modularly redundant systems of any arbitrary configuration. The procedure begins by grouping together subsystems, or stages, each of whose unit failures can combine with unit failures of another stage to cause system failure. Units in each group are isolated from those in other groups. For example, figure 2 shows, in block diagram form,

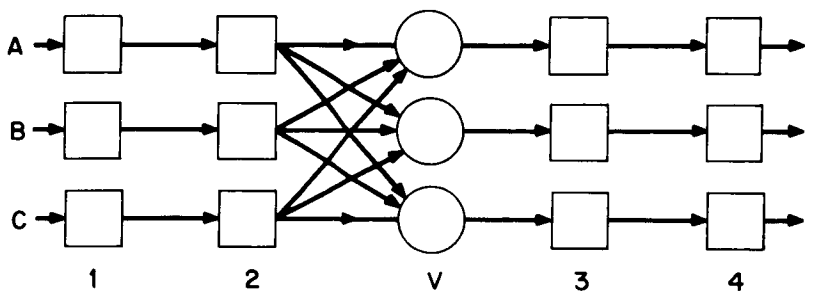


Figure 2. Portion of a Series-connected System

a portion of a series-connected, triple modularly redundant system. The boxes represent subsystems, or units, and the circles represent restorers, or voters. The failure of a unit in stage 1 can combine with a failure of a unit in stage 2 to produce system failure. Similarly, failure of units in stages 3 and 4 can combine with those of the voter stage, V, or with each other's failures, to produce system failure. The failure of a unit in stage 1 or 2, however, cannot combine with the failure of a unit in stages 3, 4, or V to produce system failure. There are therefore two groups, or blocks which can be formed: group 1, composed of stages 1 and 2, and group 2, composed of stages 3, 4, and V. Since the failure of a unit in one group cannot combine with the failure of a unit in another group to cause system failure, the groups are independent, and the reliability of each group may be computed independently of the other groups. The resultant reliabilities may then be multiplied to produce the system reliability, according to the series-chain rule.

Figure 3 shows the reliability model for the system of figure 2. If it is assumed that the subsystem units

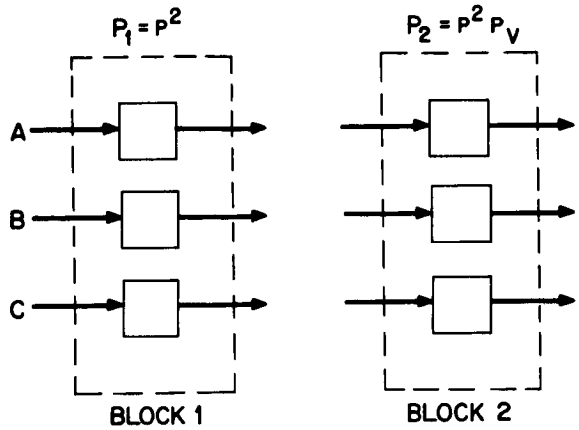


Figure 3. Block Reliability Model of the System in figure 2.

in figure 2 have equal reliabilities, p, and that the voter units have equal reliabilities, p_v, then the reliabilities of the units in block 1 are p^2. By the same argument, the reliability of a unit in block 2 is $p^2 p_v$. The reliability of each block may be computed by the formula $p_B = 3p^2 - 2p^3$, where p is the block unit reliability. The system reliability estimate is then the product of the p_B's:

$$R = \prod_{\text{all } i} P_{B_i} \qquad \text{(II-11)}$$

The rules described above for forming the blocks of a given system can be shown to reduce to one rule for series-connected systems: namely, that the units of a block consist of all of the units providing direct or indirect inputs to a given restorer, and which are also failure-linked to the restored stage. It is assumed that every system output has a restorer. For simple, series-connected systems, all of the blocks defined by this method will be failure-independent from one another, and the resulting system reliability estimate is the exact system reliability. For complex systems involving feedback, feedforward, and multiple fan-out, however, the blocks will not actually be independent. The system reliability estimate, therefore, will not be the actual reliability, but will be a lower-bound estimate.

As an example, figure 4 shows one such complex system. The rectangles in the figure symbolize triplicated stages in the triple modularly redundant system, and the ovals indicate the triplicated majority voters. The blocks for this system can be formed in accordance with the rule above. For example, the output of function 3 has three error-linked sources:

function 1, function 2, and of course, function 3. These three stages therefore form one block of the system. A second block is composed of the error-linked sources of function 5. They are: function 5, voter 3, function 2, function 4, function1, and because of the feedback loop, function 6 and voter 5. The remaining three blocks are formed in the same manner.

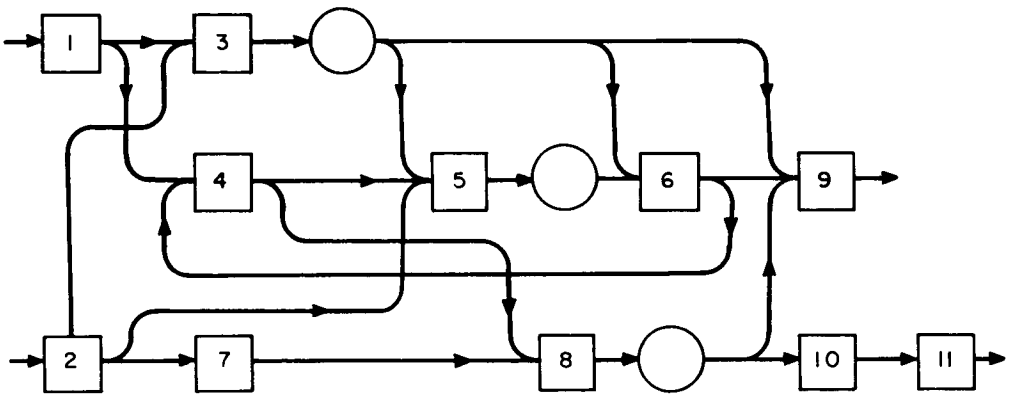


Figure 4. Majority-voted Triple-Modularly Redundant System.

The five blocks for the system are listed in table 3. The numbers in the table refer to the stage numbers in figure 4. The lettered numbers, such as V3 and V5, refer to the voters following the stages of the same numbers.

Table 3. Blocks for the System in Figure 4.

Block 1	Block 2	Block 3	Block 4	Block 5
1	1	1	6	10
2	2	2	9	11
3	4	4	V3	V8
	5	6	V5	
	6	7	V8	
	V3	8		
	V5	V3		
		V5		

It is obvious that the resulting blocks are not failure-independent. Many of the stages appear on two or more blocks. Thus the reliability estimate is a lower-bound approximation to the actual system reliability.

C. MODIFIED BLOCK ANALYSIS METHOD

The block model analysis technique has been modified to include information obtained from partial testing. The procedures for forming blocks and combining block reliabilities remain exactly the same. The individual unit reliabilities, however, must be altered to reflect test data showing them to be failed or working at the time of the test. The revised unit reliability is the product of two probabilities: (1) The probability that the unit is working at the time of the test, and (2) the conditional probability that the unit will be working at the end of the mission, given that it was working at test time.

That is,

$$r_i = \left[P_i\left(W(t_t)\right)\right] \times \left[P_i(W(t_m) / W(t_t))\right] , \qquad \text{(II-12)}$$

where $P_i(W(t_t))$ is the probability that subsystem $\underline{i}$ is operational at test time, t_t;

and $P_i(W(t_m) / W(t_t))$ is the conditional probability that subsystem $\underline{i}$ will operate continuously until mission end, t_m, given that it was operational at t_t.

The first probability may be changed to reflect operational information obtained from test data, whereas the second probability remains unchanged.

For this discussion, a single "test point" tests all three ranks at its location. The information obtained from a test point at a given location shows which of the three ranks is failed, if a failure exists at the location. A more comprehensive discussion of the test point is included later, in section III-A.

The method of computing a revised estimate of a unit's reliability can be most easily shown by a few brief examples.

Example 1. A segment of a typical triple modularly redundant system is shown in figure 5. A test point is located after stage 3, as indicated by the triangle. If the test data indicates that all three ranks are operating, the probabilities that the units are working at test time is 1. 0. Therefore, the new unit reliability will be $e^{-\lambda_i (t_2 - t_1)}$, where λ_i is the failure rate of unit i, t_2 is the mission end time, and t_1 is the time of test. The revised block reliability is now

$$P\,(t_m - t_t) \,/\, P\,(t_t - t_o), \qquad \text{(II-13)}$$

where $P(t_m - t_t)$ is the probability that at least two of the three ranks succeed from test time to mission end.

and $P(t_t - t_o)$ is the probability that the block is operating at test time. This latter factor is a result of the basic assumption that the system is functionally operational at test time.

For this example, the latter factor is, of course, unity, since the block was tested.

Example 2: See Figure 6 on the following page.

The assumptions are now made that test points are located after stages 2 and 3, and that the test points indicate that rank A of stage 2 has failed, while all other ranks are operational. All units except unit 2A, therefore, are working, and have a new probability of working at the time of test. This probability is unity (1). The corresponding probability for unit 2A is, of course, zero.

Example 3: See Figure 7 on the following page.

For this example, the existence of only one test point, showing all ranks of stage 2 working, is assumed.

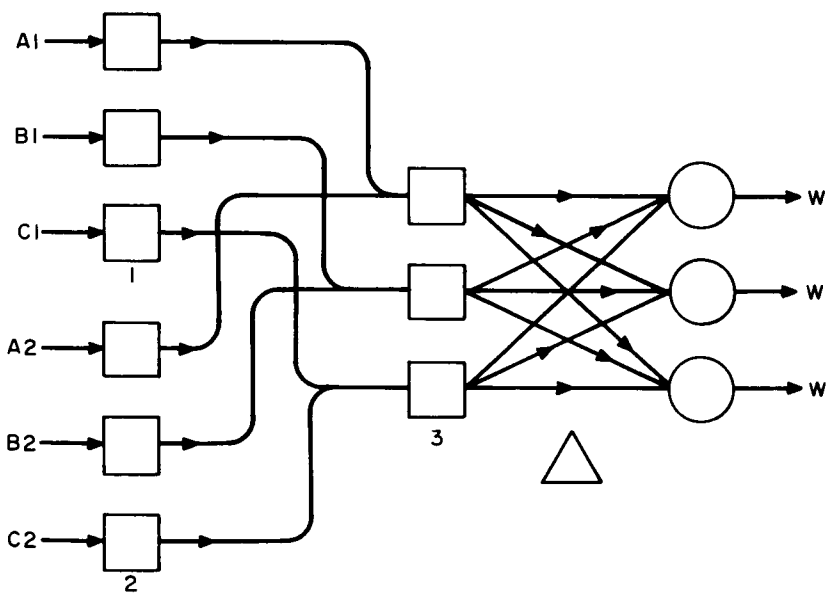


Figure 5. All three ranks observed as operational.

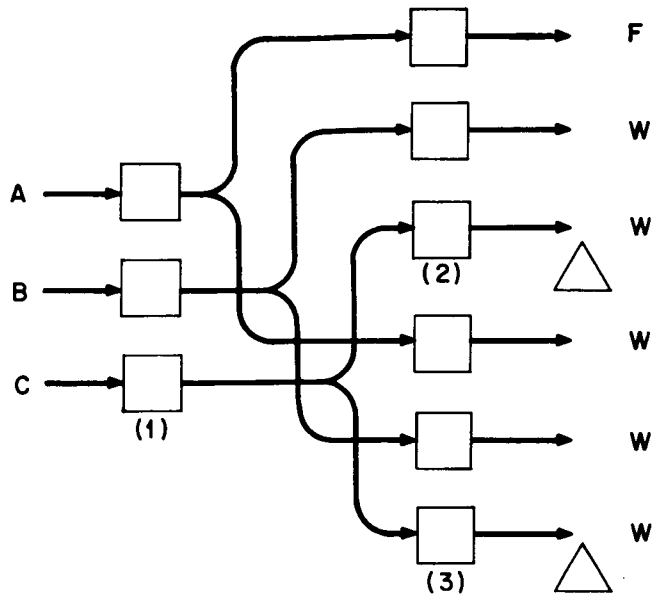


Figure 6. A failure observed in one rank.

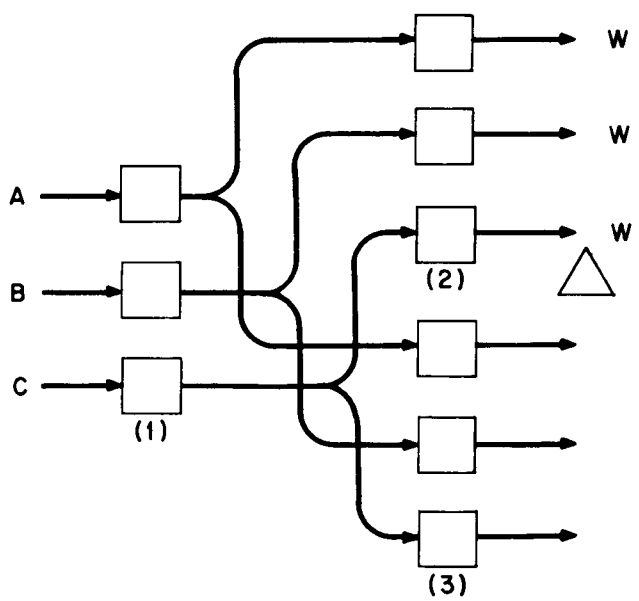


Figure 7. A tested block and an untested block with a common stage.

As a result, all units in stages 1 and 2 have a probability of operating at test, of unity (1). Nothing is known about the state of the units in stage 3, however, so their probabilities are unchanged. Since the block consisting of stages 1 and 3 is untested, the block's reliability is computed with the constraints that units 1A, 1B, and 1C are working, and that at least 2 of the ranks in the block are working at the time of the test.

Example 4: There is one additional test situation which must be handled in a manner differing from that of the three examples above. Assume that the test point in figure 7 does not show all three ranks working, but instead indicates an error at the output of rank A. In this case, the probability that subsystem A in stage 1 is operational at test time cannot be set to zero. The error in rank A of the tested block may have had no effect on the output of the untested stage. Subsystem 1A may be operating, while subsystem 2A is failed. The probability that 2A is operational must therefore be calculated.

Assume that p_1 is the a priori (without test data) probability that a subsystem in stage 1 is operational at test time, and p_2 is the a prior probability that a unit in stage 2 is operational Then the probability that at least one subsystem in rank A of the tested block is failed is given by

$$p_{F(A)} = 1 - p_1 p_2. \tag{II-14}$$

The probability that subsystem 1A is working is then given by the equation

$$p_1^1 = 1 - \left(\frac{1-p_1}{1-p_1 p_2} \right) \tag{II-15}$$

In general, the probability that a subsystem common to both a tested-failed rank and an untested block is operational at test time is

$$p_c^1 = \left(\frac{1 - p_c}{1 - \prod_{i=1}^{n} p_i} \right) \tag{II-16}$$

where p_c^1 is the revised probability that the common subsystem is operational at test time.

p_c is the a priori probability that the common subsystem is operational at test time

p_i is the a priori probability that subsystem $\underline{i}$ in the tested-failed rank is operational at test time

and n is the total number of subsystems in the tested-failed rank

The revised probability p_c^1 is used in the calculation of the reliability estimate of the untested block.

The foregoing examples illustrate the main types of modifications to be made to the block analysis procedure. In tested blocks, the probabilities that individual units are working at test are changed according to the test data. In untested blocks, the probabilities of all untested units remain unchanged. The reliabilities of all blocks are changed, however, by computing them according to the constraint that at least 2 ranks are operating, since the system is operating.

1. Test Data Effects

As an illustration of the effect of test data on the reliability estimate of a redundant system, a number of computer runs have been made with the reliability analysis program.

Consider the simple system in figure 8. Each box represents a triple modularly redundant stage of the system, and each circle represents a set of

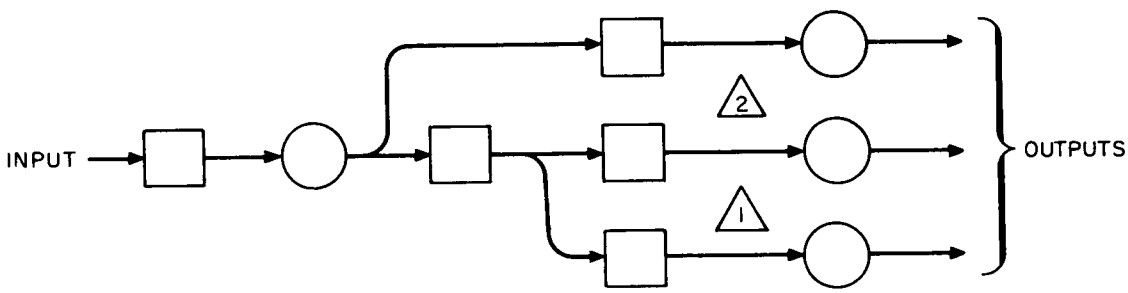


Figure 8. Example S stem with Two Test Points.

three voters. There are two test points, at the locations indicated by the numbered triangles. All units are assumed to have equal failure rates. Test point 1 is assumed to have indicated a failure in one rank, and test point 2 indicates no failures. Table 4 below shows the resulting system reliability estimates for several different test times. The initial system reliability estimate, assuming that all units are operational at time zero, is 0. 992469. The total mission time is TM.

Table 4. Reliability Estimates for System in Figure 8.

Failure Observed at Test Point One	
Test Time	Probability of Success
TM/4	0. 932153
TM/2	0. 954256
3TM/4	0. 976869

Table 5 shows a number of reliability estimates for the same system, with the same test points and initial reliability estimate. In this case, however, no failures are observed at the test points.

Table 5. Reliability Estimates for System in Figure 8.

No Failures Observed	
Test Time	Reliability
TM/4	0.994953
TM/2	0.997052
3TM/4	0.998742

As expected, the reliability estimate increases as the test is performed later into the mission. The reliability estimates for the second case, with no observed failures, are significantly higher than those of the first case.

D. COMPUTER PROGRAM CAPABILITY

The reliability analysis program computes an estimate of system reliability, using the modified block model method. The program first constructs the lists of blocks for the system, and then modifies subsystem reliabilities in accordance with the test data. The reliability estimate for each of the blocks is then calculated, and the system reliability estimate is the series product of the block reliability estimates. As stated earlier, the block reliability estimates are calculated with the constraint that the system is functionally operational at the time the test and estimation are performed.

The program has the capability to estimate the reliability after test for virtually any triple-modularly redundant digital system. Any degree of system complexity will be handled by the program, including feedback, feedforward, and multiple fan-in and fan-out. Every stage in the system, including restorer stages, must be order-three redundant.

The program has, in addition, an option to calculate an estimate of the initial reliability of the system; i. e. the reliability at time zero, assuming that all system components are operational. This estimate is computed by the unmodified block model method. The initial reliability estimate can be used as a comparison with the estimate based on the test data.

The reliability analysis program is written in FORTRAN IV, making it highly machine-independent. The size of the system to be analyzed will be limited only by individual computer storage limitations and allowable running time.

III. TEST POINT ALLOCATION PROCEDURE

A. TEST POINT DEFINITION AND IMPLEMENTATION

The general term "test point" may be defined in anyone of a number of specific ways. For example, the term could refer to all of the circuitry required to verify the operation of a particular subsystem within a redundant system, or the operation of one rank of a system. In either of these cases, the test circuitry would in general be sufficiently complex to produce a significant effect on the reliability of the system and on the confidence factor associated with any test results. Similar, although less detrimental, effects would be noted in the case of a "test point" defined as all of the equipment necessary to compare the operation of two of the subsystem or signal replicas at a given stage and subsequently determine the operational status of both replicas.

For the purposes of this study, the term "test point" can have either one of two closely related meanings. It may mean a set of simple contact points at which triplicated subsystem outputs can be sampled, or it may mean a set of circuits used to detect differences between three nominally identical output signals. In the first case, simple difference detector circuits are fabricated as part of the external test equipment while in the second case these circuits are built into the operational equipment.

The important common factors about the two types of test points are that the same test information is ultimately produced by either type, and the design of all test points of one type is the same. The information produced is, of course, the exact rank, if any, which is in error. Assuming that in any particular system, only one type of test point will be used, any size, weight, power consumption or initial cost constraints can be immediately converted to a single maximum number of test points constraint. To this analytical advantage of having only one constraint, is added the advantage of requiring extremely simple test circuitry of a type having little or no effect on the basic system design, the system reliability, or on the confidence factor associated with the test results. For the purposes of this study, these latter factors, therefore, are assumed to be negligible.

As often happens, the price of test point simplicity is paid for by a reduction in information obtainable relative to a more complicated test mechanism. In this case, the information that is lost is that which shows the exact subsystem location of a failure which causes any observed signal error. Lacking this information, the observer may have difficulty

in assessing the true implications of an observed error relative to the status of the overall system. This difficulty is illustrated by the following example.

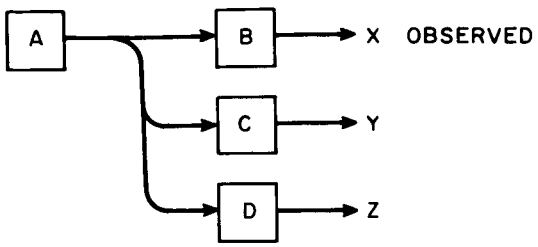


Figure 9. A Segment of One Rank of a Redundant Digital System.

Referring to Figure 9, if an error observed at point X is the result of a failure in subsystem B, the reliability of the rest of the system is significantly greater than if the failure is in subsystem A. Interestingly enough, a double failure, i. e., failures in both A and B, has exactly the same effect on system reliability as a failure in A but not B.

The reliability analysis problem is further complicated if multiple errors are observed along interconnecting strings. For example, if errors are observed at both X and Y, the observer still does not know if A alone has failed, or if B and C have failed. Again the reliability estimate for the rest of the system varies, depending on which of the possible failure patterns exists. The combined failures of A, B and C are equivalent to a single failure in A alone.

Section II above describes the procedure used in estimating the effect on the reliability caused by observed signal errors.

B. TEST POINT VALUE CRITERIA

Any attempt to develop a procedure for allocating test points within a redundant system must be preceded by the development of some function defining the "value" of a test point. That is, what mathematical criterion should be used in deciding the optimum placement of each succeeding test point added to a system. The most desirable criterion would be one which defines the most "valuable" test point location as that which will allow a test point to supply information which will have the greatest effect on the system reliability estimate. This effect would be measured by the change in the system reliability estimate relative to an estimate made without the information of a test point at that location or competing test point locations.

1. Criterion Based on the Expected Change in the Reliability Estimate.

Consider a test point value criterion based on direct measurement of the expected change in the estimate of reliability. Assume that we wish to determine the value of placing a test point at the output of the block or stage shown in figure 10. We must first measure

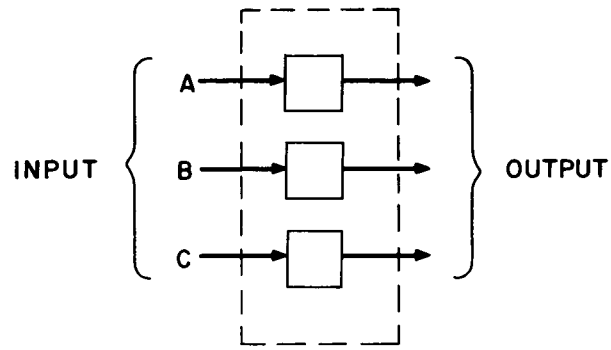


Figure 10. Potential Test Point Location

the expected reliability without a test point. Assume that the subsystems in the block have equal failure rates. The probability that the block is functionally operational at test time is given by:

$$R(t_1) = 3\left[p\ (t_1-t_o)\right]^2 - 2\left[p\ (t_1-t_o)\right]^3, \qquad \text{(III-1)}$$

where $p\ (t_1-t_o)$ is the conditional probability that a subsystem will be operating at test time, t_1, given that it was operational at t_o.

The probability that the block is operational at the end of the mission, t_m, is given similarly by the formula:

$$R(t_m) = 3\left[p\ (t_m-t_o)\right]^2 - 2\left[p\ (t_m-t_o)\right]^3. \qquad \text{(III-2)}$$

Now, one of the basic assumptions of the present study is that the system is functionally opeartional at the time of test, t_1. Therefore, it is known that every block in the system is functionally operational; i. e., that at least two of the three subsystems in every stage are working at test time. The expected reliability of the block in figure III-2 can then be computed from equations (III-1) and (III-2). It is:

$$\overline{R} = \frac{3\left[p(t_m-t_o)\right]^2 - 2\left[p(t_m-t_o)\right]^3}{3\left[p(t_1-t_o)\right]^2 - 2\left[p(t_1-t_o)\right]^3} \qquad \text{(III-3)}$$

$\overline{R}$ is the conditional probability that the block will operate continuously until t_m, given that it was functionally operational at t_1.

Let us now define a test point value function, the purpose of which is to measure the expected change in the reliability estimate, caused by the addition of a test point at a given location in a system. The value function is

$$V \equiv \left| \left(\sum_{\text{all } i} P_i R_i \right) - \overline{R} \right| . \qquad \text{(III-4)}$$

In this equation,

P_i = the conditional probability that a particular failure pattern, i, will be observed at the location, given that the system is working.

R_i = the reliability estimate of the block, given that failure pattern i is observed.

$\overline{R}$ = the expected reliability estimate without the information of the test point.

Since the block is assumed to be functionally operational at test time, there are two possible observed failure patterns at a test point: (1) there is a failure in one rank of the block, or (2) there are no failures; all ranks are operational. The probability that failure pattern (1) will be observed is given by the following equation:

$$P_1 = \frac{3 \left[P(t_1 - t_o) \right]^2 \left[1 - P(t_1 - t_o) \right]}{3 \left[P(t_1 - t_o) \right]^2 - 2 \left[P(t_1 - t_o) \right]^3} . \qquad \text{(III-5)}$$

The probability that failure pattern (2) will be observed is:

$$P_2 = \frac{\left[P(t_1 - t_o) \right]^3}{3 \left[P(t_1 - t_o) \right]^2 - 2 \left[P(t_1 - t_o) \right]^3} . \qquad \text{(III-6)}$$

If failure pattern (1) is observed, the reliability estimate of the block is

$$R_1 = \left[P(t_m - t_1) \right]^2 \qquad \text{(III-7)}$$

Similarly, the reliability estimate of the block when failure pattern (2) is observed, is

$$R_2 = \left(\left[P(t_m - t_1) \right]^2 \right) \left(3 - 2 \left[P(t_m - t_1) \right] \right) . \qquad \text{(III-8)}$$

For a given test point, then, the quantity $\sum_{\text{all } i} P_i R_i$, the expected reliability estimate with test information, becomes

$$\sum_{\text{all } i} P_i R_i = P_1 R_1 + P_2 R_2 . \qquad \text{(III-9)}$$

When equations (III-5) through (III-8) are substituted in equation (III-9), the result is

$$\sum_{\text{all } i} P_i R_i = \frac{3\left[P(t_1-t_o)\right]^2\left(1-\left[P(t_1-t_o)\right]\right)\left[P(t_m-t_1)\right]^2+\left[P(t_1-t_o)\right]^3\left[P(t_m-t_1)\right]^2\left(\left[3-2\ P(t_m-t_1)\right]\right)}{3\left[P(t_1-t_o)\right]^2 \quad -2\left[P(t_1-t_o)\right]^3} \qquad \text{(III-10)}$$

which reduces to

$$\sum_{\text{all } i} P_i R_i = \frac{3\left[P(t_1-t_o)\right]^2\left[P(t_m-t_1)\right]^2 - 2\left[P(t_1-t_o)\right]^3\left[P(t_m-t_1)\right]^3}{3\left[P(t_1-t_o)\right]^2 - 2\left[P(t_1-t_o)\right]^3}. \qquad \text{(III-11)}$$

However, the product $P(t_1-t_o) \cdot P(t_m-t_1)$ is equal to the probability $P(t_m-t_o)$, since all subsystem P's are assumed to be exponentials. Therefore, equation (III-11) can be reduced further, to

$$\sum_{\text{all } i} P_i R_i = \frac{3\left[P(t_m-t_o)\right]^2 - 2\left[P(t_m-t_o)\right]^3}{3\left[P(t_1-t_o)\right]^2 - 2\left[P(t_1-t_o)\right]^3}. \qquad \text{(III-12)}$$

Equation (III-12) is the expected reliability estimate of the block with a test point. It can be seen, however, that this equation is identically equal to equation (III-3), the expected reliability estimate without the test point. When these equations are substituted into the value function equation (III-4), the result is zero:

$$V = \left| \left(\sum_{\text{all } i} P_i R_i \right) - \bar{R} \right| = \left| \bar{R} - \bar{R} \right| \equiv 0. \qquad \text{(III-13)}$$

The expected reliability estimate with test data is identically equal to that without test data. Of course, a given set of test data will produce, in general, a change in the reliability estimate. But a value function to measure the a priori change will always result in zero value for all test points. Because of this basic difficulty, a different approach to the definition of a test point value function has been taken. This approach is based on one of the basic principles of information theory.

2. Information Theory Criterion.

It is known that in a situation in which there is a binary (two possible outcomes) result, the average amount of information obtained is a function of the results' probabilities of occurrence. This information is a maximum when the two results are equally probable. The result obtained from a test point in a redundant digital system is such a binary situation. The test point can observe either one failure or no failures. Because any redundant systems under consideration are expected to be operating in a high reliability (above 0.95) region, the

amount of information obtained from a test point can be maximized by maximizing the probability of observing a failure at the potential test point location. This is the location most nearly equalizing the probability of observing either the failed or working state. It can be concluded, therefore, that each succeeding test point added to the system should be located at the point in the system at which the probability of observing an erroneous signal is at a maximum.

It is on the basis of this conclusion that the test point allocation computer program has been developed. The value of placing a test point at a potential location is defined as the conditional probability that a failure will be observed, given that the system is functionally operational. The computer program has been constructed to handle low reliability systems, where the probabilities that a failure will be observed at a given location may be greater than 0.5. In such systems the optimum placement will not, in general, be at the point of maximum probability of failure, but rather at the point at which the probability is closest to 0.5. For this reason, the "value" which program actually computes is the difference between the probability of failure and 0.5. For high reliability systems, however, the effective value criterion defines the most valuable test point location as the point with maximum probability of failure.

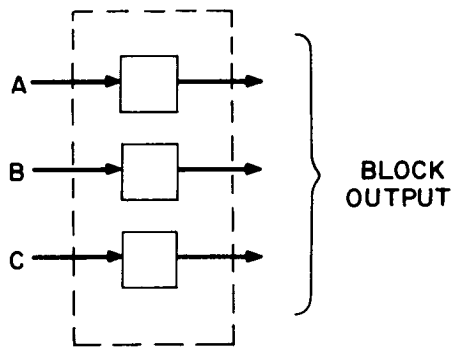


Figure 11. Block Output, Potential Test Point Location

Assume that a potential test point location is the output of the block shown in figure 11. The probability that a subsystem is operational is $P(t_1\text{-}t_o)$; the subsystems in all three ranks have equal failure rates. The value of placing a test point at the block's output is therefore

$$V = 1 - \frac{\left[P(t_1\text{-}t_o)\right]^3}{3\left[P(t_1\text{-}t_o)\right]^2 - 2\left[P(t_1\text{-}t_o)\right]^3} \qquad \text{(III-14)}$$

If we assume that the subsystems have unequal failure rates, this value function equation must be expanded. The resulting function is:

$$V = 1 - \frac{\left[P_a(t_1-t_o)\right]\left[P_b(t_1-t_o)\right]\left[P_c(t_1-t_o)\right]}{\left[P_a(t_1-t_o)\right]\left[P_b(t_1-t_o)\right] + \left[P_a(t_1-t_o)\right]\left[P_c(t_1-t_o)\right] + \left[P_b(t_1-t_o)\right]\left[P_c(t_1-t_o)\right] - 2\left[P_a(t_1-t_o)\right]\left[P_b(t_1-t_o)\right]\left[P_c(t_1-t_o)\right]} \quad \text{(III-15)}$$

The potential test point locations are defined as the outputs of the blocks in the block reliability model. Each test point, therefore, could be located at either the output of a restored function or a system output. It can be seen that at this location in a given block, the total number of subsystems tested is greater than at any other location in the block.

This test point allocation procedure follows the techniques used in the block reliability model, i. e. the blocks are considered to be failure-independent. The placement of each succeeding test point therefore is not dependent on the locations of the previously allocated points.

a) Example Allocation.

A typical system to which the test point allocation procedure could be applied is shown in figure 12. There are five possible test point locations: the outputs of stages 3, 5, and 8 and at the two system outputs, stages 9 and 11. The allocation procedure was applied to this system, using the computer

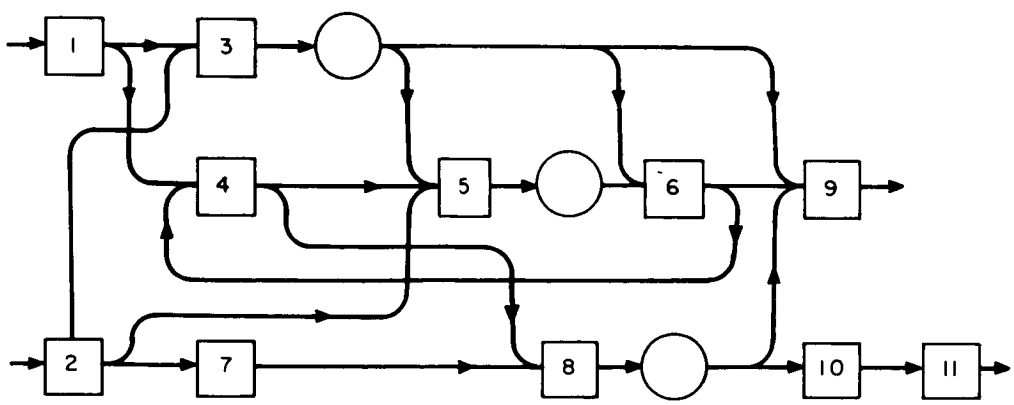


Figure 12. Majority-voted Triple-modularly Redundant System.

program. All units were given equal failure rates of 0.2106×10^{-4} per unit time. The test time was set at 50 time units. The resulting allocations, with their corresponding "values" are shown in table 6.

Table 6. Test Points Allocated to the System in Figure 12.

TEST PT NUMBER	VALUE	LOCATE AT STAGE
1	0.24751×10^{-1}	8
2	0.21713×10^{-1}	5
3	0.15590×10^{-1}	9
4	0.94028×10^{-2}	11
5	0.94028×10^{-2}	3

C. COMPUTER PROGRAM CAPABILITY

The test point allocation program allocates a limited number of test points within a triple-modularly redundant digital system. The program calculates the value of placing a test point at each of the potential test point locations. Since the potential test point locations consist of the outputs of all of the blocks in the block model, the program calculates the probability that an erroneous signal will be observed at each of the block outputs.

The result of the program is an ordered list of the potential test point locations, listed in order of decreasing value. The user may then efficiently allocate a limited number (N) of test points within the system by placing them at the first N locations in the list.

The computer program will handle virtually any triple-modularly redundant digital system. All of the stages in the system, including voter stages, must be order-three redundant. Virtually any degree of system complexity can be accomodated by the program, including feedback, feedforward, and multiple fan-in and fan-out.

The test point allocation program is written in FORTRAN IV, making it highly machine-independent. The size and complexity of the system to which the program allocates test points is limited only by individual computer storage and running time limitations.

IV. PROGRAMS APPLIED TO THE MARINER C SEQUENCER

In order to demonstrate the applicability of both computer programs to practicable systems, they have been applied to a specific system design configuration. The system used is the Mariner C Redundant Spacecraft Sequencer, previously developed for the Jet Propulsion Laboratories. *

The first step in the application of the programs to a specific system is the conversion of the system to block diagram form and the subsequent formation of a model of the system. The stages in the system model are then numbered according to a specific procedure. The parameters of the system model are read into the program from the model configuration. The construction of a system model is described in detail in appendices A and B.

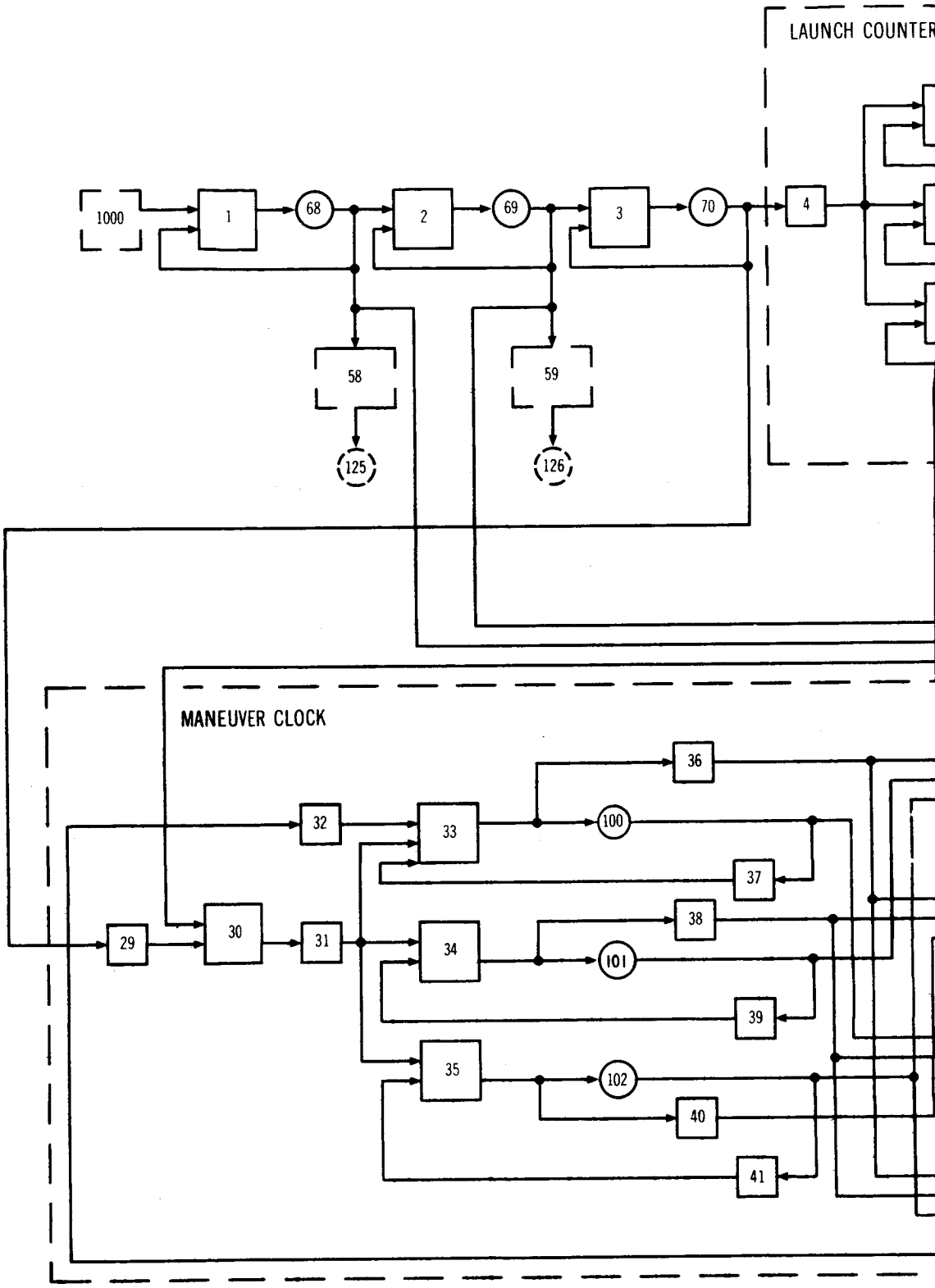
The completed system model of the redundant sequencer is shown in figure 13. All of the output relay circuitry was assumed to be external to the system. Each of the boxes in the figure represents a function stage, and the circles indicate the restorer stages. The individual stages shown in dotted lines in the figure represent special "artificial" stages added to the system. The use of artificial stages is described in the appendices. The purpose of the artificial input stages nos. 1000, 1001, and 1002 is to shorten certain table searching in the programs, and thus decrease program running time. The pairs of artificial output function and restorer stages were added to allow the programs to include the corresponding system outputs in the list of potential test point locations. The single artificial restorer stages are added to all system outputs which are not restored.

The failure rates of the subsystems in each stage are shown in table 7. The failure rates were drawn from the component failure rates listed in the Task II report of the sequencer design study.

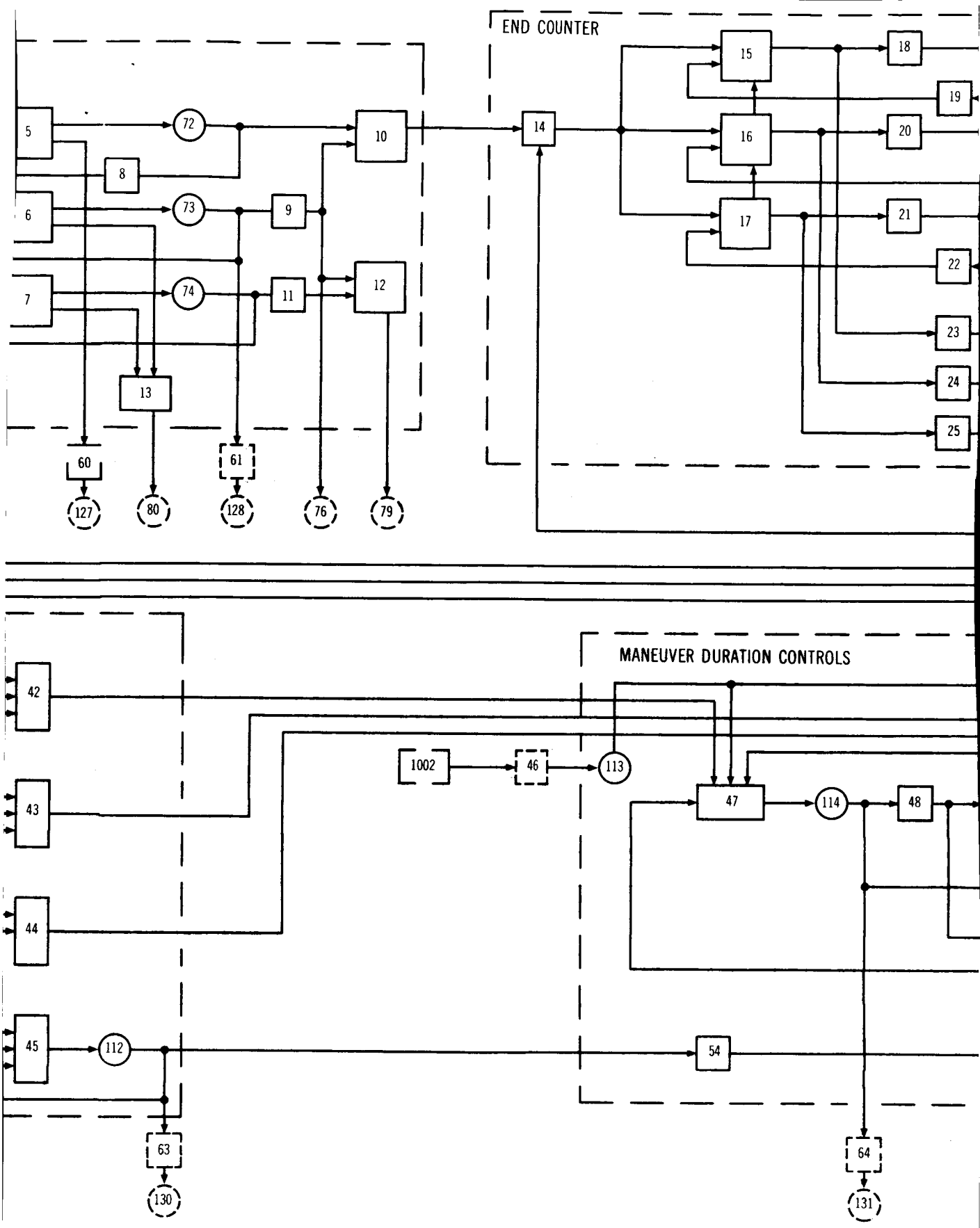
Table 8 shows the results of the test point allocation program. The restorers are listed in the order in which test points should be added to the system. The corresponding value of placing a test point at each of the restorers is also shown.

The reliability analysis program was applied to the redundant sequencer by assigning test points to the first ten test point locations allocated by the test point allocation program. Five different sets of observations at the ten locations were assumed. Table 9 shows the assumed test observations for each of the five sets.

*NASA contract No. NAS 7-100; JPL Subcontract No. 950777, Design Study for a Redundant Spacecraft Sequencer.


LAUNCH COUNTER
1000
1
68
2
69
3
70
4
58
59
125
126
MANEUVER CLOCK
36
32
33
100
37
29
30
31
34
38
101
39
35
102
40
41

4 - 3/4


END COUNTER
MANEUVER DURATION CONTROLS
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
42
43
44
45
46
47
48
54
60
61
63
64
72
73
74
112
113
114
127
128
130
131
1002
80
76
79

4-3/4 (2)

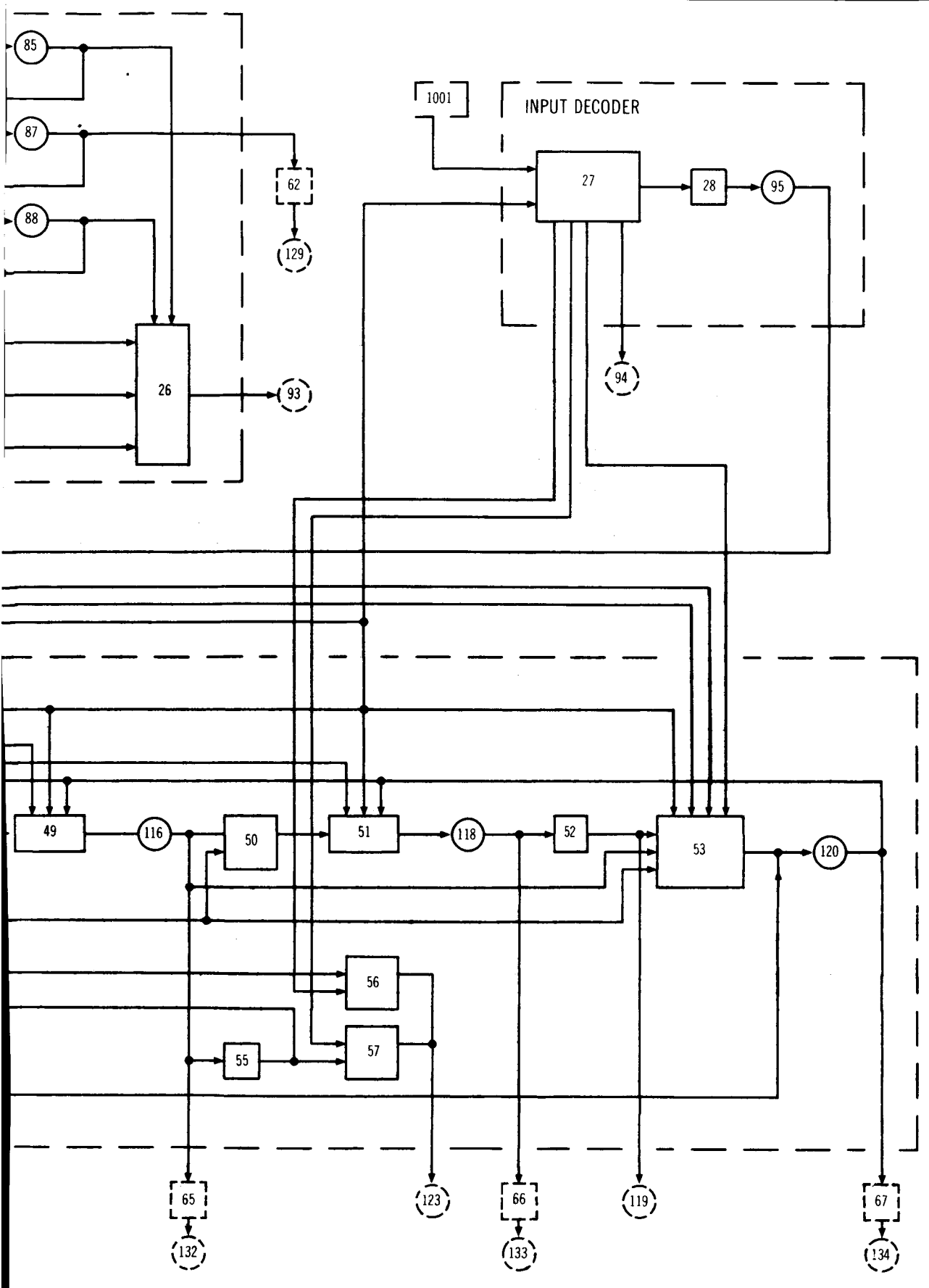


Figure 13. Block Model of Redundant Mariner C Spacecraft Sequencer

Table 7. Block Model Sybsystem Failure Rates

Stages	Failure Rates %/1000 Hours	Stages	Failure Rates %/1000 Hours	Stages	Failure Rates %/1000 Hours
1	0.05004			40	0.00792
2	0.03710	22	0.00100	41-45	0.00100
3	0.03812	23	0.01584	46	0.00916
4	0.02398	24	0.01188	47	0.00206
5	0.00642	25	0.00596	48	0.00143
6	0.01238	26	0.00900	49	0.00206
7	0.00999	27	0.04940	50	0.00143
8-13	0.00100	28-30	0.00100	51	0.00206
14	0.01433	31	0.01270	52	0.00100
15	0.00459	32	0.00226	53	0.04777
16	0.00376	33	0.00339	54	0.00200
17	0.00399	34	0.00256	55	0.00143
18	0.00366	35	0.00279	56, 57	0.00100
19	0.00100	36	0.00792	58-67	0
20	0.00466	37	0.00100	68-114	0.00300
21	0.00170	38	0.00792	115-134	0
		39	0.00100		

For each of the five reliability estimates, the total mission time was assumed to be 7,000 hours. The time of the test was set at 5,000 hours after the system time - zero, the time at which all components are assumed to have been operational. The resulting reliability estimate for each of the five sets of test data is shown in table 10. The initial reliability estimate, at time - zero is 0.999169.

Table 8. Test Points Allocated to Sequencer Model

Test Pt. Number	Value of Placing Test Point	Locate at Restorer Number
1	.513548280-00	93
2	.511247780-00	116
3	.509875730-00	118
4	.508612420-00	123
5	.508393590-00	80
6	.508118720-00	112
7	.507956980-00	95
8	.507903650-00	68
9	.507824170-00	114
10	.507808920-00	94
11	.506581750-00	70
12	.506430430-00	69
13	.506320670-00	73
14	.505965800-00	74
15	.505610970-00	85
16	.505584060-00	127
17	.505584060-00	72
18	.505487620-00	87
19	.505230400-00	88
20	.505175490-00	101
21	.505086230-00	102
22	.505041540-00	100
23	.501372470-00	113
24	.501348630-00	79
25	.500749600-00	120
26	.500599770-00	119
27	.500599770-00	76
28	.500449870-00	134
29	.500449870-00	133
30	.500449870-00	132
31	.500449870-00	131
32	.500449870-00	130
33	.500449870-00	129
34	.500449870-00	128
35	.500449870-00	126
36	.500449870-00	125
37	.500449870-00	153

Table 9. Sets of Test Observations Assumed

Test Point Location	Rank Observed as Failed				
	Set 1	Set 2	Set 3	Set 4	Set 5
93	0	1	1	1	1
116	0	0	0	2	1
118	0	2	1	3	1
123	0	0	0	1	1
80	0	3	1	2	1
112	0	0	0	3	1
95	0	1	1	1	1
68	0	0	0	2	1
114	0	2	1	3	1
94	0	0	0	1	1

Table 10. Resulting Sequencer Reliability Estimates

Set	Reliability Estimate
1	0.999903
2	0.982376
3	0.978401
4	0.970831
5	0.958743

V. CONCLUSIONS

The reliability figure obtained from the foregoing technique is not an exact solution to the reliability of complex systems, but a lower-bound estimate. However, since failure rate estimates themselves are subject to inaccuracies, the analysis time required to produce an exact reliability estimate of each different system to be analyzed is difficult to justify. The block model analysis technique is the simplest and most accurate approximation to system reliability available.

In the test point allocation procedure, the system model used is the block reliability model. The potential test point locations consist of all of the block outputs, the points at which all stages in each block can be tested. Since the block model assumes independence between all blocks, the solution to the value of placing a test point at each of the potential locations is calculated independently of all other locations. The allocation procedure provides optimal placement of a limited number of test points consistent with the independence assumptions of the block model.

The test point allocation technique and the reliability analysis procedure described herein are sufficiently generalized to facilitate their application to virtually any physically realizable system. The modified block reliability analysis procedure is completely compatible with the test point allocation procedure described. Test data obtained from the allocated test points can be used directly in the estimation of system reliability.

Both procedures have been programmed in the FORTRAN IV language for implementation on a general purpose digital computer. The programs provide an extremely simple, efficient means for estimating the reliability of triple-modularly redundant digital systems of any arbitrary configuration. The size and/or complexity of the system to which the procedures may be applied is limited only by computer storage and running time limitations.

APPENDIX A

TEST POINT ALLOCATION PROGRAM

USER'S INSTRUCTION MANUAL

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

I. INTRODUCTION

The test point allocation program allocates a limited number of test points within a modularly redundant digital system. It provides an extremely simple, fast means of determining the optimal placement of test points within a system. The program is written in FORTRAN IV; it is therefore highly machine-independent.

A. WHAT THE PROGRAM DOES

The test point allocation program allocates a limited number of test points within a modularly redundant digital system. The program computes the value[1] of placing a test point at each of the available test point locations in the system. The available test point locations are composed of the inputs to each of the restorers in the system model.

The actual results of the program is a listing of all restorers in the system model, in the order in which test points should be added to the system. Listed with each restorer is the value of placing a test point at that restorer location. With these results, the user can efficiently allocate any limited number of test points to the system, up to a maximum of one test point at each restorer.

B. SYSTEM CONFIGURATION LIMITATIONS

The test point allocation program can be applied only to majority-voted redundant digital systems. The systems must, in addition, be order-three redundant; i. e., all stages, including voter stages, must be triplicated. Within these limitations, there are no restrictions on the configuration of the system. Virtually any degree of system complexity can be handled by the program, including feedback loops, feed-forward, multiple fan-in and fan-out. The size of the system is limited only by computer storage limitations and computer running time.

A summary flow diagram of the test point allocation program is shown in figure A-1. A detailed flow diagram appears in section V of this appendix.

[1] The value of a test point is defined in section III-A of the main body of this report.

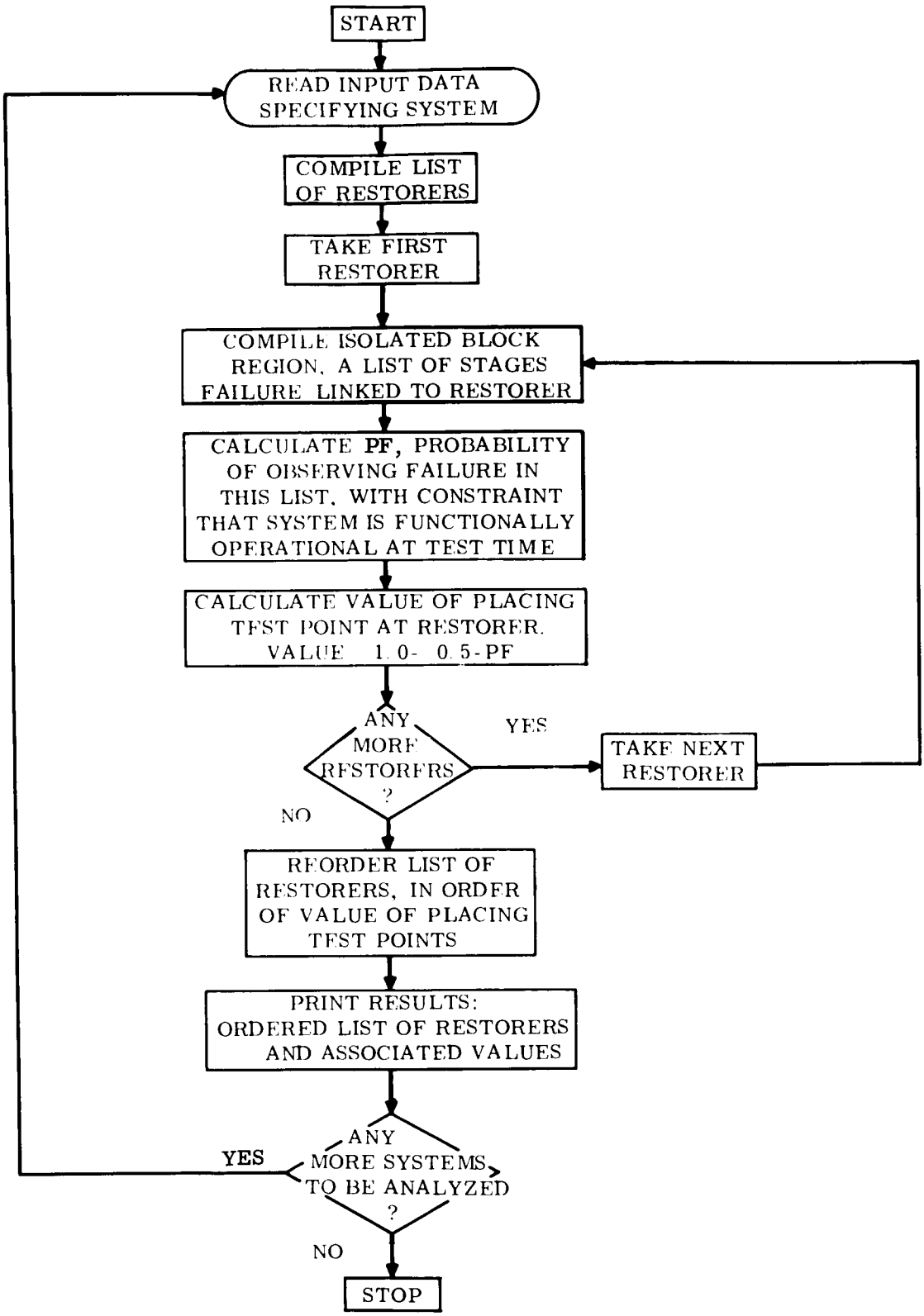


Figure A-1. Summary Flow Diagram of the Test Point Allocation Program

II. HOW TO USE THE PROGRAM

The system to which test points are to be allocated is first converted to simple block diagram form. A system model is then constructed which is very similar to the block diagram. Finally, an ID number is assigned to each stage of the system, according to a procedure described below. The system model thus completed, the exact system configuration can be read into the program. The necessary input data describing the system is compiled directly from the system model.

The following paragraphs describe, first, the construction of the system model, and secondly, the input data required by the program.

A. SYSTEM MODEL

The first step in the construction of the system model is the conversion of the system to block diagram form. Each of the "boxes" in the block diagram indicates either a function stage or a restorer stage. The usual procedure is to use boxes to indicate the function stages, and circles to indicate restorer stages. All interconnections between stages are shown in the diagram, as well as the location of inputs from outside the system and outputs to external equipment. A failure rate is then assigned to each of the three units, or subsystems, in every stage.

1. Artificial Stages

The next step in the modelling procedure is the addition to the block diagram of artificial stages. There are two places in the system at which they might be added: (1) at all of the system inputs, and (2) at some of the system outputs. The following paragraphs describe all of the situations in which these artificial stages are used.

a) Artificial Input Stages

The purpose of placing artificial stages at system inputs is to shorten the program running time, by eliminating unecessary table searching.

Figure A-2 is a block diagram of a simple system to which the program could be applied. The two system inputs are

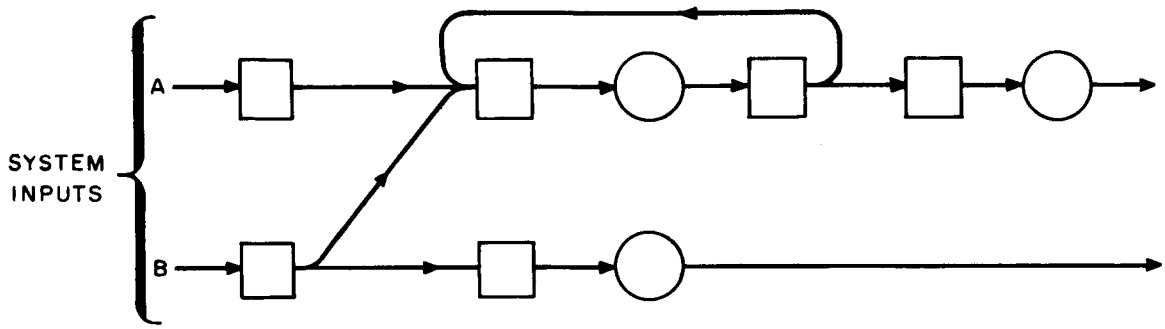


Figure A-2. System Block Diagram, Input Stages Needed

labeled A and B. An artificial stage is added to both A and B, as shown in figure A-3. The input labels A and B in figure A-2 have been replaced by the two artificial stages.

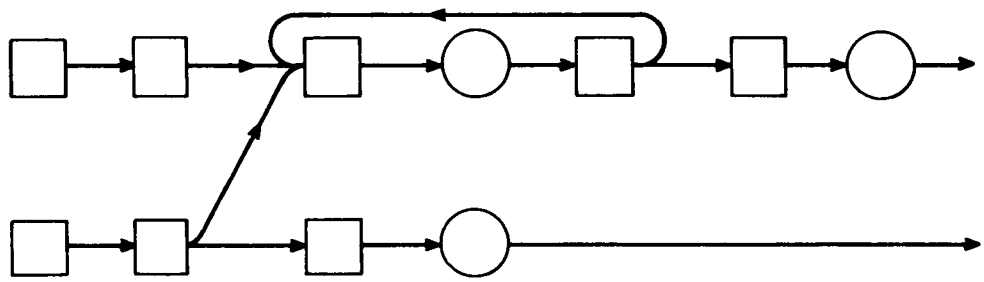


Figure A-3. Input Stages Added to figure A-2

The addition of these input stages completes the model for the system shown. Any failure rate, including zero, can be applied to the artificial input stages because the program does not include these failure rates in any calculations.

b) Artificial Output Stages

The first place at which an artificial output stage must be added is at every system output which does not come directly from a restorer. This situation is shown in figure A-4. The two system outputs are labeled A and B. Output B is not restored,

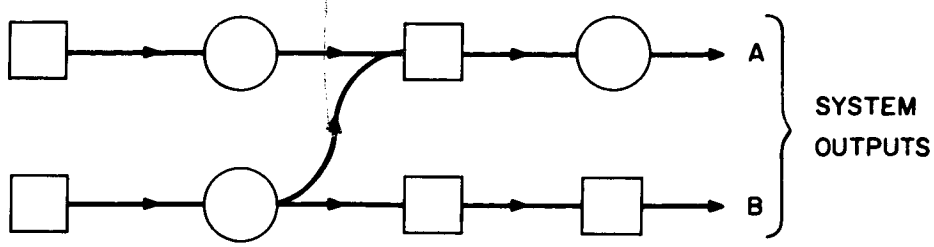


Figure A-4. System needing Artificial Output Stages

so an artificial restorer stage must be added, as shown in figure A-5.

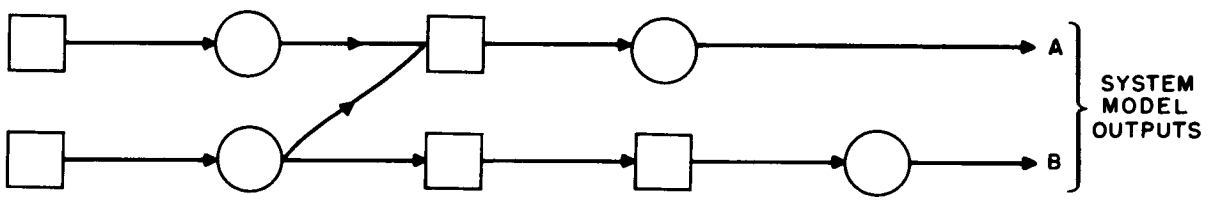


Figure A-5. Output stage added to output B of figure A-4.

There is one other situation in which an output stage must be added. This occurs when the output of an internal restorer is also a system output, and the user wishes to include this output as a potential test point location. Since the program considers placing test points only at restorer inputs, the system model must be altered. The block diagram shown in figure A-6 illustrates this case.

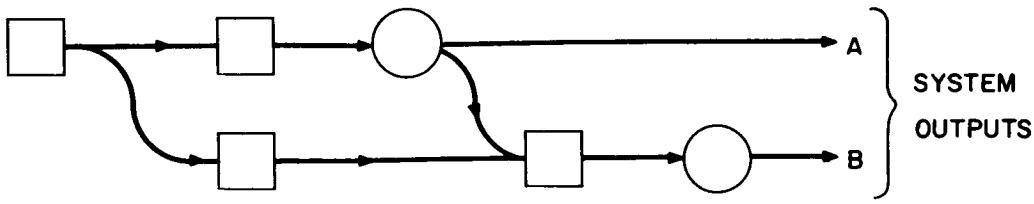


Figure A-6. Potential test point at output A

Assume that the user wishes to include output A as a potential test point location. Since the program considers only restorer inputs as potential locations, an artificial restorer stage must be added at point A. The system model permits restorers to be placed only at the outputs of function stages, however, so an artificial function stage must be inserted between the system restorer and the artificial output restorer. The units in this function stage must be assigned zero failure rates, so that they do not affect the test point value calculations for point A. The failure rates for the artificial restorer stage do not enter into any calculations, so their values are unimportant.

2. Stage ID Numbers

The final step in the construction of the system model is the assignment of an ID number to every stage in the system. The procedure is as follows. First, the function stages, excluding artificial input stages, are numbered from 1 to N, where N is the total number of actual function stages. The restorer stages are then numbered in the following manner. The restorer of function i is assigned an ID number of N+i. The restorer of function 1 is assigned an ID number of N + 1, and the restorer of function N has an ID number of 2N, etc. This numbering procedure enables the program to distinguish between functions and restorers. Finally the artificial input stages are assigned ID numbers of 1,000 or greater, in any order. This enables the program to recognize system inputs when tracing signal paths, thereby eliminating much unnecessary table searching. (For systems with a total number of stages, excluding artificial input stages, of 1,000 or greater, the number 1,000 can be raised in the program by changing the instruction on lines 55 and 96 of the program listing.)

The numbering of the system model stages in the above manner completes the construction of the model. The program input data defining the system is taken directly from the resulting model.

B. INPUT DATA CARDS REQUIRED

This section lists the necessary input data cards for the test point allocation program. The cards are listed in the order in which they must appear in the data deck. Corresponding FORMAT statements and READ specifications for each item are also shown. Each of the following paragraphs describes one data card or group of data cards. (NOTE: The FORMAT (14I5) used to read many of the data cards which have less than 14 items. It is simply a generalized FORMAT, used to reduced the number of separate FORMAT statements.)

LCT, NOF. FORMAT (14I5). READ LCT, NOF.

LCT is the length of the connection table, i. e. , the total number of interconnections in the system model. NOF is the total number of functions in the system model, excluding restorers.

IFR. FORMAT (14I5) READ (IFR(IT), IT=1, LCT).

IFR is the list of stages which provide inputs to other stages in the system model. These stages include any artificial input stages. The position of each entry in IFR must be exactly the same as the position of the corresponding stage in the ITO list, the stage which receives the input.

ITO FORMAT (14I5) READ (ITO(IT), IT=1, LCT).

ITO is the list of stages which recieve outputs from other stages in the system model. These stages include any artificial output stages in the model. The position of each entry in ITO must be exactly the same as the position of the corresponding stage in the IFR list, the stage which provides the input.

TO, T1. FORMAT (2F10. 0). READ TO, T1.

TO is the time zero of the system model; i. e. the time at which all system components are assumed to have been operational. T1 is the time at which the test will be made. The units used for these times must, of course, correspond to the time unit used for the failure rates.

IEQULR. FORMAT (14I5) READ IEQULR.

This is the equal reliability flag which indicates, if it equals one, that all three units in every stage have equal failure rates. If different failure rates are assigned to the units of any stage, IEQULR is made zero and a separate failure rate is read in for every unit in the system.

P (IST, 1, 1). FORMAT (6E 12. 6). READ (P(IST, 1, 1), IST=1, KNO)

This is the list of failure rates for the system units. One failure rate is read in for each stage in the system, and this value is used for the three units in the stage. The failure rates for the function stages are read in first, followed by those of the restorer stages. For the case of a function which is not restored, the field corresponding to the appropriate restorer number may be left blank. The failure values for non-present voters is not used by the program. This READ statement is used only when IEQULR=1. If IEQULR=0, the following read statement is used.

P(IST, IRK, 1). FORMAT (6E12. 6). READ ((P(IST, IRK, 1), IRK=1, 3), IST=1, KNOF).

This is the list of failure rates for the system units. One failure rate is read in for each unit in the system. The failure rates of the three units in a stage are read in consecutively, and values are read in according to stage ID numbers; i. e. function stages first, followed by restorer stages. This READ statement is used only when IEQULR=0. If IEQULR=1, the preceding read statement is used.

LPRINT. FORMAT (14I5). READ LPRINT.

This is the printout option flag, which specifies which of two printout modes will be employed. When LPRINT=0 the "normal operation" mode is used, in which the program prints only a listing of input data and final results of the analysis. When IPRINT=1, the "debug operation" is employed, in which the program prints the above information, plus many of the intermediate computational results.

III. OUTPUT TO BE EXPECTED

This section outlines the printout to be expected from the test point allocation program. The first part of the section describes the output obtained during normal operation of the program.

Following this is a description of the output available during possible debugging operations, providing the user with a more extensive view of the intermediate computational results. An input data constant, LPRINT, specifies which of the output options will be used in any one program run.

A. NORMAL OPERATION

The normal operation printout mode is specified by setting the input data constant, LPRINT, to zero. In this mode, the program prints, first, a listing of the input data specifying the system which has been analyzed, and, secondly, the results of the analysis.

The first item to be printed is a complete list of the system interconnections. This listing is followed by the statement of the total number of system function stages, the zero time of the system, TO, and the time of test, T1. Next, the failure rate of each unit, or subsystem, is listed. The failure rates of the units in each restorer are listed on the same line as those of the corresponding function stage.

The printout of the results of the program analysis consists of a listing of the optimum test point locations. These locations are listed as the restorers at which the test points should be located, arranged in descending order of test point value. The listing is arranged in table form. Each line consists of: (1) the number of the test point; (2) the value of placing it at the indicated location; and (3) the restorer at whose input the test point should be located.

A sample of the printout from the normal operation mode is shown on the following page.

TEST POINT ALLOCATIONS

SYSTEM INTERCONNECTIONS

FROM	TO	FROM	TO	FROM	TO	FROM	TO	FROM	TO
1000	1	1	6	6	2	6	5	5	3
5	4	2	7	3	8	4	9		

THIS SYSTEM CONTAINS 5 FUNCTIONS

TIME ZERO = 0.
TEST TIME = 50.

UNIT FAILURE RATES

	FUNCTIONS			RESTORERS		
STAGE	RANK 1	RANK 2	RANK 3	RANK 1	RANK 2	RANK 3
1	.100503-03	.100503-03	.100503-03	.100503-03	.100503-03	.100503-03
2	.100503-03	.100503-03	.100503-03	.100503-03	.100503-03	.100503-03
3	.100503-03	.100503-03	.100503-03	.100503-03	.100503-03	.100503-03
4	.100503-03	.100503-03	.100503-03	.100503-03	.100503-03	.100503-03
5	.100503-03	.100503-03	.100503-03	.100503-03	.100503-03	.100503-03

TEST PT. NUMBER	VALUE OF PLACING A TEST POINT	LOCATE AT RESTORER NUMBER
1	.543583020-00	9
2	.543583020-00	8
3	.529411690-00	7
4	.514888410-00	6

B. DEBUG OPERATION

The debug operation printout mode is specified by setting the input data constant, LPRINT, to one. In this mode, the program prints all of the information provided by the normal operation mode. In addition, further information is printed to enable a user to examine some of the intermediate computational operations in greater detail.

The first additional listing is composed of the stages which would be tested at the first potential test point location examined by the program. This list is followed by the conditional probability that a failure will be observed at that location, given that the system is working. The program then prints the value of placing a test point at that location. These items are printed for each potential test point location in turn, as it is examined.

With the additional information of the debug operation mode, the program user can obtain a check on both the operation of the program and the correctness of the input data. The listing of the stages tested at each potential test point location provides a further check on the completeness of the system connection list, in addition to that provided by the printing of the input data.

A sample of the printout from the debug operation mode is shown on the following page. The system represented in this case is the same one previously shown for the normal operation mode.

TEST POINT ALLOCATIONS

SYSTEM INTERCONNECTIONS

FROM	TO	FROM	TO	FROM	TO	FROM	TO	FROM	TO
1000	1	1	6	6	2	6	5	5	3
5	4	2	7	3	8	4	9		

THIS SYSTEM CONTAINS 5 FUNCTIONS

TIME ZERO = 0.
TEST TIME = 50.

UNIT FAILURE RATES

	FUNCTIONS			RESTORERS		
STAGE	RANK 1	RANK 2	RANK 3	RANK 1	RANK 2	RANK 3
1	.100503-03	.100503-03	.100503-03	.100503-03	.100503-03	.100503-03
2	.100503-03	.100503-03	.100503-03	.100503-03	.100503-03	.100503-03
3	.100503-03	.100503-03	.100503-03	.100503-03	.100503-03	.100503-03
4	.100503-03	.100503-03	.100503-03	.100503-03	.100503-03	.100503-03
5	.100503-03	.100503-03	.100503-03	.100503-03	.100503-03	.100503-03

POTENTIAL TEST LOCATION AT RESTORER NO. 6
TESTED STAGES

1
PROBABILITY THAT A FAILURE IS OBSERVED= .148884-01
VALUE OF PLACING A TEST POINT HERE= .514888-00

POTENTIAL TEST LOCATION AT RESTORER NO. 7
TESTED STAGES

2 6
PROBABILITY THAT A FAILURE IS OBSERVED= .294117-01
VALUE OF PLACING A TEST POINT HERE= .529412-00

POTENTIAL TEST LOCATION AT RESTORER NO. 8
TESTED STAGES

3 5 6
PROBABILITY THAT A FAILURE IS OBSERVED= .435830-01
VALUE OF PLACING A TEST POINT HERE= .543583-00

POTENTIAL TEST LOCATION AT RESTORER NO. 9
TESTED STAGES

4 5 6
PROBABILITY THAT A FAILURE IS OBSERVED= .435830-01
VALUE OF PLACING A TEST POINT HERE= .543583-00

TEST PT. NUMBER	VALUE OF PLACING A TEST POINT	LOCATE AT RESTORER NUMBER
1	.543583020-00	9
2	.543583020-00	8
3	.529411690-00	7
4	.514888410-00	6

IV. PROGRAM VARIABLES AND CONSTANTS

This section contains a list of important program variables and constants with a brief explanation of each item. It is intended as an aid to the program user who requires a more detailed description of the program operations.

B1 - Non-subscripted variable - B1 is the probability that rank 1 of a given block is operational at test time. This quantity is used in the calculation of the value of placing a test point at the output of the block.

B2 - Non-subscripted variable - B2 is the probability that rank 2 of a given block is operational at test time. It is used in exactly the same manner as B1.

B3 - Non-subscripted variable - B3 is the probability that rank 3 of a given block is operational at test time. It is used in exactly the same way as B1.

IBR - Subscripted variable - This array is used to store the locations in the block lists. It stores the ID numbers of stages with multiple inputs, in the order in which they are encountered during the tracing of a signal path. When the beginning of a path is reached, the program goes to the last entry in IBR, which is the ID number of the last branch location passed. The program then traces the next branch with an input to this location.

IBRX - Subscripted variable - This array is used to store the locations in the connection table of the branch locations stored in IBR. This connection table location is used as a starting point in the search for the next branch with an input to the fan-in stage.

IEQULR - Non-subscripted constant - may have one of two values:

0 - a separate failure rate is read in for each unit in every stage.

1 - a failure rate is read in for each stage, and the value is used as the failure rate for each of the three units in the stage.

IFR - Subscripted constants - This array is the "from" list of the connection table. It stores the ID numbers of the stages which provide inputs to other stages in the system model. A given system connection will initiate two entries in the connection table: the stage providing the output will be entered in the "from" list, IFR, and the stage receiving this output will be entered the same location in the "to" list, ITO.

IORDRD - Subscripted variables - This array stores an ordered list of restorer locations at which test points may be placed. The restorers are listed in order of descending "value". There is a one-to-one correspondence between the location of restorers in IORDRD, and the location of associated values in PORDRD.

LPRINT - Non-subscripted constant - This is the printout option flag, which specifies which of two printout options will be employed. LPRINT may have one of two values:

0 - "Normal operation" mode. The program prints a listing of input data specifying the system analyzed, plus the final analysis results.

1 - "Debug operation" mode. In addition to the printout obtained in the normal mode, the program prints many of the intermediate computational results.

IRL - Subscripted variable - This matrix stores the completed block lists, after they are completed. IRL stores one complete list for each restorer and system output.

IRST - Subscripted variable - This array holds the complete list of restorer ID numbers. The list is compiled from the connection table, by searching through the table for ID numbers which are greater than NOF, the number of functions, and less than 1,000.

ITO - Subscripted constants - This array is the "to" list of the connection table. It stores the ID numbers of stages which receive outputs of other stages in the system model. See IFR, in this section.

IUSD - Subscripted variable - This array is used in the generation of a list of subsystems which have multiple inputs (fan-in units). This latter list is used in the construction of the block lists. It is constructed by searching through the connection table for stages with two or more inputs. When a stage ID number is put in the fan-in list NFB, then the corresponding

location in IUSD is changed from zero to one. This assures that the program does not examine the connection table entry repeatedly. There is a one-to-one correspondence between connection table locations and IUSD locations.

KNOF - Non-subscripted constant - KNOF is equal to twice the number of functions in the system model (KNOF = 2x NOF This is the total number of stages in a system with NOF functions, all of which are restored.

LCT - Non-subscripted constant - LCT is the Length of the Connection Table, the total number of connections in the system model.

LCT1 - Non-subscripted constant - LCT1 is equal to LCT-1. It is used as a DO index limit for table searching.

LL - Subscripted variables - LL stores the lengths of the block lists stored in IRL.

NFB - Subscripted variable - This array holds a list of fan-ins; i. e., stages having two or more inputs. NFB is used for locating branches during the construction of block lists. It is compiled by a comparison of entries in the connection table.

NFI - Non-subscripted variable - NFI is the total number of stages in the system model which have two or more inputs. The value of NFI is therefore the length of the fan-in list, NFB.

NI - Subscripted variable - This array stores the number of inputs to each of the stages on the fan-in list, NFB. There is a one-to-one correspondence between entries in NFI and those in NFB.

NOF - Non-subscripted constant - NOF is the total Number of Functions in the system model.

NR - Non-subscripted variable - NR is used as a DO loop index. Its value at a given time represents the location in the voter list of the voter whose block list is being utilized.

NRT - Non-subscripted variable - NRT is equal to the total number of restorers in the system model. Its value is computed during the construction of the voter list, IRST.

P - Subscripted variables - This array stores all of the unit failure rates, which are read in as data. It stores, also, the probability that each unit in the system model is operational at test time. This latter item is computed by the program. There are two storage locations in P for each unit, or subsystem, in the system model.

PF - Subscripted variables - This array stores the list of values computed for all block lists. These values indicate the optimum placement of test points within the system model.

PORDRD - Subscripted variables - This array stores an ordered list of the values stored in PF. A list of the restorers associated with this ordered list is stored in IORDRD.

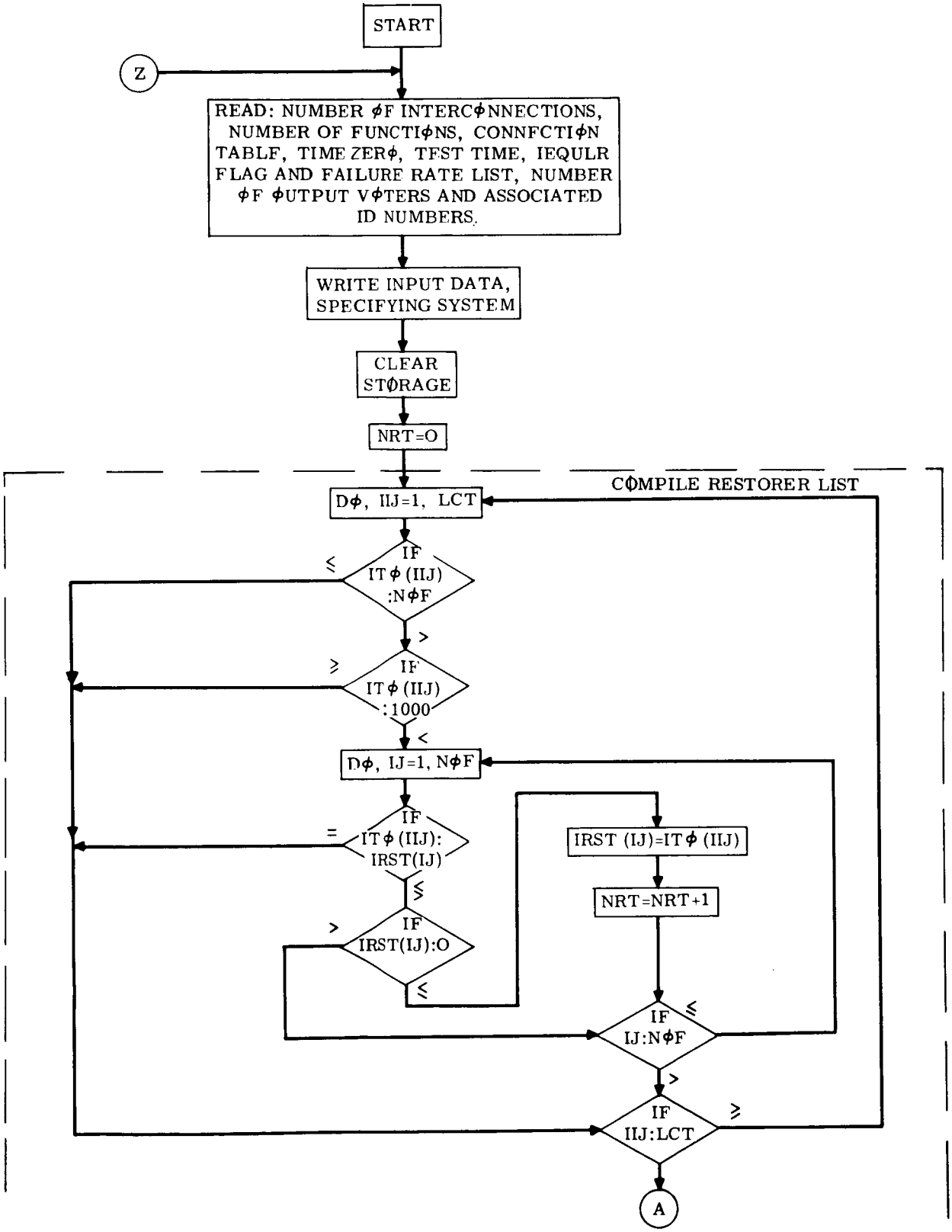
R - Non-subscripted variable - R is the probability that the last block constructed is functionally operational; i.e., that at least two of its three ranks are operational.

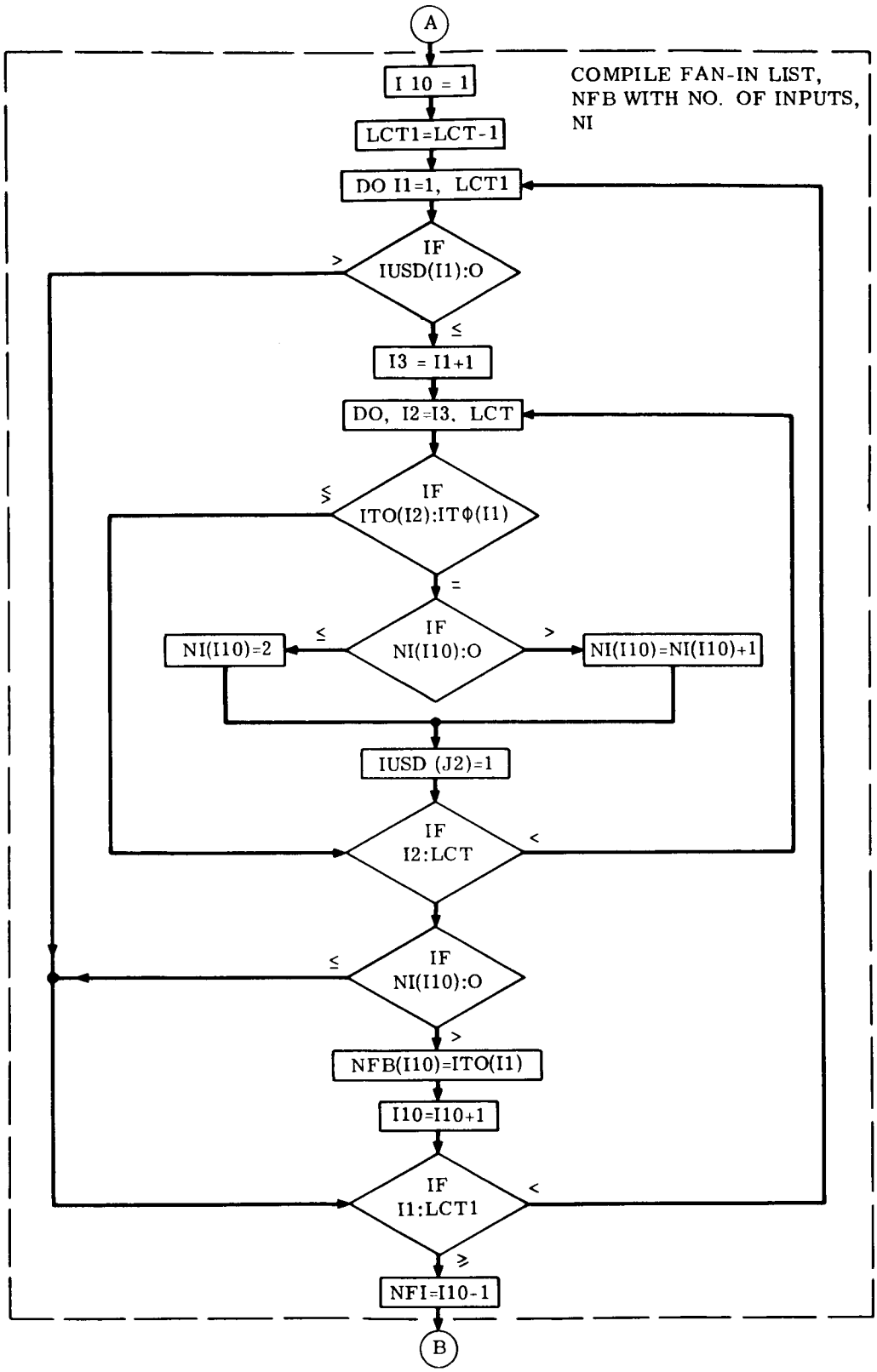
T0 - Non-subscripted constant - T0 is the time zero of the system; i.e. the time at which all subsystems are assumed to have been operational. The units used for T0 must be the same as those of T1 and the subsystem failure rates.

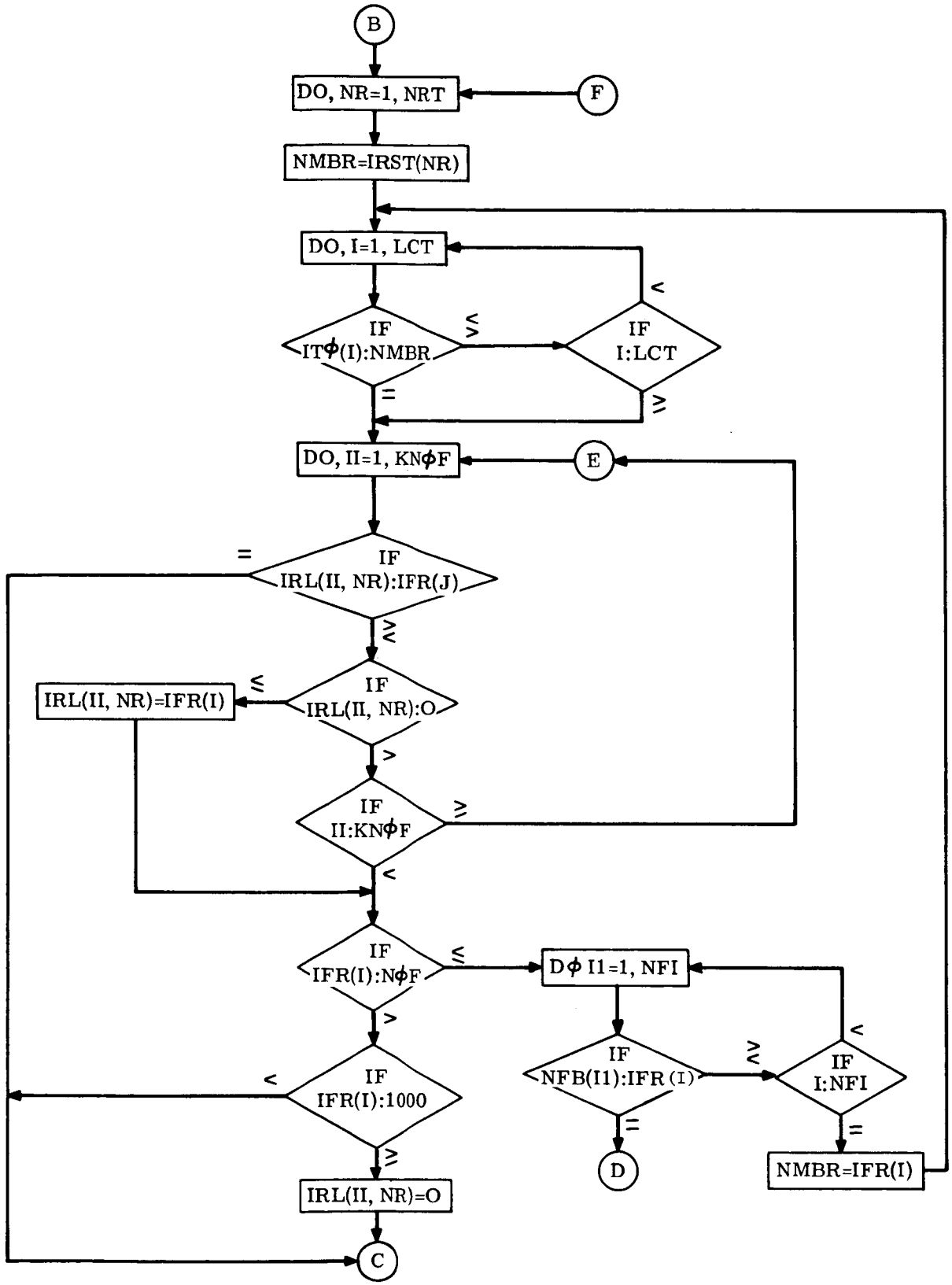
T1 - Non-subscripted constant - T1 is the time of test. The units used for T1 must be the same as those of T0 and the subsystem failure rates.

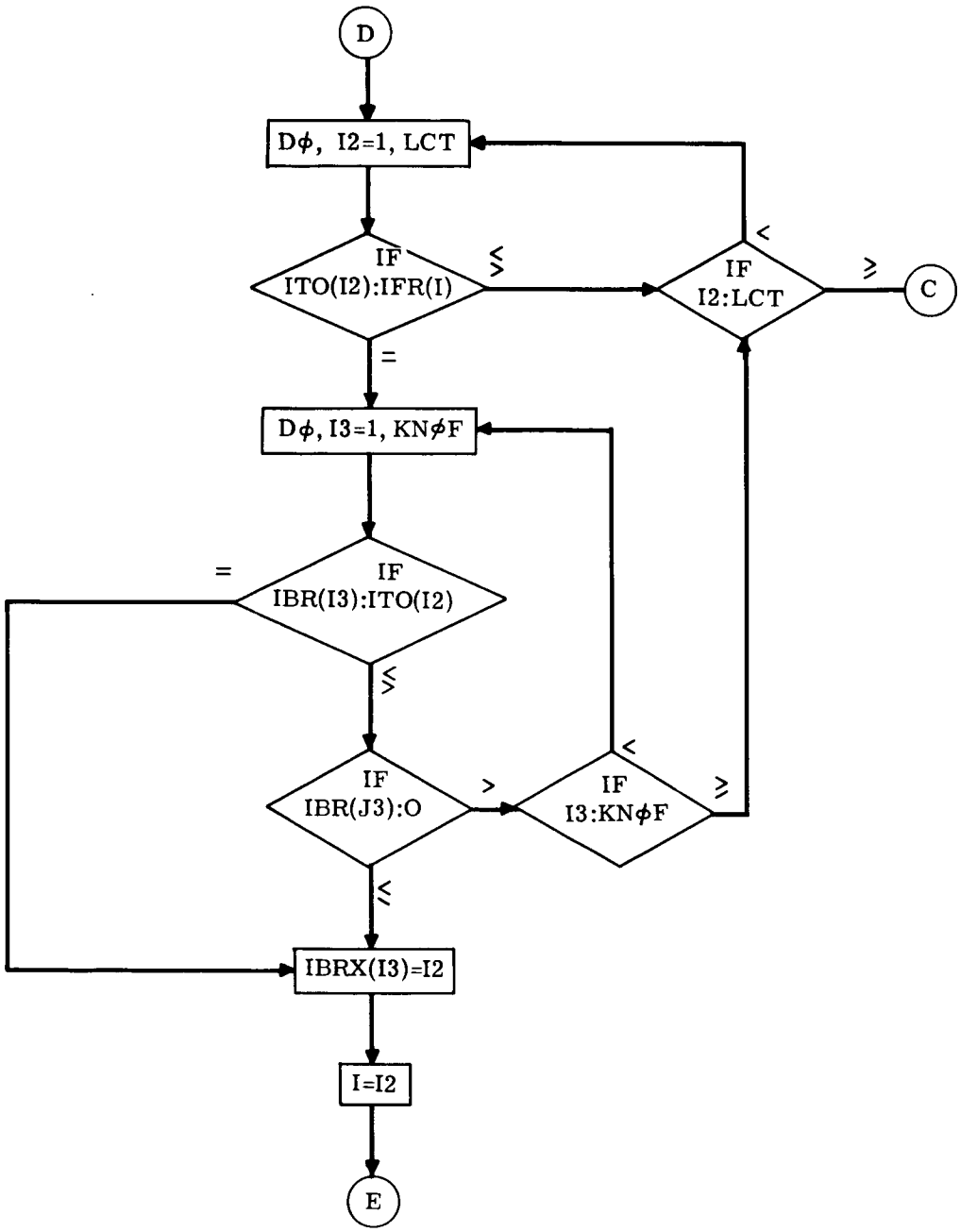
V. PROGRAM FLOW CHART

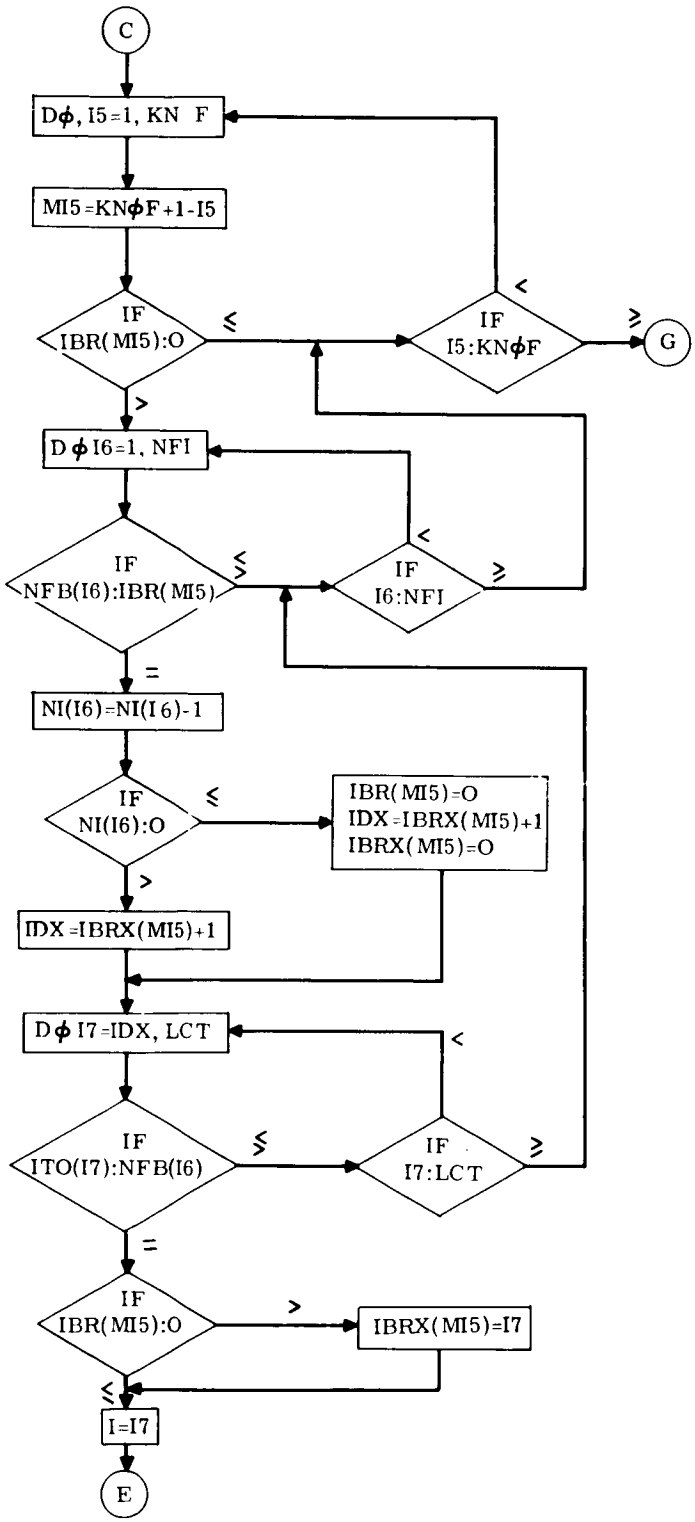
The following pages contain a detailed flow chart of the test point allocation program.

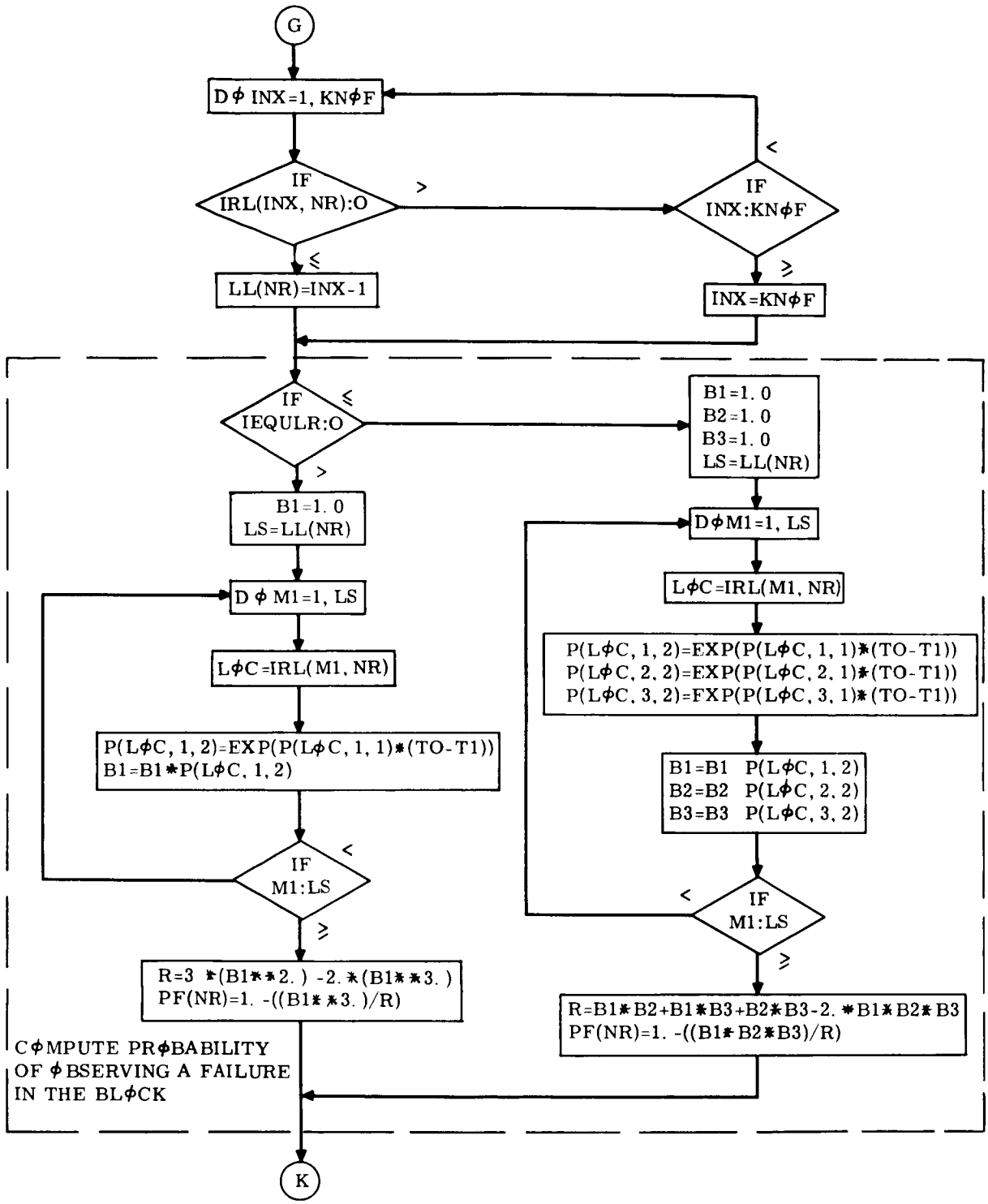


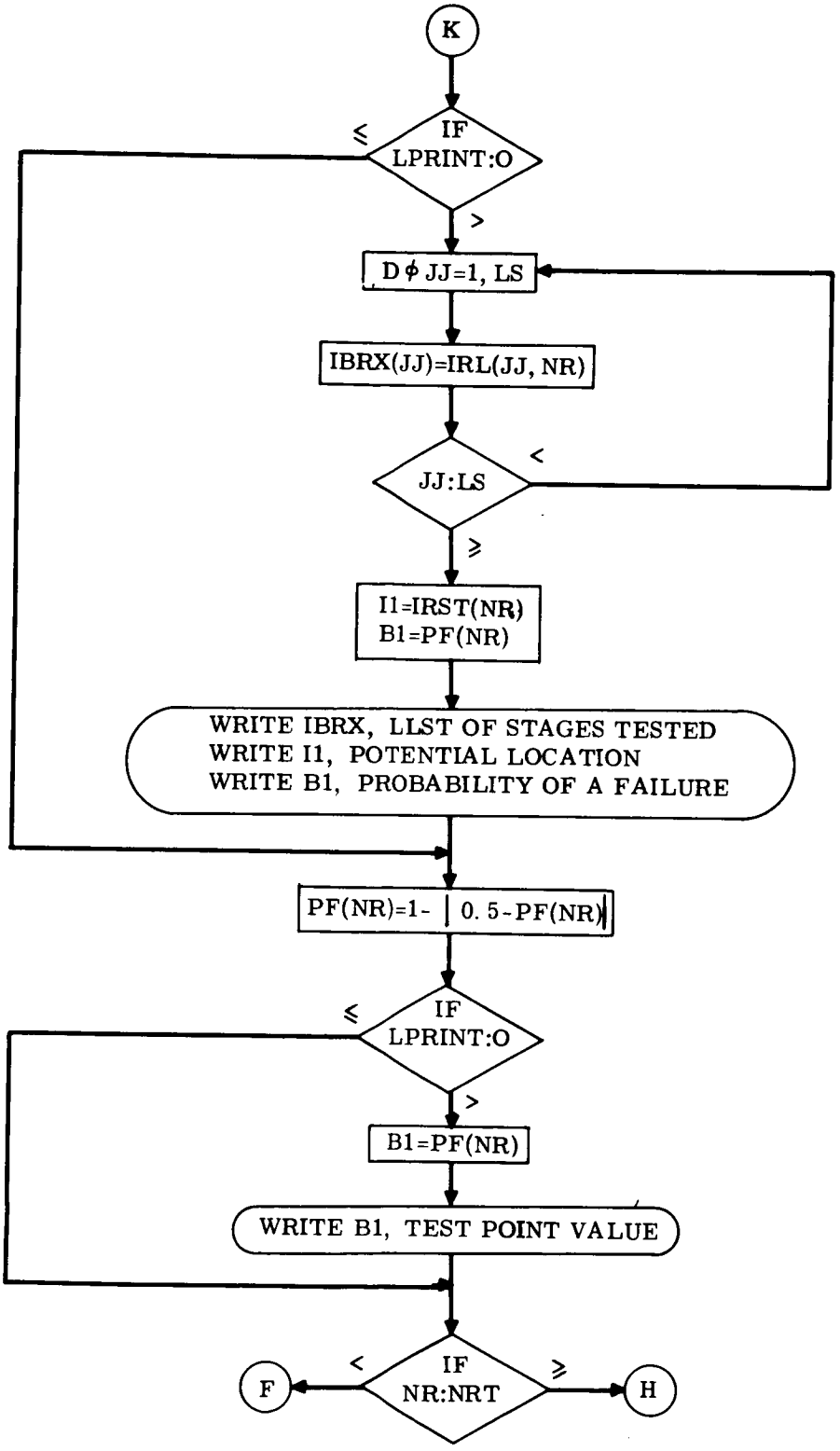

A
COMPILE FAN-IN LIST, NFB WITH NO. OF INPUTS, NI
I 10 = 1
LCT1=LCT-1
DO I1=1, LCT1
IF IUSD(I1):O
>
≤
I3 = I1+1
DO, I2=I3, LCT
IF ITO(I2):ITΦ(I1)
≤
=
IF NI(I10):O
≤
>
NI(I10)=2
NI(I10)=NI(I10)+1
IUSD (J2)=1
IF I2:LCT
<
IF NI(I10):O
≤
>
NFB(I10)=ITO(I1)
I10=I10+1
IF I1:LCT1
<
≥
NFI=I10-1
B

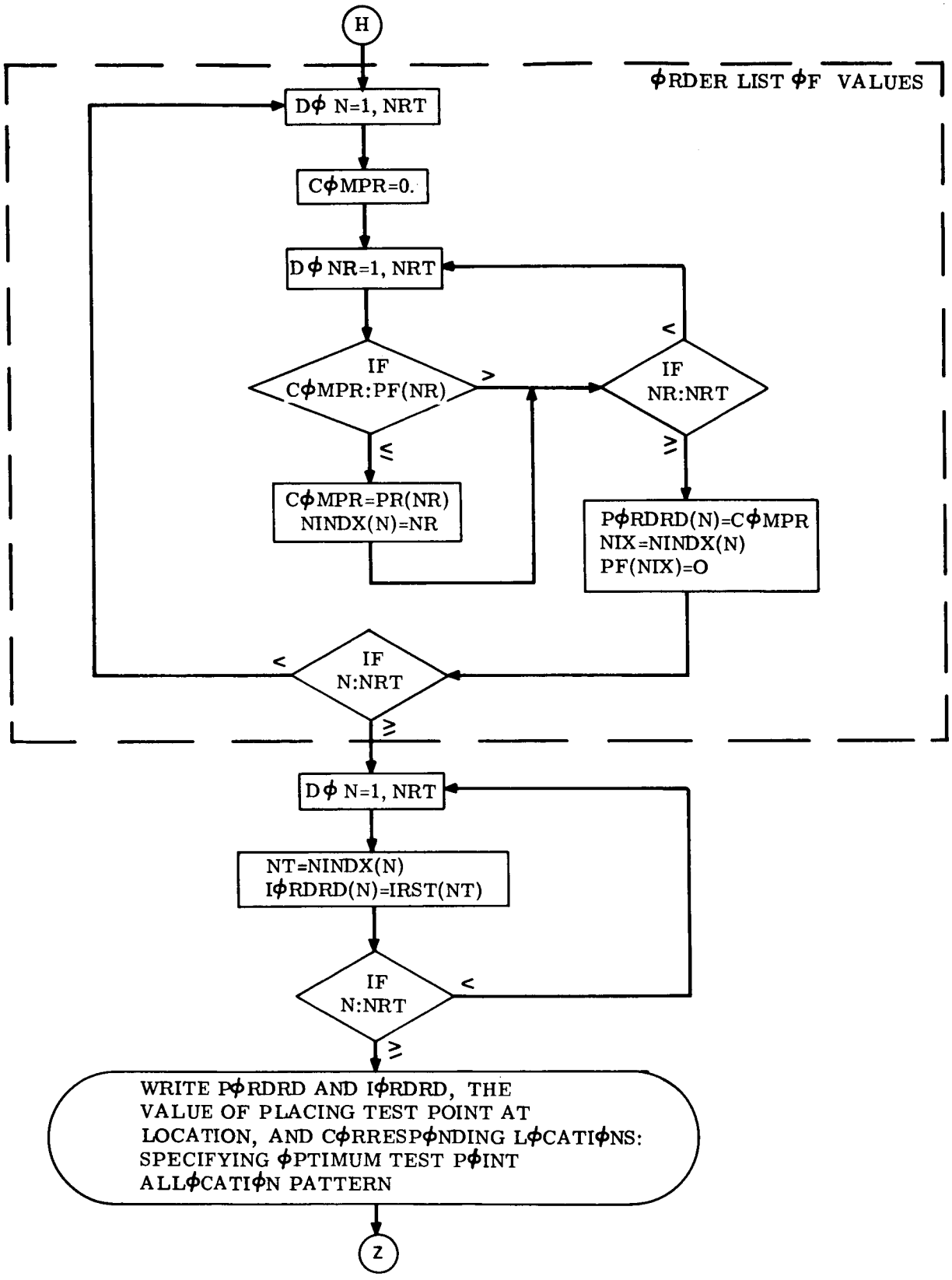

B
DO, NR=1, NRT
F
NMBR=IRST(NR)
DO, I=1, LCT
IF ITϕ(I):NMBR
IF I:LCT
DO, II=1, KNϕF
E
IF IRL(II, NR):IFR(J)
IRL(II, NR)=IFR(I)
IF IRL(II, NR):O
IF II:KNϕF
IF IFR(I):NϕF
Dϕ I1=1, NFI
IF NFB(I1):IFR(I)
IF I:NFI
NMBR=IFR(I)
D
IF IFR(I):1000
IRL(II, NR)=O
C


D
Dϕ, I2=1, LCT
IF
ITO(I2):IFR(I)
≤
≥
IF
I2:LCT
<
≥
C
=
Dϕ, I3=1, KNϕF
=
IF
IBR(I3):ITO(I2)
≤
IF
IBR(J3):O
>
IF
I3:KNϕF
<
≥
≤
IBRX(I3)=I2
I=I2
E


C
Dϕ, I5=1, KN F
MI5=KNϕF+1-I5
IF IBR(MI5):O
IF I5:KNϕF
G
Dϕ I6=1, NFI
IF NFB(I6):IBR(MI5)
IF I6:NFI
NI(I6)=NI(I6)-1
IF NI(I6):O
IBR(MI5)=O
IDX=IBRX(MI5)+1
IBRX(MI5)=O
IDX=IBRX(MI5)+1
Dϕ I7=IDX, LCT
IF ITO(I7):NFB(I6)
IF I7:LCT
IF IBR(MI5):O
IBRX(MI5)=I7
I=I7
E


G
DΦ INX=1, KNΦF
IF IRL(INX, NR):O
>
IF INX:KNΦF
<
≤
≥
LL(NR)=INX-1
INX=KNΦF
IF IEQULR:O
≤
>
B1=1. 0
B2=1. 0
B3=1. 0
LS=LL(NR)
B1=1. 0
LS=LL(NR)
DΦM1=1, LS
D Φ M1=1, LS
LΦC=IRL(M1, NR)
P(LΦC, 1, 2)=EXP(P(LΦC, 1, 1)*(TO-T1))
P(LΦC, 2, 2)=EXP(P(LΦC, 2, 1)*(TO-T1))
P(LΦC, 3, 2)=FXP(P(LΦC, 3, 1)*(TO-T1))
LΦC=IRL(M1, NR)
P(LΦC, 1, 2)=EXP(P(LΦC, 1, 1)*(TO-T1))
B1=B1*P(LΦC, 1, 2)
B1=B1 P(LΦC, 1, 2)
B2=B2 P(LΦC, 2, 2)
B3=B3 P(LΦC, 3, 2)
IF M1:LS
<
≥
IF M1:LS
<
≥
R=3 *(B1**2.) -2. *(B1**3.)
PF(NR)=1. -((B1**3.)/R)
R=B1*B2+B1*B3+B2*B3-2. *B1*B2*B3
PF(NR)=1. -((B1*B2*B3)/R)
CΦMPUTE PRΦBABILITY
OF ΦBSERVING A FAILURE
IN THE BLΦCK
K


K
IF
LPRINT:O
≤
>
DϕJJ=1, LS
IBRX(JJ)=IRL(JJ, NR)
JJ:LS
<
≥
I1=IRST(NR)
B1=PF(NR)
WRITE IBRX, LLST OF STAGES TESTED
WRITE I1, POTENTIAL LOCATION
WRITE B1, PROBABILITY OF A FAILURE
PF(NR)=1-|0.5-PF(NR)|
IF
LPRINT:O
≤
>
B1=PF(NR)
WRITE B1, TEST POINT VALUE
IF
NR:NRT
<
≥
F
H


H
ΦRDER LIST ΦF VALUES
DΦ N=1, NRT
CΦMPR=0.
DΦ NR=1, NRT
IF CΦMPR:PF(NR)
>
<
IF NR:NRT
≤
≥
CΦMPR=PR(NR)
NINDX(N)=NR
PΦRDRD(N)=CΦMPR
NIX=NINDX(N)
PF(NIX)=O
IF N:NRT
<
≥
DΦ N=1, NRT
NT=NINDX(N)
IΦRDRD(N)=IRST(NT)
IF N:NRT
<
≥
WRITE PΦRDRD AND IΦRDRD, THE VALUE OF PLACING TEST POINT AT LOCATION, AND CΦRRESPΦNDING LΦCATIΦNS: SPECIFYING ΦPTIMUM TEST PΦINT ALLΦCATIΦN PATTERN
Z

VI. FORTRAN IV PROGRAM LISTING

The following pages contain a complete FORTRAN IV listing of the test point allocation program.

```
00100     1.   C      TEST PT ALLOCATION
00101     2.          DIMENSION IRL(150,75),IFR(200),ITO(200),IRST(75),NFB(150)
00103     3.          DIMENSION NI(150),IBR(150),IBRX(150),LL(75),NINDX(150),IUSD(200)
00104     4.          DIMENSION P(150,3,2),PF(75),PORDRD(75),IORDRD(75)
00105     5.    400   READ(5,301)LCT,NOF
00111     6.          READ(5,301)(IFR(IT),IT=1,LCT)
00117     7.          READ(5,301)(ITO(IT),IT=1,LCT)
00125     8.          READ(5,305)T0,T1
00131     9.          READ(5,301)IEQULR
00134    10.          KNOF=2*NOF
00135    11.          IF(IEQULR)450,450,451
00140    12.    451   READ(5,307)(P(IST,1,1),IST=1,KNOF)
00146    13.          DO 800 IST=1,KNOF
00151    14.          P(IST,2,1)=P(IST,1,1)
00152    15.    800   P(IST,3,1)=P(IST,1,1)
00154    16.          GO TO 601
00155    17.    450   READ(5,307)((P(IST,IRK,1),IRK=1,3),IST=1,KNOF)
00166    18.    601   READ(5,301)LPRINT
00171    19.          WRITE(6,309)
00173    20.          WRITE(6,310)(IFR(I),ITO(I),I=1,LCT)
00202    21.          WRITE(6,311)NOF
00205    22.          WRITE(6,312)T0,T1
00211    23.          WRITE(6,314)(N,P(N,1,1),P(N,2,1),P(N,3,1),P(N+NOF,1,1),P(N+NOF,2,1
00211    24.         1),P(N+NOF,3,1),N=1,NOF)
00211    25.   C-----CLEAR STORAGE
00225    26.          DO 108 LL1=1,KNOF
00230    27.          IBR(LL1)=0
00231    28.          IBRX(LL1)=0
00232    29.          NI(LL1)=0
00233    30.          NFB(LL1)=0
00234    31.          IUSD(LL1)=0
00235    32.          DO 108 NR=1,NOF
00240    33.          IRST(NR)=0
00241    34.    108   IRL(LL1,NR)=0
00244    35.          R=1.
00244    36.   C-----COMPILE RESTORER LIST
00245    37.          NRT=0
00246    38.          DO 102 IIJ=1,LCT
00251    39.          IF(ITO(IIJ)-NOF)102,102,103
00254    40.    103   IF(ITO(IIJ)-1000)104,102,102
00257    41.    104   DO 101 IJ=1,NOF
00262    42.          IF(ITO(IIJ)-IRST(IJ))105,102,105
00265    43.    105   IF(IRST(IJ))106,106,101
00270    44.    106   IRST(IJ)=ITO(IIJ)
00271    45.          NRT=NRT+1
00272    46.          GO TO 102
00273    47.    101   CONTINUE
00275    48.    102   CONTINUE
00275    49.   C-----COMPILE FAN-IN LIST,NFB,AND NO. OF INPUTS,NI
00277    50.          I10=1
00300    51.          LCT1=LCT-1
00301    52.          DO 908 I1=1,LCT1
00304    53.          IF(IUSD(I1))901,901,908
00307    54.    901   I3=I1+1
00310    55.          DO 906 I2=I3,LCT
00313    56.          IF(ITO(I2)-ITO(I1))906,902,906
00316    57.    902   IF(NI(I10))903,903,904
00321    58.    903   NI(I10)=2
00322    59.          GO TO 905
00323    60.    904   NI(I10)=NI(I10)+1
00324    61.    905   IUSD(I2)=1
00325    62.    906   CONTINUE
00327    63.          IF(NI(I10))908,908,907
00332    64.    907   NFB(I10)=ITO(I1)
00333    65.          I10=I10+1
```

```
00334    66.     908  CONTINUE
00336    67.          NFI=I10-1
00336    68.    C-----BEGIN COMPILING ISOLATED BLOCKS
00337    69.     111  DO 70 NR=1,NRT
00342    70.          NMBR=IRST(NR)
00343    71.     9    DO 3 I=1,LCT
00346    72.          IF(ITO(I)-NMBR)3,2,3
00351    73.     3    CONTINUE
00353    74.     2    DO 15 II=1,KNOF
00356    75.          IF(IRL(II,NR)-IFR(I))14,25,14
00361    76.     14   IF(IRL(II,NR))16,16,15
00364    77.     16   IRL(II,NR)=IFR(I)
00365    78.          GO TO 4
00366    79.     15   CONTINUE
00370    80.     4    IF(IFR(I)-NOF)5,5,109
00373    81.     109  IF(IFR(I)-1000)25,110,110
00376    82.     110  IRL(II,NR)=0
00377    83.          GO TO 25
00400    84.     5    DO 7 I1=1,NFI
00403    85.          IF(NFB(I1)-IFR(I))7,8,7
00406    86.     7    CONTINUE
00410    87.          NMBR=IFR(I)
00410    88.    C-----IFR IS NOT A FAN-IN STAGE
00411    89.          GO TO 9
00411    90.    C-----IFR IS A FAN-IN STAGE
00412    91.     8    NI(I1)=NI(I1)-1
00413    92.          DO 10 I2=1,LCT
00416    93.          IF(ITO(I2)-IFR(I)) 10,11,10
00421    94.     11   DO 12 I3=1,KNOF
00424    95.          IF(IBR(I3)-ITO(I2))18,19,18
00427    96.     18   IF(IBR(I3))20,20,12
00432    97.     20   IBR(I3)=ITO(I2)
00433    98.     19   IBRX(I3)=I2
00434    99.          I=I2
00435   100.          GO TO 2
00436   101.     12   CONTINUE
00440   102.     10   CONTINUE
00440   103.    C-----END OF BRANCH---SEARCH FOR NEXT BRANCH
00442   104.     25   DO 26 I5=1,KNOF
00445   105.          MI5=KNOF+1-I5
00446   106.          IF(IBR(MI5))26,26,27
00451   107.     27   DO 28 I6=1,NFI
00454   108.          IF(NFB(I6)-IBR(MI5))28,29,28
00457   109.     29   NI(I6)=NI(I6)-1
00460   110.          IF(NI(I6))30,30,22
00463   111.     30   IBR(MI5)=0
00464   112.          IDX=IBRX(MI5)+1
00465   113.          IBRX(MI5)=0
00466   114.          GO TO 31
00467   115.     22   IDX=IBRX(MI5)+1
00470   116.     31   DO 32 I7=IDX,LCT
00473   117.          IF(ITO(I7)-NFB(I6))32,33,32
00476   118.     33   IF(IBR(MI5))35,35,34
00501   119.     34   IBRX(MI5)=I7
00502   120.     35   I=I7
00503   121.          GO TO 2
00504   122.     32   CONTINUE
00506   123.     28   CONTINUE
00510   124.     26   CONTINUE
00512   125.          DO 39 INX=1,KNOF
00515   126.          IF(IRL(INX,NR))38,38,39
00520   127.     38   LL(NR)=INX-1
00521   128.          GO TO 502
00522   129.     39   CONTINUE
00524   130.          LL(NR)=KNOF
```

```
00525   131.     502  IF(IEQULR)503,503,453
00530   132.     453  B1=1.0
00531   133.          LS=LL(NR)
00532   134.          DO 454 M1=1,LS
00535   135.          LOC=IRL(M1,NR)
00536   136.          P(LOC,1,2)=EXP(P(LOC,1,1)*(T0-T1))
00537   137.     454  B1=B1*P(LOC,1,2)
00541   138.          R=3.*(B1**2.)-2.*(B1**3.)
00542   139.          PF(NR)=1.-((B1**3.)/R)
00543   140.          GO TO 619
00544   141.     503  B1=1.0
00545   142.          B2=1.0
00546   143.          B3=1.0
00547   144.          LS=LL(NR)
00550   145.          DO 504 M1=1,LS
00553   146.          LOC=IRL(M1,NR)
00554   147.          P(LOC,1,2)=EXP(P(LOC,1,1)*(T0-T1))
00555   148.          P(LOC,2,2)=EXP(P(LOC,2,1)*(T0-T1))
00556   149.          P(LOC,3,2)=EXP(P(LOC,3,1)*(T0-T1))
00557   150.          B1=B1*P(LOC,1,2)
00560   151.          B2=B2*P(LOC,2,2)
00561   152.     504  B3=B3*P(LOC,3,2)
00563   153.          R=B1*B2+B1*B3+B2*B3-2.*B1*B2*B3
00564   154.          PF(NR)=1.-((B1*B2*B3)/R)
00565   155.     619  IF(LPRINT)621,621,620
00570   156.     620  DO 623 JJ=1,LS
00573   157.     623  IBRX(JJ)=IRL(JJ,NR)
00575   158.          I1=IRST(NR)
00576   159.          B1=PF(NR)
00577   160.          WRITE(6,315)I1
00602   161.          WRITE(6,316)(IBRX(JJ),JJ=1,LS)
00610   162.          WRITE(6,317)B1
00613   163.     621  PF(NR)=1.-ABS(0.5-PF(NR))
00614   164.          IF(LPRINT)70,70,622
00617   165.     622  B1=PF(NR)
00620   166.          WRITE(6,318)B1
00620   167.    C-----MAKE IRL LIST FOR NEXT RESTORER ON LIST
00623   168.      70  CONTINUE
00623   169.    C-----ORDER LIST OF FAILURE PROBABILITIES
00625   170.          DO 403 N=1,NRT
00630   171.          COMPR=0.
00630   172.    C-----FIND HIGHEST OF REMAINING (PF)S
00631   173.          DO 402 NR=1,NRT
00634   174.          IF(COMPR-PF(NR))401,401,402
00637   175.     401  COMPR=PF(NR)
00640   176.          NINDX(N)=NR
00641   177.     402  CONTINUE
00643   178.          PORDRD(N)=COMPR
00643   179.    C-----DELETE HIGHEST PF FROM PF(NR)
00644   180.          NIX=NINDX(N)
00645   181.          PF(NIX)=0
00646   182.     403  CONTINUE
00650   183.          DO 404 N=1,NRT
00653   184.          NT=NINDX(N)
00654   185.     404  IORDRD(N)=IRST(NT)
00654   186.    C-----PRINT RESULTS
00656   187.          WRITE(6,490)(N,PORDRD(N),IORDRD(N),N=1,NRT)
00666   188.     703  GO TO 400
00667   189.     301  FORMAT(14I5)
00670   190.     305  FORMAT(2F10.0)
00671   191.     307  FORMAT(6E12.6)
00672   192.     309  FORMAT(23H1TEST POINT ALLOCATIONS///)
00673   193.     310  FORMAT(24H SYSTEM INTERCONNECTIONS//3X,47HFROM TO   FROM TO   FROM
00673   194.         1 TO   FROM TO   FROM TO/3X,47H-------   -------   -------   -----
00673   195.         2-   -------/(5(I6,I4)))
```

```
00674   196.     311  FORMAT(21H0THIS SYSTEM CONTAINS,I3,1X,9HFUNCTIONS/)
00675   197.     312  FORMAT(12H TIME ZERO =F7.0/12H TEST TIME =F7.0//)
00676   198.     314  FORMAT(19H UNIT FAILURE RATES//11X,30H----------FUNCTIONS--------
00676   199.         1--,8X,30H----------RESTORERS----------/2X,5HSTAGE,4X,6HRANK 1,6X,

00676   200.         26HRANK 2,6X,6HRANK 3,8X,6HRANK 1,6X,6HRANK 2,6X,6HRANK 3/(I5,2X,E1
00676   201.         31.6,2(E12.6),E14.6,2(E12.6)))
00677   202.     490  FORMAT(9H1TEST PT.,5X,16HVALUE OF PLACING,7X,9HLOCATE AT/8H  NUMBE
00677   203.         1R,8X,12HA TEST POINT,6X,15HRESTORER NUMBER//(I6,E23.9,I13))
00700   204.     315  FORMAT(40H1POTENTIAL TEST LOCATION AT RESTORER NO.I5)
00701   205.     316  FORMAT(14H TESTED STAGES//(5I10))
00702   206.     317  FORMAT(40H PROBABILITY THAT A FAILURE IS OBSERVED=E11.6)
00703   207.     318  FORMAT(36H VALUE OF PLACING A TEST POINT HERE=E11.6)
00704   208.          END

      END OF LISTING.      0 *DIAGNOSTIC* MESSAGE(S).
PHASE 1 TIME =    1 SEC.
PHASE 2 TIME =    0 SEC.
PHASE 3 TIME =    1 SEC.
PHASE 4 TIME =    0 SEC.
PHASE 5 TIME =    0 SEC.
PHASE 6 TIME =    1 SEC.


TOTAL COMPILATION TIME =    3  SEC
```

APPENDIX B

RELIABILITY ANALYSIS PROGRAM

USER'S INSTRUCTION MANUAL

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

I. INTRODUCTION

The reliability analysis program calculates an estimate of the reliability of a modularly redundant system, using the results obtained from a limited number of test points. It provides an extremely simple means of determining the probability that a system will be functionally operational at the end of a given time period. The program is completely compatible with the test point allocation program described in Appendix A. The failed/working information generated by the test points allocated by the latter program is used directly in the reliability estimation program.

The program is written in FORTRAN IV; it is therefore highly machine-independent.

A. WHAT THE PROGRAM DOES

The reliability analysis program computes an estimate of the reliability of a triple-modularly redundant digital system, given that the system is functionally operational at the time of test, and given the failed/working results of a limited number of test points.

The reliability estimate produced by the program is based on the Block Model technique described in section II, part B of the body of this report. In order to include test information in the estimate, a modified version of the technique has been developed for use in the program. The modified technique is also described in the body of this report, in section II, part C.

The program uses three distinct time variables: T_0, T_1, and T_M, as illustrated in figure B-1. It is assumed that, at T_0, the system is completely failure-free; i.e., that every system component is operational. The time period of interest to the user, i.e., the mission time, is the period from T_0 to T_M. The limited testing is performed at T_1, some time between T_0 and T_M. It is further assumed, as mentioned above, that the system is functionally operational at T_1; i.e. that at least two of the three subsystems in every stage are operational at test time.

In addition to the reliability estimate based on tests performed at T_1, the program contains an option for computing an estimate of the initial reliability at T_0, with no test data, but with the assumption that all subsystems are operational.

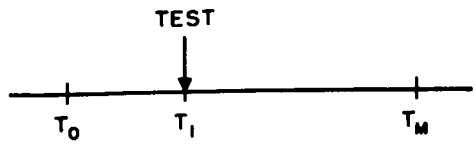


Figure B-1. The Time Variables Used

B. SYSTEM CONFIGURATION LIMITATIONS

The reliability analysis program can be applied only to majority-voted redundant digital systems. The systems must, in addition, be order-three redundant; i.e., all stages, including voter stages, must be triplicated. Within these limitations, there are no restrictions on the configuration of the system. Virtually any degree of system complexity can be handled by the program. The size of the system which can be analyzed by the program is limited only by computer running time available, and individual computer storage limitations.

C. TEST POINT INFORMATION REQUIRED

All of the test points which produce information usable by the reliability analysis program are located at the inputs of restorers.

In the program, therefore, the location of each test point is specified by the restorer at whose input the test point is located. The test result from each of these test points is read into the program in the form of the rank, if any, which is observed as incorrect. Since the system is assumed to be functionally operational at the time of the test, only one rank can be failed at each test point location.

A summary flow chart of the program is shown in figure B-2. A detailed flow chart appears in section V of this appendix.

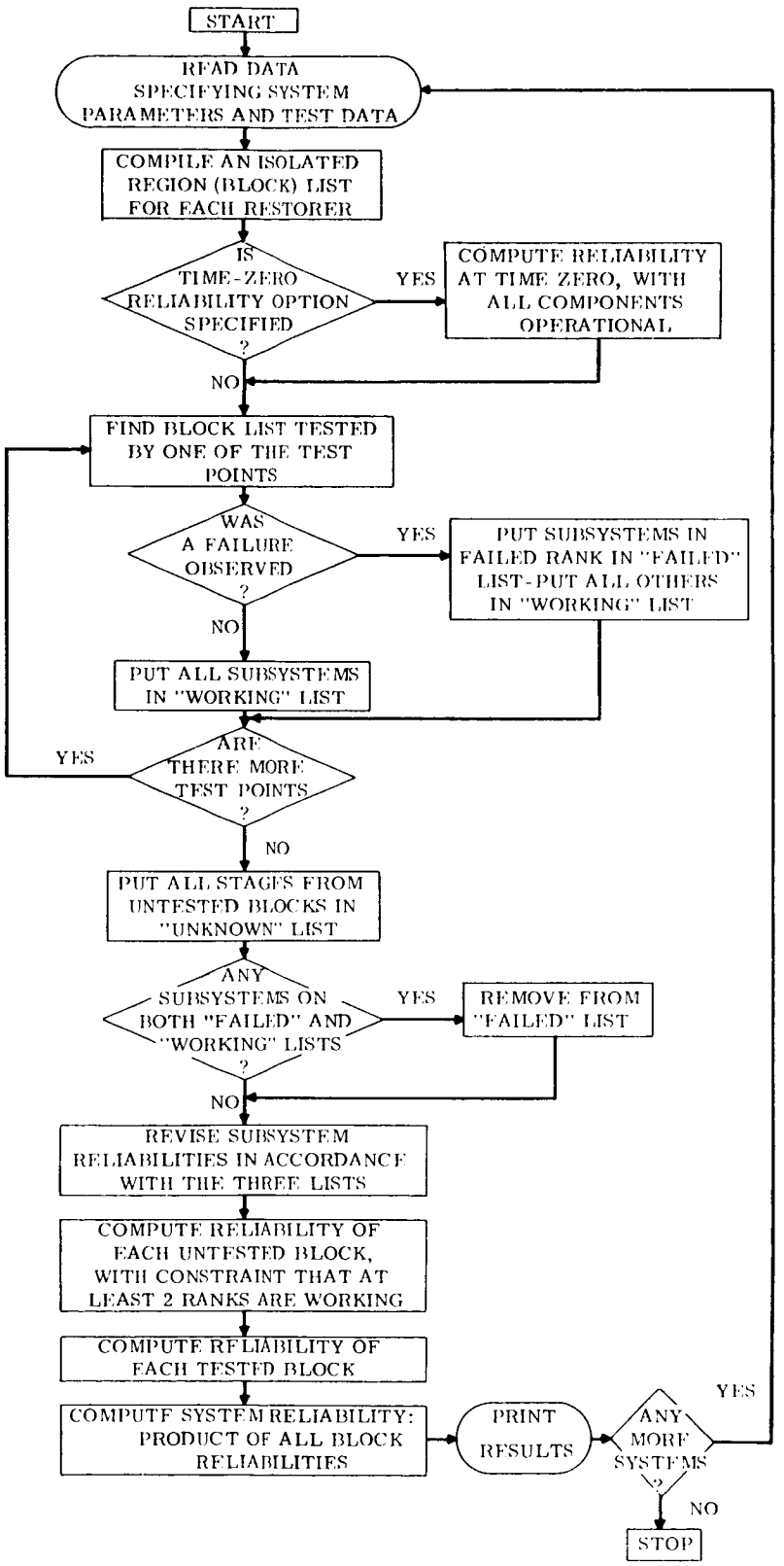


Figure B-2. Summary Flow Diagram of the Reliability Analysis Program

II. HOW TO USE THE PROGRAM

The system to be analyzed is first converted to simple block diagram form. A system model is then constructed which is very similar to the block diagram. Finally, an ID number is assigned to each stage of the system, according to a procedure described below. The system model thus completed, the exact system configuration can be read into the program. The necessary input data describing the system is compiled directly from the system model.

The following paragraphs describe, first, the construction of the system model, and secondly, the input data required by the program.

A. SYSTEM MODEL

If the system to be analyzed has first had test points allocated to it by the test point allocation program, the same system model used for that program can be used for the reliability analysis program. If not, this section describes the construction of the model.

The first step in the construction of the system model is the conversion of the system to block diagram form. Each of the "boxes" in the block diagram indicates either a function stage or a restorer stage. The usual procedure is to use boxes to indicate the function stages, and circles to indicate restorer stages. All interconnections between stages are shown in the diagram, as well as the location of inputs from outside the system and outputs to external equipment. A failure rate is then assigned to each of the three units, or subsystem in every stage.

1. Artificial Stages

The next step in the modeling procedure is the addition to the block diagram of artificial stages. There are two places in the system at which they might be added: at all of the system inputs and at some of the system outputs. The following paragraphs describe all of the situations in which these artificial stages are used.

a) Artificial Input Stages

The purpose of placing artificial stages at system inputs is to shorten the program running time, by eliminating unnecessary table searching.

Figure B-3 is a block diagram of a simple system to which the program could be applied. The two system inputs are labeled A and B. An artificial stage is added to both in figure B-4. The input labels A and B in figure B-3 have been replaced by the two artificial input stages.

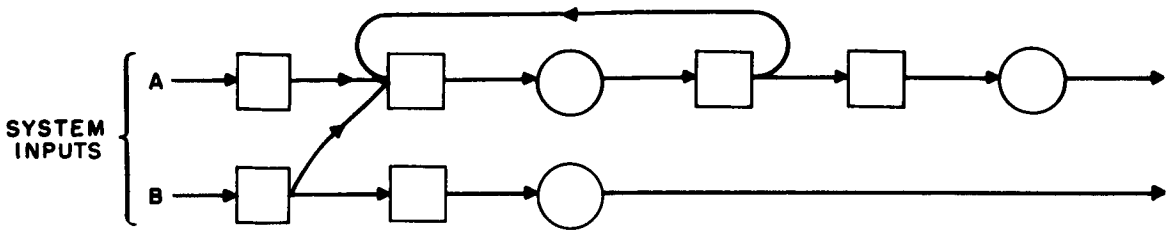


Figure B-3. System Block Diagram, Input Stages Needed.

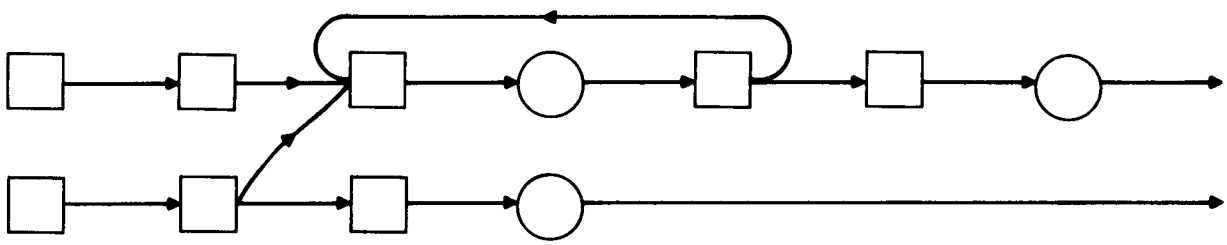


Figure B-4. Input Stages Added to Figure B-3.

The addition of these input stages completes the model for the system shown. Any failure rate, including zero, can be applied to the artificial inputs because the program does not include these failure rates in any calculations.

b) Artificial Output Stages

The first place at which an artificial output stage must be added is at every system output which does not come directly from a restorer. This situation is shown in figure B-5. The two system outputs are labeled A and B. Output B is not restored, so an artificial restorer stage must be added, as shown in figure B-6.

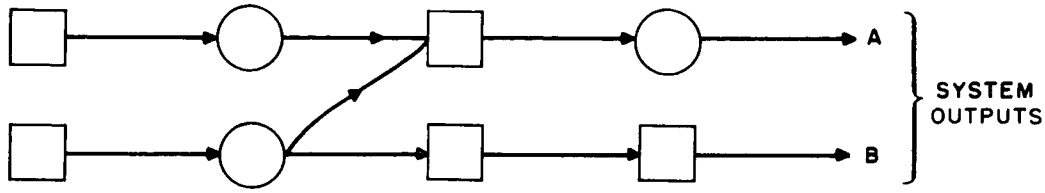


Figure B-5. Artificial Output Stage Needed.

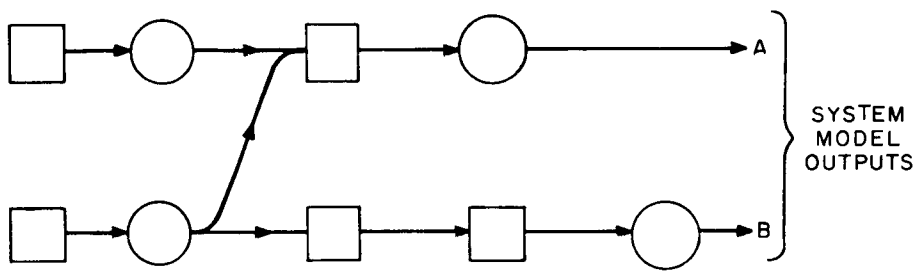


Figure B-6. Output Stage added to Output B of Figure B-5.

There is one other situation in which an output stage must be added. This occurs when the output of an internal restorer is also a system output, and a test point is located at this output. Since the program considers test points placed only at restorer inputs, the system model must be altered. The block diagram shown in figure B-7 illustrates this case. Assume that the user has placed a test point at output A, as shown by the X in the figure. Since the program considers only restorer inputs as test point locations, an artificial restorer stage must be added at point A. The system model permits restorers to be placed only at the outputs of function stages, however, so an artificial function stage must be inserted between the system restorer and the artificial output restorer. The units in this function stage must be assigned zero failure rates, so that they do not affect the system

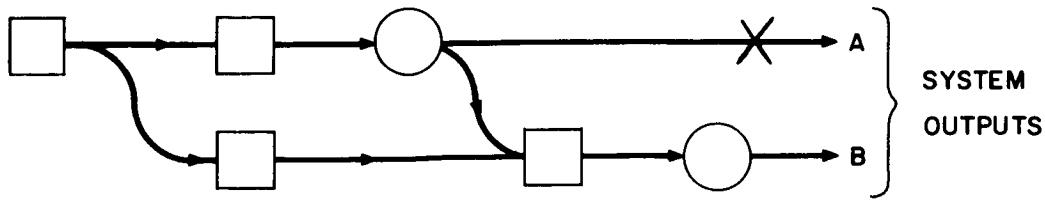


Figure B-7. Test Point at Output A.

reliability calculatons. The failure rates for the artificial restorer stage will not enter into any reliability calculations, so their values are unimportant.

2. Stage ID Numbers

The final step in the construction of the system model is the assignment of an ID number to every stage in the system. The procedure is as follows. First, the function stages, excluding artificial input stages, are numbered from 1 to N, where N is the total number of actual function stages. The restorer stages are then numbered in the following manner. The restorer of function i is assigned an ID number of N + i. The resistor of function 1 is assigned an ID number of N + 1, and the restorer of function N has an ID number of 2N, etc. This numbering procedure enables the program to distinguish between functions and restorers. Finally the artificial input stages are assigned ID numbers of 1,000 or greater, in any order. This enables the program to recognize system inputs when tracing signal paths, thereby eliminating much unnecessary table searching. (For systems containing a total number of stages, excluding artificial input stages, of 1,000 or greater, the number 1,000 may be raised in the program by changing the instruction on lines 55 and 96 of the program listing.)

The numbering of the system model stages in the above manner completes the construction of the model. The program input data defining the system is taken directly from the resulting model.

B. INPUT DATA CARDS REQUIRED

This section lists the necessary input data cards for the reliability analysis program. The cards are listed in the order in which they must appear in the data deck. Corresponding FORMAT statements and READ specifications for each item are also shown. Each of the following paragraphs describes one data card or group of data cards. (NOTE: The FORMAT (14I5) is used to read many of the data cards which have less than 14 items. It is simply a generalized FORMAT, used to reduced the number of separate FORMAT statements.)

LCT, NOF. FORMAT (14I5). READ LCT, NOF.

LCT is the length of the connection table, i.e., the total number of interconnections in the system model. NOF is the total number of functions in the system model, excluding restorers.

IFR. FORMAT (14I5) READ (IFR (IT), IT = 1, LCT).

IFR is the list of stages which provide inputs to other stages in the system model. These stages include any artificial input stages. The position of each entry in IFR must be exactly the same as the position of the corresponding stage in the ITO list, which receives the input.

ITO. FORMAT (14I5). READ (ITO (IT), IT = 1, LCT).

ITO is the list of stages which receive outputs from other stages in the system model. These stages include any artificial output stages in the model. The position of each entry in ITO must be exactly the same as the position of the corresponding stage in the IFR list, which provides the input.

TO, T1, TM. FORMAT (3F10.0). READ TO, T1, TM.

TO is the time zero of the system model; i.e., the time at which all system components are assumed to have been operational. T1 is the time of the test. TM is the end-of-mission time. The units used for all times must, of course, correspond to that used for the unit failure rates.

ITZREL. FORMAT (14I5). READ ITZREL.

This is the time-zero-reliability flag, which specifies whether the program will compute the system reliability estimate at time zero, without test data (ITZREL = 1), or not compute it (ITZREL = 0).

NTP. FORMAT (14I5). READ NTP.

This is the total number of test points applied to the system.

ITEST. FORMAT (14I5). READ ((ITEST (NR, NFR), NFR - 1,2), NR = 1, NTP).

This is the test results data. The data consists of a list of the restorers that have test points at their inputs, together with a list of the rank observed as failed at each of these locations. The list of test point locations as read in, is alternated with the list of failed ranks; i. e., each restorer number is followed by the rank observed as failed at that location.

IEQULR. FORMAT (14I5). READ IEQULR.

This equal reliability flag indicates, if it equals one, that all three units in every stage have equal failure rates. If different failure rates are assigned to the units of any stage, IEQULR is made zero and a separate failure rate is read in for every unit in the system.

P (IST, 1, 1). FORMAT (6E 12. 6). READ (P (IST, 1, 1), IST = 1, KNOF).

This is the list of failure rates for the system units. One failure rate is read in for each stage in the system, and this value is used for the three units in the stage. The failure rates for the function stages are read in first, followed by those of the restorer stages. For the case of a function which is not restored, the field corresponding to the appropriate restorer number may be left blank. The failure values for non-present voters is not used by the program. This READ stagement is used only when IEQULR = 1. If IEQULR = 0, the following read statement is used.

P(IST, IRK, 1). FORMAT (6E12. 6). READ (P (IST, IRK, 1), IRK = 1,3), IST = 1, KNOF).

This is the list of failure rates for the system units. One failure rate is read in for each unit in the system. The failure rates of the three units in a stage are read in consecutively, and values are read in according to stage ID numbers; i. e. function stages first, followed by restorer stages. This READ statement is used only when IEQULR = 0. If IEQULR = 1, the preceding read statement is used.

NOV. FORMAT (14I5). READ NOV.

This constant represents the number of output voters whose stage reliability estimates are to be included in the overall system reliability estimate.

NVR (NV). FORMAT (14I5). READ (NVR (NV), NV = 1, NOV).

This is the list of ID numbers for output voter stages whose reliability estimates are to be included in the system reliability estimate. This READ statement is used only when NOV is non-zero.

LPRINT. FORMAT (14I5). READ LPRINT.

This is the printout option flag, which specifies which of two printout modes will be employed. When LPRINT= 0, the "normal operation" mode is used, in which the program prints only a listing of input data and final results of the analysis. When LPRINT= 1, the "debug operation" is employed, in which the program prints the above information, plus many of the intermediate computational results.

III. OUTPUT TO BE EXPECTED

This section outlines the printout to be expected from the test point allocation program. The first part of the section describes the output obtained during normal operation of the program. Following this is a description of the output available during possible debugging operations, providing the user with a more extensive view of the intermediate computational results. An input data constant, LPRINT, specifies which of the output options will be used in any one program run.

A. NORMAL OPERATION

The normal operation printout mode is specified by setting the input data constant, LPRINT, to zero. In this mode, the program prints, first, a listing of the input data specifying the system which has been analyzed, and, secondly, the results of the analysis.

The first item to be printed is a complete list of the system interconnections. This listing is followed by the statement of the total number of system function stages, then the zero time of the system, TO, the time of test, T1, and the time of the mission end, TM.

The test data used in the reliability estimation are then printed. This is a two column list. The first column contains the ID number of each restorer at which a test point is located. The second column lists the rank that is observed as failed at each of the test locations. As mentioned earlier, this item is zero if no erroneous signal is observed at the test point.

Next, the failure rate of each unit, or subsystem, is listed. The failure rates of the units in each restorer are listed on the same line as those of the corresponding function stage.

The printout of the results of the analysis consists of the statement of the estimate of the system reliability at test time. In addition, if the user has specified the calculation of the initial, time zero reliability estimate, this value is printed.

A sample of the printout obtained during the normal operation mode is shown on the following page.

```
POST-TEST RELIABILITY ANALYSIS

SYSTEM INTERCONNECTIONS

  FROM TO    FROM TO    FROM TO    FROM TO    FROM TO
  -------    -------    -------    -------    -------
 1000   1      1   6      6   2      6   5      5   3
    5   4      2   7      3   8      4   9

THIS SYSTEM CONTAINS  5 FUNCTIONS

TIME ZERO =        0.
TEST TIME =       50.
MISSION TIME =   100.
TEST DATA
  RESTORER     RANK
   TESTED     FAILED
      7          0
      8          1

UNIT FAILURE RATES
           ----------FUNCTIONS----------          ----------RESTORERS----------
 STAGE     RANK 1       RANK 2       RANK 3      RANK 1       RANK 2       RANK 3
   1    .100503-03   .100503-03   .100503-03   .100503-03   .100503-03   .100503-03
   2    .100503-03   .100503-03   .100503-03   .100503-03   .100503-03   .100503-03
   3    .100503-03   .100503-03   .100503-03   .100503-03   .100503-03   .100503-03
   4    .100503-03   .100503-03   .100503-03   .100503-03   .100503-03   .100503-03
   5    .100503-03   .100503-03   .100503-03   .100503-03   .100503-03   .100503-03

BEFORE TEST, SYSTEM MISSION RELIABILITY (TIME ZERO TO MISSION END)=   .992469-00

AFTER TEST, SYSTEM MISSION RELIABILITY (TEST TIME TO MISSION END)=   .954256-00
```

B. DEBUG OPERATION

The debug operation printout mode is specified by setting the input data constant, LPRINT, to one. In this mode, the program prints all of the information provided by the normal operation mode. In addition, further information is printed to enable a user to examine some of the intermediate computational operations in greater detail.

The first addition item printed, following the input data listing, is the storage block IUNKN. This is a list of all of the stages which are in untested blocks; i.e., stages which are failure - linked to untested restored functions. In this two column listing, the first column lists every stage in the system; the second column contains a 1 (one) if the corresponding stage is in an untested block, and a 0 (zero) if a stage is not in an untested block.

The next listing is the storage block IWORK. This is a list of all subsystems which the test data indicate are operational at test time. The subsystems are listed by stage and rank. A 1 (one) indicates that a given subsystem is definitely operational at test time, and a 0 (zero) indicates either that the subsystem is failed or that the operational state cannot be determined from the data.

The block IFAIL is printed next. In this listing, a non-zero entry specifies that the subsystem is in the failed rank of a tested block. The value of the entry is equal to the number of tested-failed ranks in which the subsystem is located.

The next listing contains the two probabilities which together form the reliability estimate of each subsystem in the system. The first is the probability that a given subsystem is operational at test time. The second is the conditional probability that the subsystem will be operational at the mission end, given that it was operating at test time. The product of these is the subsystem reliability estimate used in the estimation of the system reliability.

Following this are the lists of stages which form each of the untested blocks. Each block list is followed by the product of the reliabilities of the blocks listed.

The final additional items printed during the debug operation mode are the lists of stages which form each of the tested blocks in the system. Each list is followed by the product of the reliabilities of the blocks listed at that point. The final system reliability estimate is the product of the untested block reliability estimates and the tested block reliability estimates.

A sample of the printout obtained during the debug operation mode is shown.

```
POST-TEST RELIABILITY ANALYSIS


SYSTEM INTERCONNECTIONS

  FROM TO    FROM TO    FROM TO    FROM TO    FROM TO
  -------    -------    -------    -------    -------
 1000   1      1   6      6   2      6   5      5   3
    5   4      2   7      3   8      4   9

THIS SYSTEM CONTAINS  5 FUNCTIONS

TIME ZERO =        0.
TEST TIME =       50.
MISSION TIME =   100.


TEST DATA
  RESTORER     RANK
   TESTED     FAILED
      7          0
      8          1

UNIT FAILURE RATES

          ----------FUNCTIONS---------          ----------RESTORERS----------
STAGE     RANK 1       RANK 2       RANK 3      RANK 1       RANK 2       RANK 3
  1    .100503-03   .100503-03   .100503-03   .100503-03   .100503-03   .100503-03
  2    .100503-03   .100503-03   .100503-03   .100503-03   .100503-03   .100503-03
  3    .100503-03   .100503-03   .100503-03   .100503-03   .100503-03   .100503-03
  4    .100503-03   .100503-03   .100503-03   .100503-03   .100503-03   .100503-03
  5    .100503-03   .100503-03   .100503-03   .100503-03   .100503-03   .100503-03
```

STAGES IN UNTESTED BLOCKS

STAGE	1=YES,0=NO
1	1
2	0
3	0
4	1
5	1
6	1
7	0
8	0
9	0
10	0

UNITS OBSERVED AS OPERATIONAL

STAGE	--RANK---		
	1	2	3
1	0	0	0
2	1	1	1
3	0	1	1
4	0	0	0
5	0	1	1
6	1	1	1
7	0	0	0
8	0	0	0
9	0	0	0
10	0	0	0

UNITS IN TESTED FAILED RANKS

STAGE	--RANK---		
	1	2	3
1	0	0	0
2	0	0	0
3	1	0	0
4	0	0	0
5	1	0	0
6	0	0	0
7	0	0	0
8	0	0	0
9	0	0	0
10	0	0	0

SUBSYSTEM PROBABILITIES

	PROBABILITY OF OPERATION AT T1			PROBABILITY OF OPERATION AT TM, GIVEN THAT OPERATIONAL AT T1		
STAGE	RANK 1	RANK 2	RANK 3	RANK 1	RANK 2	RANK 3
1	.994987	.994987	.994987	.994987	.994987	.994987
2	1.000000	1.000000	1.000000	.994987	.994987	.994987
3	.000000	1.000000	1.000000	.000000	.994987	.994987
4	.994987	.994987	.994987	.994987	.994987	.994987
5	.498744	1.000000	1.000000	.994987	.994987	.994987
6	1.000000	1.000000	1.000000	.994987	.994987	.994987
7	.994987	.994987	.994987	.994987	.994987	.994987
8	.994987	.994987	.994987	.994987	.994987	.994987
9	.994987	.994987	.994987	.994987	.994987	.994987
10	.994987	.994987	.994987	.994987	.994987	.994987

UNTESTED BLOCKS RELIABILITY

UNTESTED BLOCK NO. 1 CONTAINS THE FOLLOWING STAGES

1

PRODUCT OF RELIABILITIES OF ALL UNTESTED BLOCKS ABOVE= .999777-00

UNTESTED BLOCK NO. 2 CONTAINS THE FOLLOWING STAGES

4
5
6

PRODUCT OF RELIABILITIES OF ALL UNTESTED BLOCKS ABOVE= .984417-00

TESTED BLOCKS RELIABILITY

TESTED BLOCK NO. 1 CONTAINS THE FOLLOWING STAGES

2
6

PRODUCT OF RELIABILITIES OF ALL TESTED BLOCKS ABOVE= .999702-00

TESTED BLOCK NO. 2 CONTAINS THE FOLLOWING STAGES

3
5
6

PRODUCT OF RELIABILITIES OF ALL TESTED BLOCKS ABOVE= .970010-00

BEFORE TEST, SYSTEM MISSION RELIABILITY (TIME ZERO TO MISSION END)= .992469-00

AFTER TEST, SYSTEM MISSION RELIABILITY (TEST TIME TO MISSION END)= .954256-00

IV. PROGRAM VARIABLES AND CONSTANTS

This section contains a list of important program variables and constants with a brief explanation of each item. It is intended as an aid to the investigator who requires a more detailed description of the program than is provided by a program user's manual.

IBR — Subscripted variable - This array is used during the construction of the block lists. It stores the ID numbers of stages with multiple inputs, in the order in which they are encountered during the tracing of a signal path. When the beginning of a path is reached, the program goes to the last entry in IBR, which is the ID number of the last branch location passed. The program then traces the next branch with an input to this location.

IBRX — Subscripted variable - This array is used to store the locations in the connection table of the branch locations stored in IBR. This connection table location is used as a starting point in the search for the next branch with an input to the fan-in stage.

IEQULR — Non-subscripted constant - may have one of two values:

0 - a separate failure rate is read in for each unit in every stage.

1 - a failure rate is read in for each stage, and the value is used as the failure rate for each of the three units in the stage.

IFAIL — Subscripted variable - This array is used to store the locations of all units which appear in the failed rank of a tested block. There is a one-to-one correspondence between the locations in IFAIL and the locations in the system. When a unit in a given system location is found in a failed rank, the corresponding location in IFAIL is increased by one.

IFR — Subscripted constants - This array is the "from" list of the connection table. It stores the ID numbers of the stages which provide inputs to other stages in the system model. A given system connection will initiate two entries in the connection table: the stage providing the output will be entered in the "from" list, IFR, and the stage receiving this output will be entered the same location in the "to" list, ITO.

LPRINT Non-subscripted constant - This is the printout option flag, which specifies which of two printout options will be employed. LPRINT may have one of two values:

0 - "Normal operation" mode. The program prints a listing of input data specifying the system analyzed, plus the final analysis results.

1 - "Debug operation" mode. In addition to the printout obtained in the normal mode, the program prints many of the intermediate computational results.

IRL Subscripted variable - This matrix stores the completed block lists, after they are completed. IRL stores one complete list for each restorer and system output.

IRST Subscripted variable - This array holds the complete list of restorer ID numbers. The list is compiled from the connection table, by searching through the table for ID numbers which are greater than NOF, the number of functions and less than 1,000.

ITEST Subscripted constants - This array stores the test point data. There are two entries for each test point: The first entry is the restorer before which the test point is located; the second entry is the rank that was observed as failed. If a given test point indicates no failure, the rank number is zero.

ITO Subscripted constants - This array is the "to" list of the connection table. It stores the ID numbers of stages which receive outputs of other stages in the system model. See IFR, in this section.

ITZREL Non-subscripted constant - This is the Time - Zero - Reliability flag. It may have one of two values:

0 - The program does not compute the initial system reliability.

1 - The program computes the initial reliability of the system at time zero, assuming that all units are operational.

IUNKN Subscripted variable - This matrix holds a list of stages which are included in untested blocks. IUNKN is initially set to zero, and ones are inserted in appropriate locations in accordance with the test data and the block lists. There is a one-to-one correspondence between the stage ID numbers and the IUNKN locations.

IUSD Subscripted variable - This array is used in the generation of a list of subsystems which have multiple inputs (fan-in units). This latter list is used in the construction of the block lists. It is constructed by searching through the connection table for stages with two or more inputs. When a stage ID number is put in the

fan-in list NFB, then the corresponding location in IUSD is changed from zero to one. This assures that the program does not examine the connection table entry repeatedly. There is a one-to-one correspondence between connection table locations and IUSD locations.

IWORK — Subscripted variable - IWORK stores the locations in the system model of units which are in working ranks of tested blocks. There is a one-to-one correspondence between the stage and rank locations in the model, and the IWORK locations. When a subsystem is observed to be in the working rank of a tested block, the corresponding location in IWORK is changed from zero to one.

KNOF — Non-subscripted constant - KNOF is equal to twice the number of functions in the system model (KNOF = 2 X NOF). This is the total number of stages in a system with NOF functions, all of which are restored.

LCT — Non-subscripted constant - LCT is the Length of the Connection Table, the total number of connections in the system model.

LCT1 — Non-subscripted constant - LCT1 is equal to LCT - 1. It is used as a DO index limit for table searching.

LL — Subscripted variables - LL stores the lengths of the block lists stored in IRL.

NFB — Subscripted variable - This array holds a list of fan-ins; i.e., stages having two or more inputs. NFB is used for locating branches during the construction of block lists. It is compiled by a comparison of entries in the connection table.

NFI — Non-subscripted variable - NFI is the total number of stages in the system model which have two or more inputs. The value of NFI is therefore the length of the fan-in list, NFB.

NI — Subscripted variable - This array stores the number of inputs to each of the stages on the fan-in list, NFB. There is a one-to-one correspondence between entries in NFI and those in NFB.

NOF — Non-subscripted constant - NOF is the total Number of Functions in the system model.

NOV — Non-subscripted constant - NOV is the total Number of Output Voters whose reliability estimates are to be included in the system reliability estimate. Since the blocks consist only of inputs to voters, any output voters will not be included in the estimation of system reliability. NOV indicates to the program, if non-zero, that there are NOV voters whose reliability estimates must be included separately.

NR - Non-subscripted variable - NR is used as a DO loop index. Its value at a given time represents the location in the voter list of the voter whose block list is being utilized.

NRT - Non-subscripted variable - NRT is equal to the total number of restorers in the system model. Its value is computed during the construction of the voter list, IRST.

NTP - Non-subscripted constant - equal to the totalNumber of Test Points in the test point list, ITEST.

NVR - Subscripted constants - This is a list of output voters whose reliability estimates are to be included in the system reliability estimate. This list is read into the program.

P - Subscripted variables - This array stores:

1. Unit failure rates.
2. Probability that a unit is working at test time.
3. Probability that a unit will operate until mission end, given that it was working at test time.

The first item is read in. The second and third items are computed from the first, in accordance with the test data. There are three locations in P for every unit, or subsystem, in the system model.

PP - Subscripted variables - This array stores the probabilities that each rank in every block is working at test time. There are two locations available for each rank in each block. The first location the probability without test data. The second location stores the probability with test data, if it is necessary to change the probability.

PTSTD - Non-subscripted variable - PTSTD stores the reliability estimate for all tested blocks. It is made up of the product of the tested blocks' reliability estimates.

PUNT - Non-subscripted variable - PUNT stores the reliability estimate for all untested blocks. It is made up of the product of the untested blocks' reliability estimates.

R - Non-subscripted variable - R is the initial reliability of the system at time zero, without the inclusion of test information. In the calculation of this value, it is assumed that all subsystems are operational.

REL - Non-subscripted variable - REL is the system reliability estimate, calculated at test time, and based on the test data. REL is the product of PTSTD and PUNT.

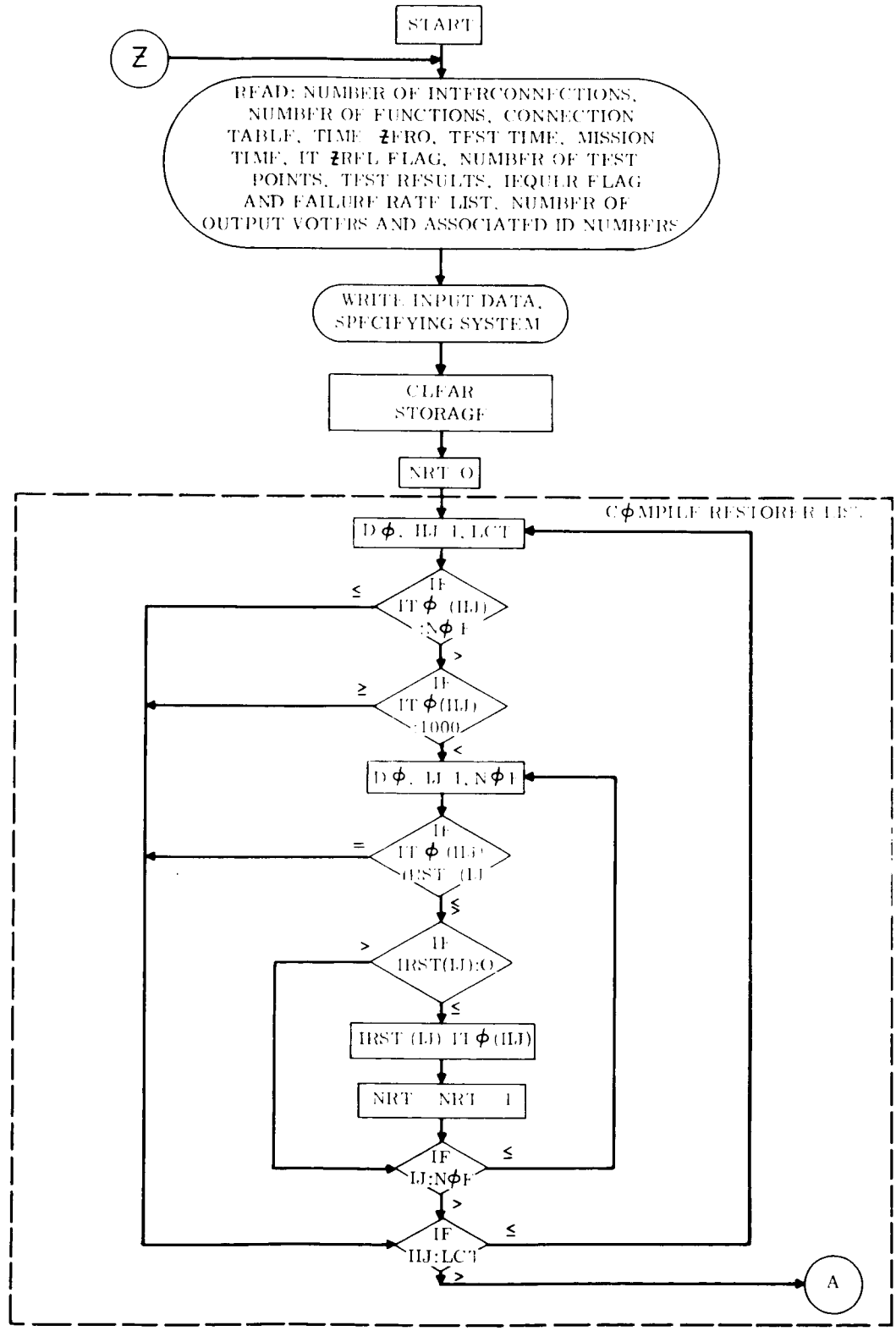
TM - Non-subscripted constant - TM is the time of the mission end. The units of TM must be the same as the units of T1, T0, and the subsystem failure rates.

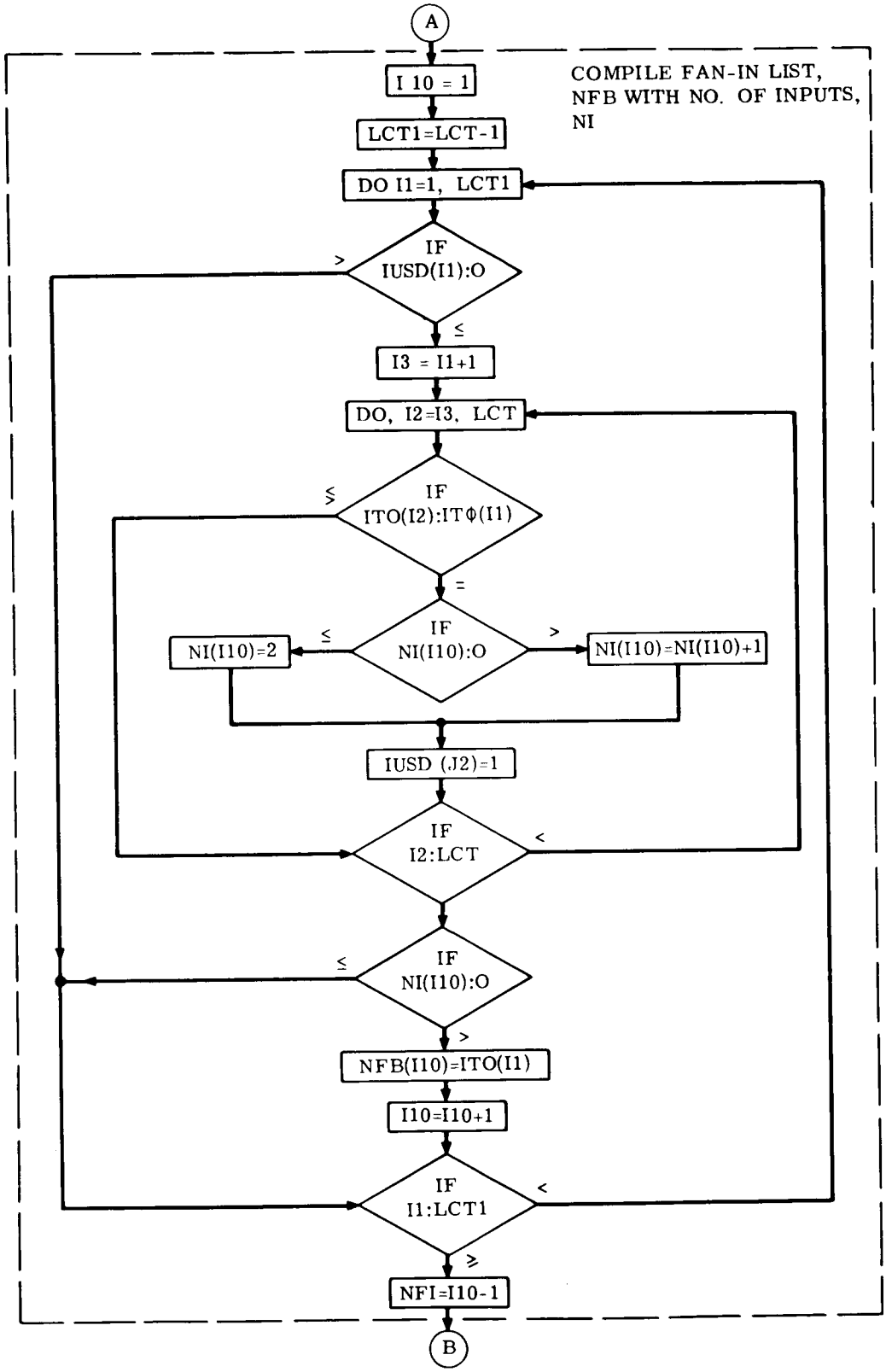
T0 - Non-subscripted constant - T0 is the time zero of the system; i.e. the time at which all subsystems are assumed to have been operational. The units used for T0 must be the same as those of T1, TM, and the subsystem failure rates.

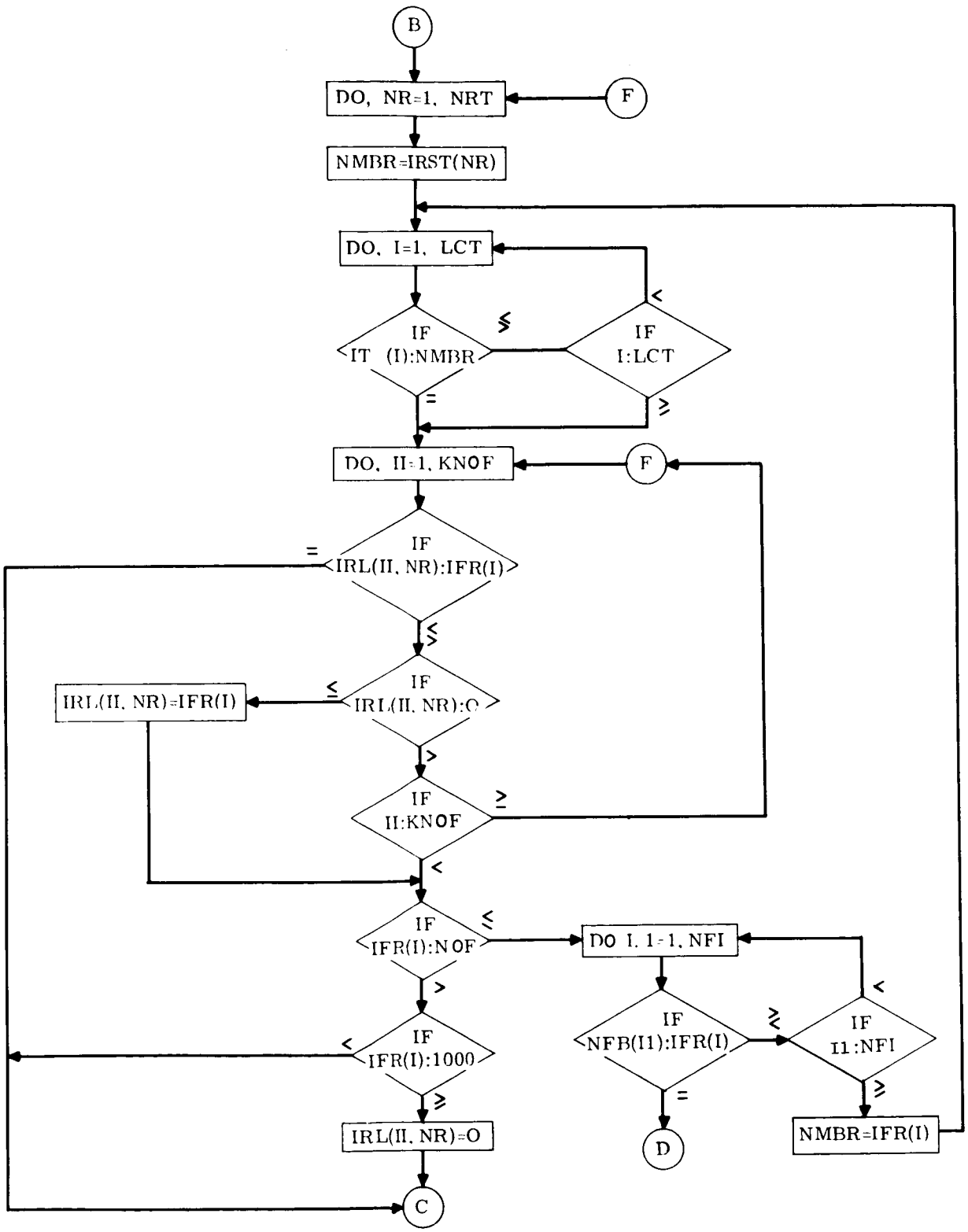
T1 - Non-subscripted constant - T1 is the time of test. The units used for T1 must be the same as those of T0, TM, and the subsystem failure rates.

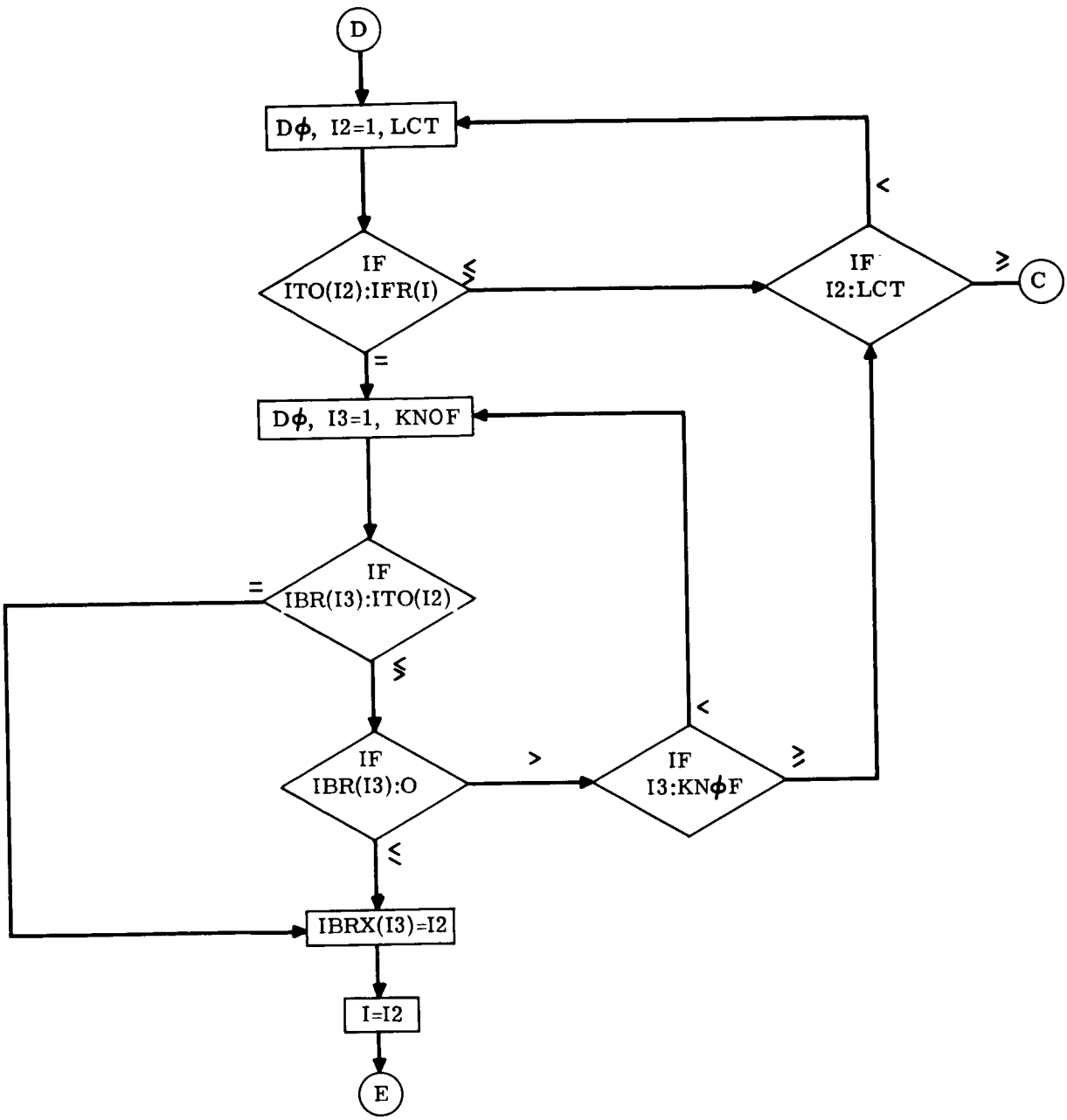
V. PROGRAM FLOW CHART

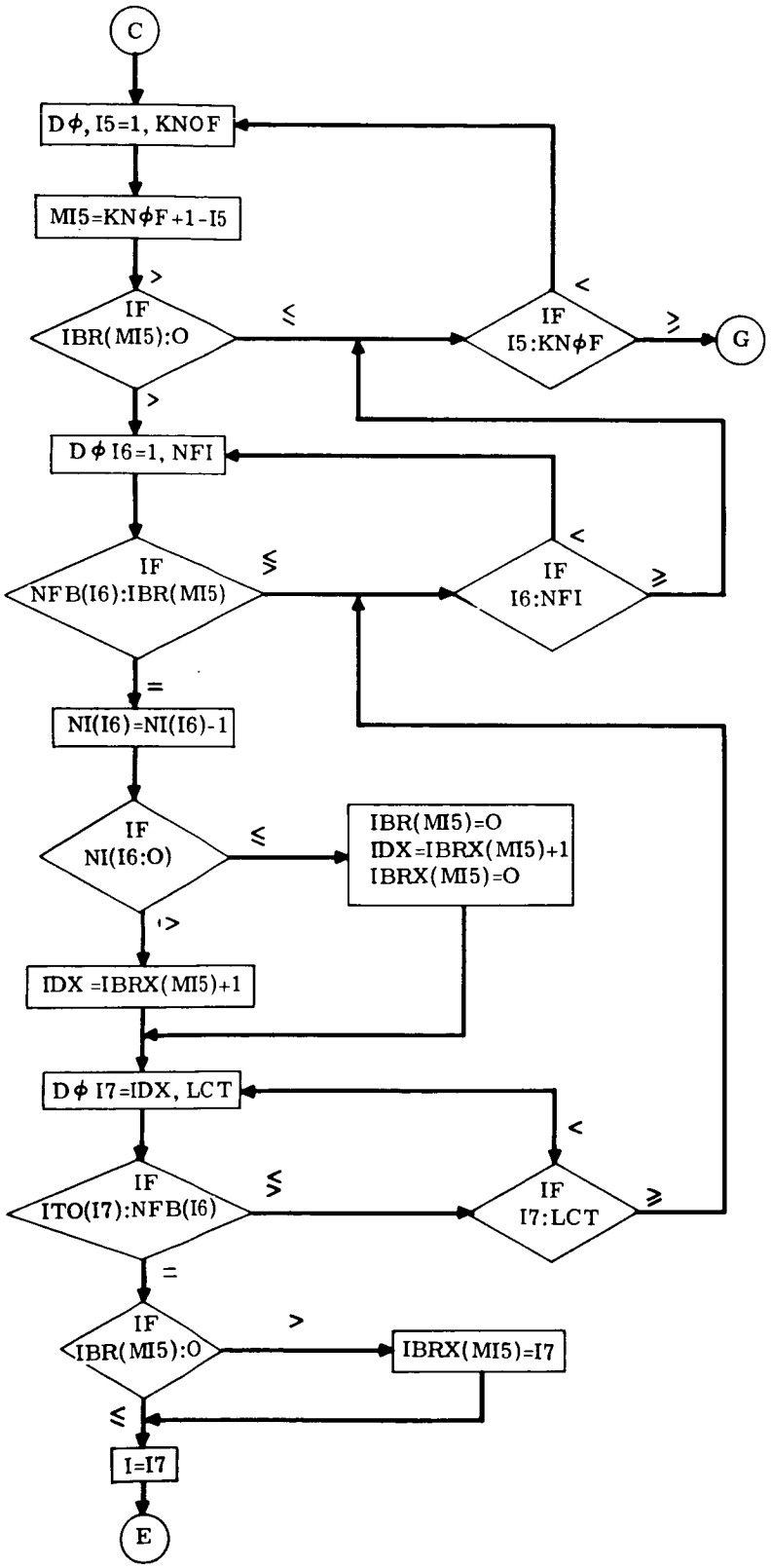
The following pages contain a detailed flow chart of the reliability analysis program.

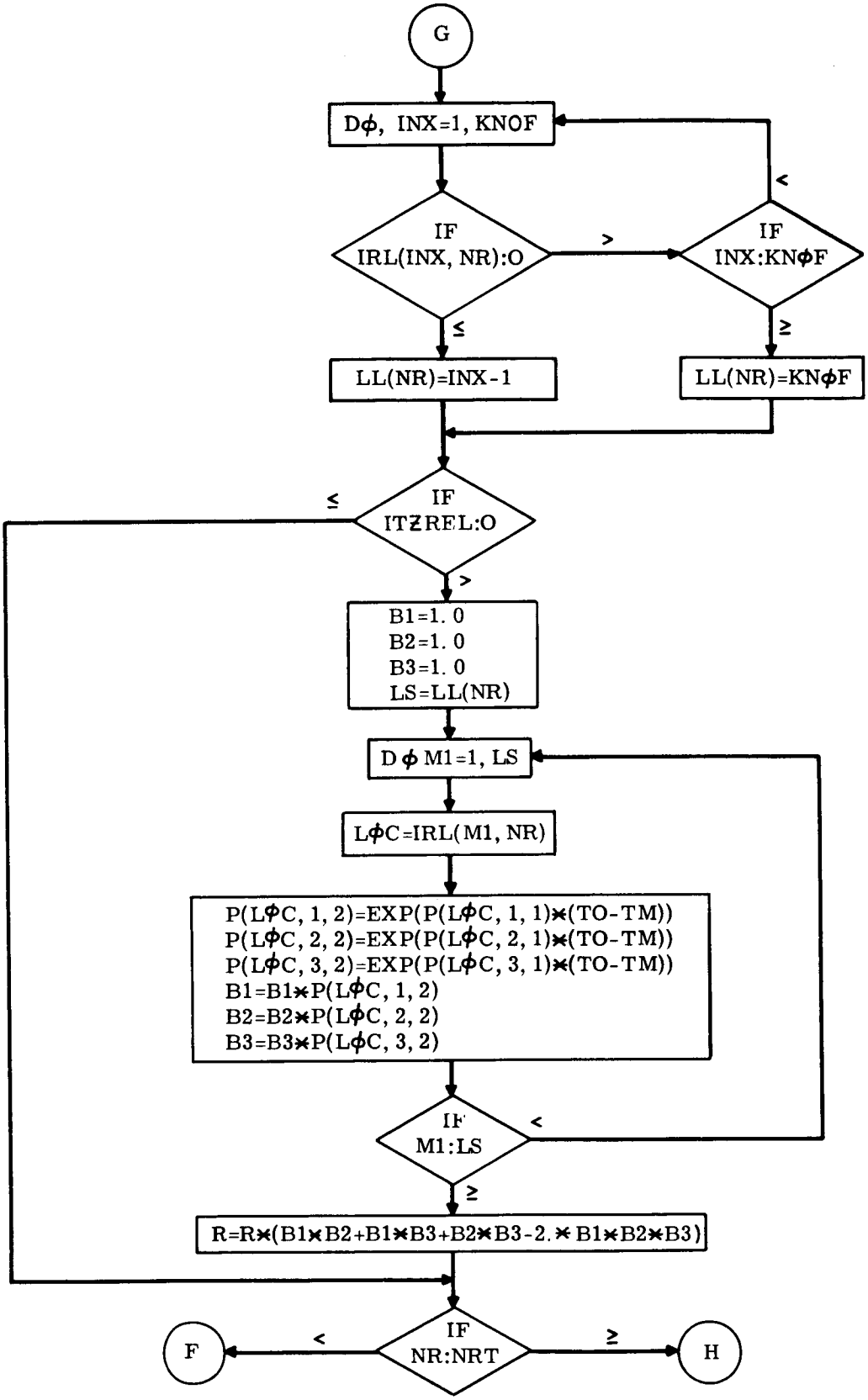


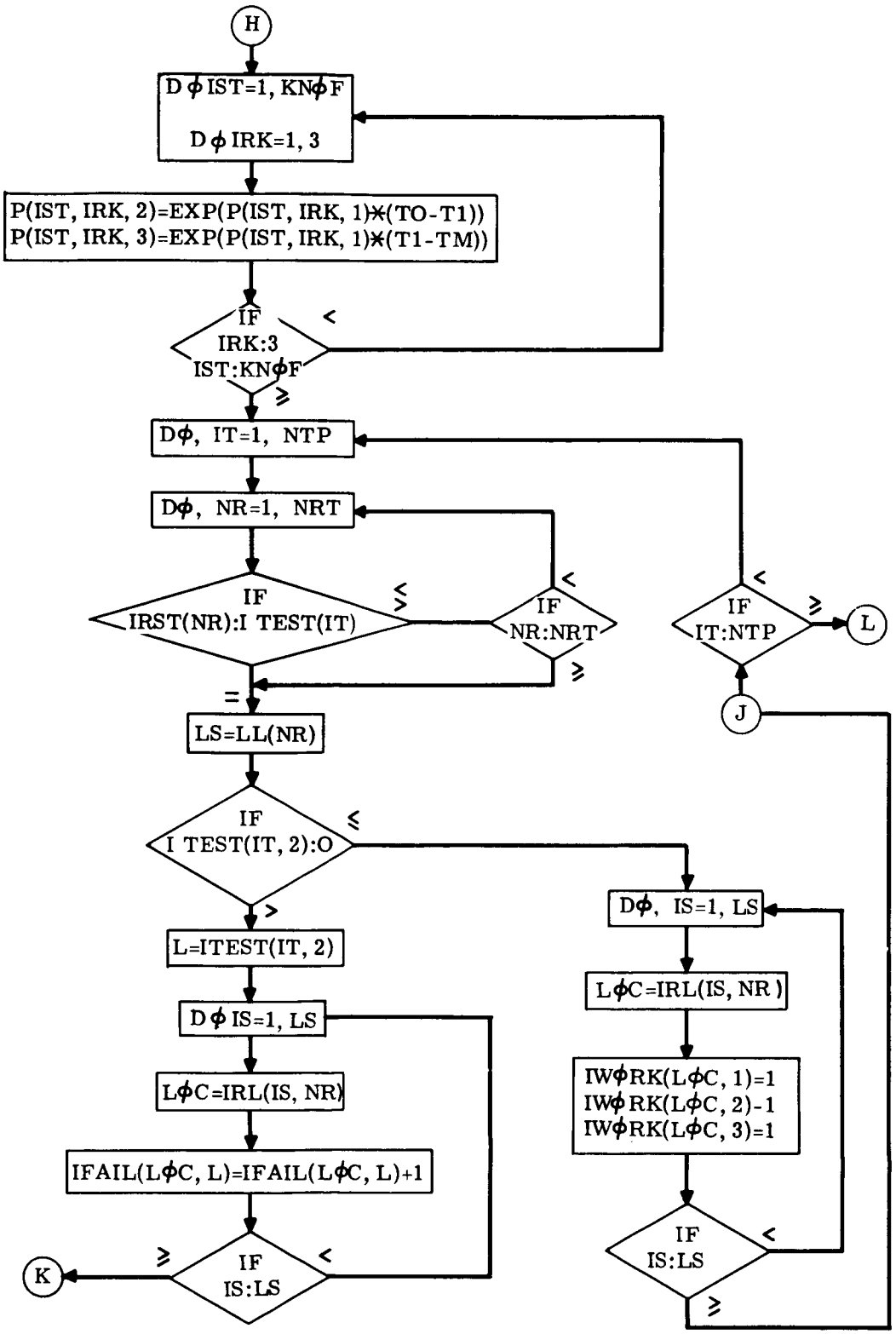

A
COMPILE FAN-IN LIST, NFB WITH NO. OF INPUTS, NI
I 10 = 1
LCT1=LCT-1
DO I1=1, LCT1
IF IUSD(I1):O
>
≤
I3 = I1+1
DO, I2=I3, LCT
IF ITO(I2):ITΦ(I1)
≤
=
IF NI(I10):O
≤
NI(I10)=2
>
NI(I10)=NI(I10)+1
IUSD (J2)=1
IF I2:LCT
<
IF NI(I10):O
≤
>
NFB(I10)=ITO(I1)
I10=I10+1
IF I1:LCT1
<
≥
NFI=I10-1
B

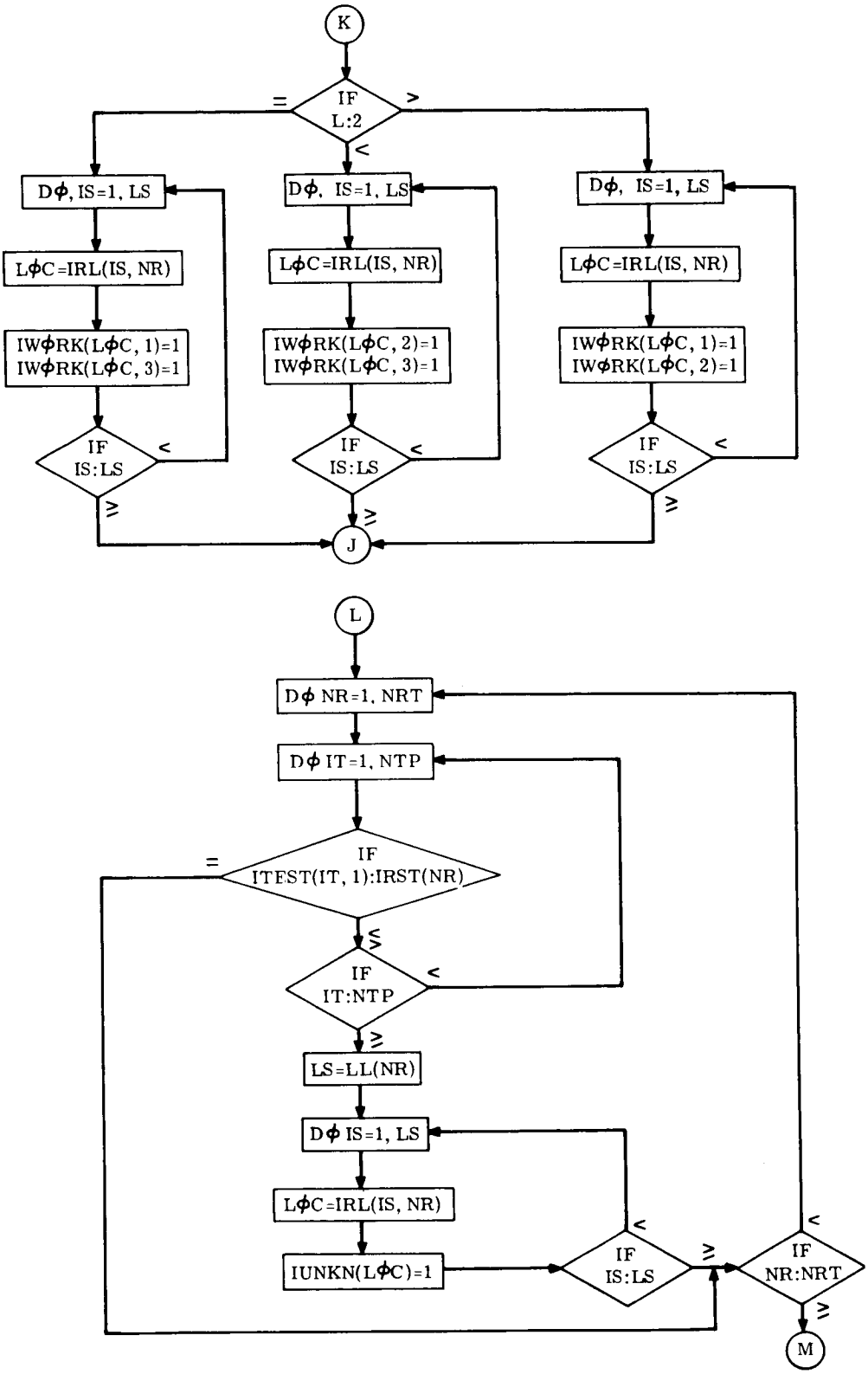

B
DO, NR=1, NRT
F
NMBR=IRST(NR)
DO, I=1, LCT
IF
IT (I):NMBR
IF
I:LCT
DO, II=1, KNOF
F
IF
IRL(II, NR):IFR(I)
IF
IRL(II, NR):O
IRL(II, NR)=IFR(I)
IF
II:KNOF
IF
IFR(I):NOF
DO I, 1=1, NFI
IF
IFR(I):1000
IF
NFB(I1):IFR(I)
IF
I1:NFI
IRL(II, NR)=O
D
NMBR=IFR(I)
C

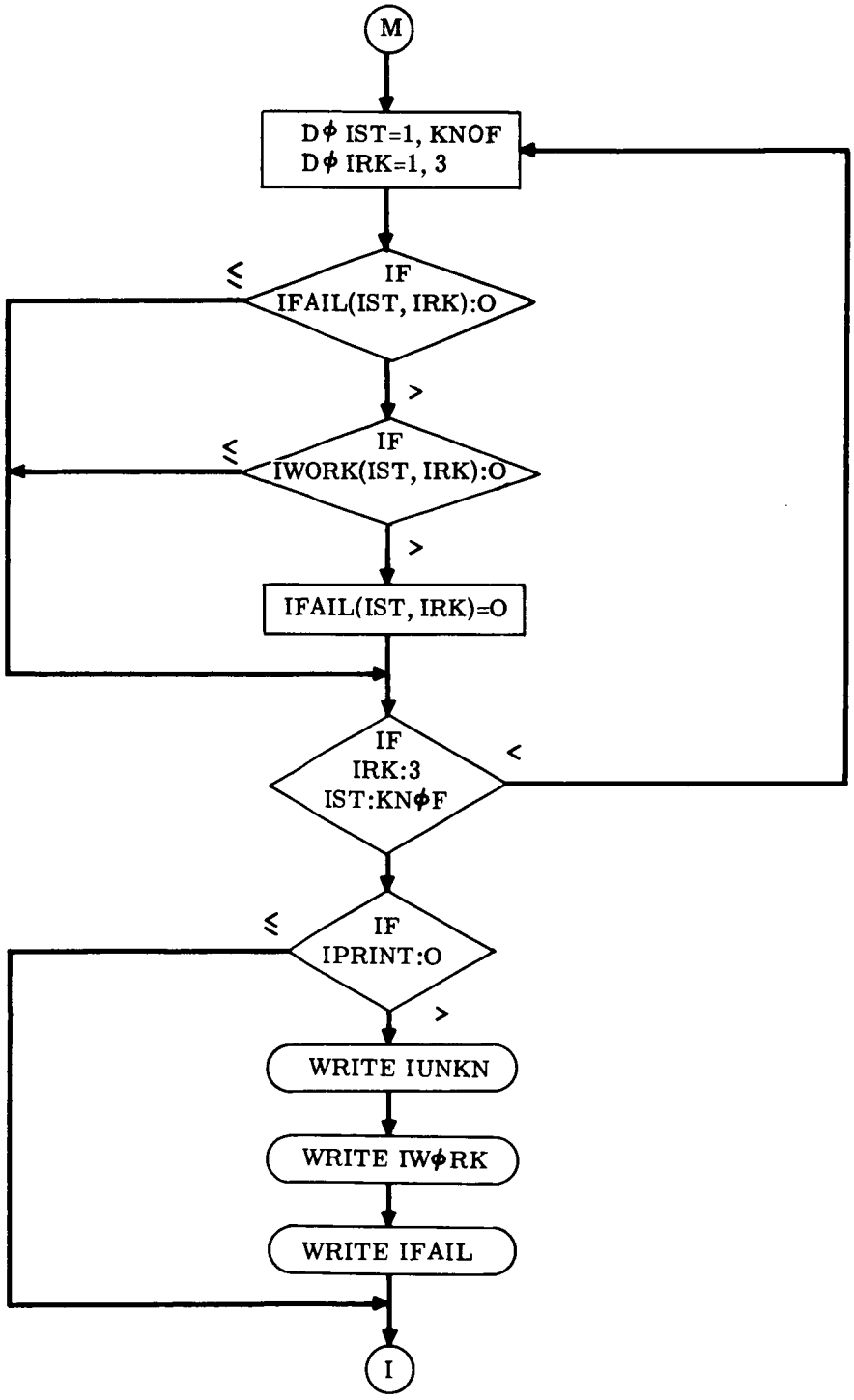

D
Dϕ, I2=1, LCT
IF
ITO(I2):IFR(I)
≤
=
<
IF
I2:LCT
≥
C
Dϕ, I3=1, KNOF
IF
IBR(I3):ITO(I2)
=
≤
<
IF
IBR(I3):O
>
IF
I3:KNϕF
≥
≤
IBRX(I3)=I2
I=I2
E

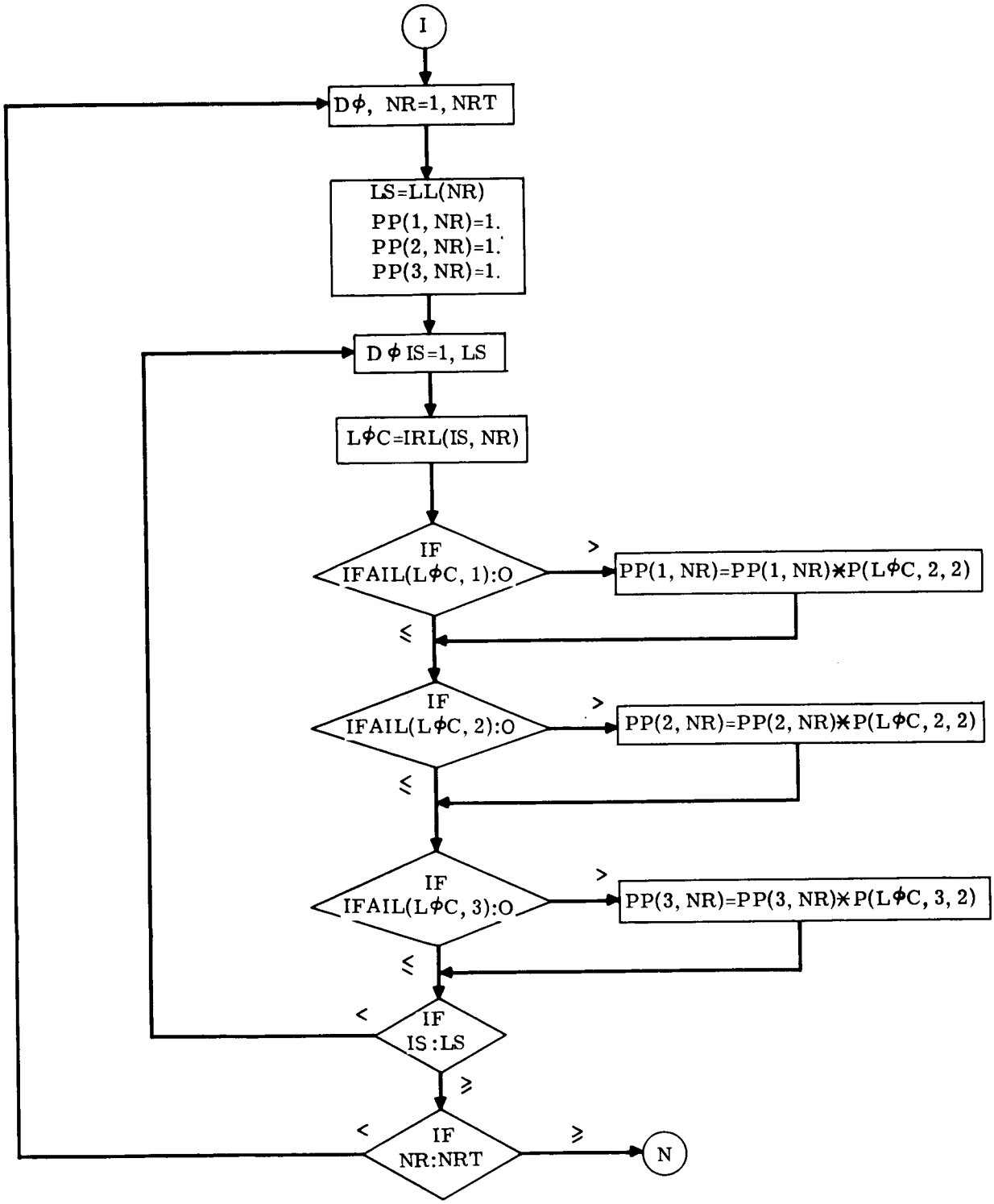

C
DΦ, I5=1, KNOF
MI5=KNΦF+1-I5
IF IBR(MI5):O
>
≤
IF I5:KNΦF
<
≥
G
>
DΦ I6=1, NFI
IF NFB(I6):IBR(MI5)
≤
IF I6:NFI
<
≥
=
NI(I6)=NI(I6)-1
IF NI(I6:O)
≤
IBR(MI5)=O
IDX=IBRX(MI5)+1
IBRX(MI5)=O
>
IDX =IBRX(MI5)+1
DΦ I7=IDX, LCT
IF ITO(I7):NFB(I6)
≤
IF I7:LCT
<
≥
=
IF IBR(MI5):O
>
IBRX(MI5)=I7
≤
I=I7
E

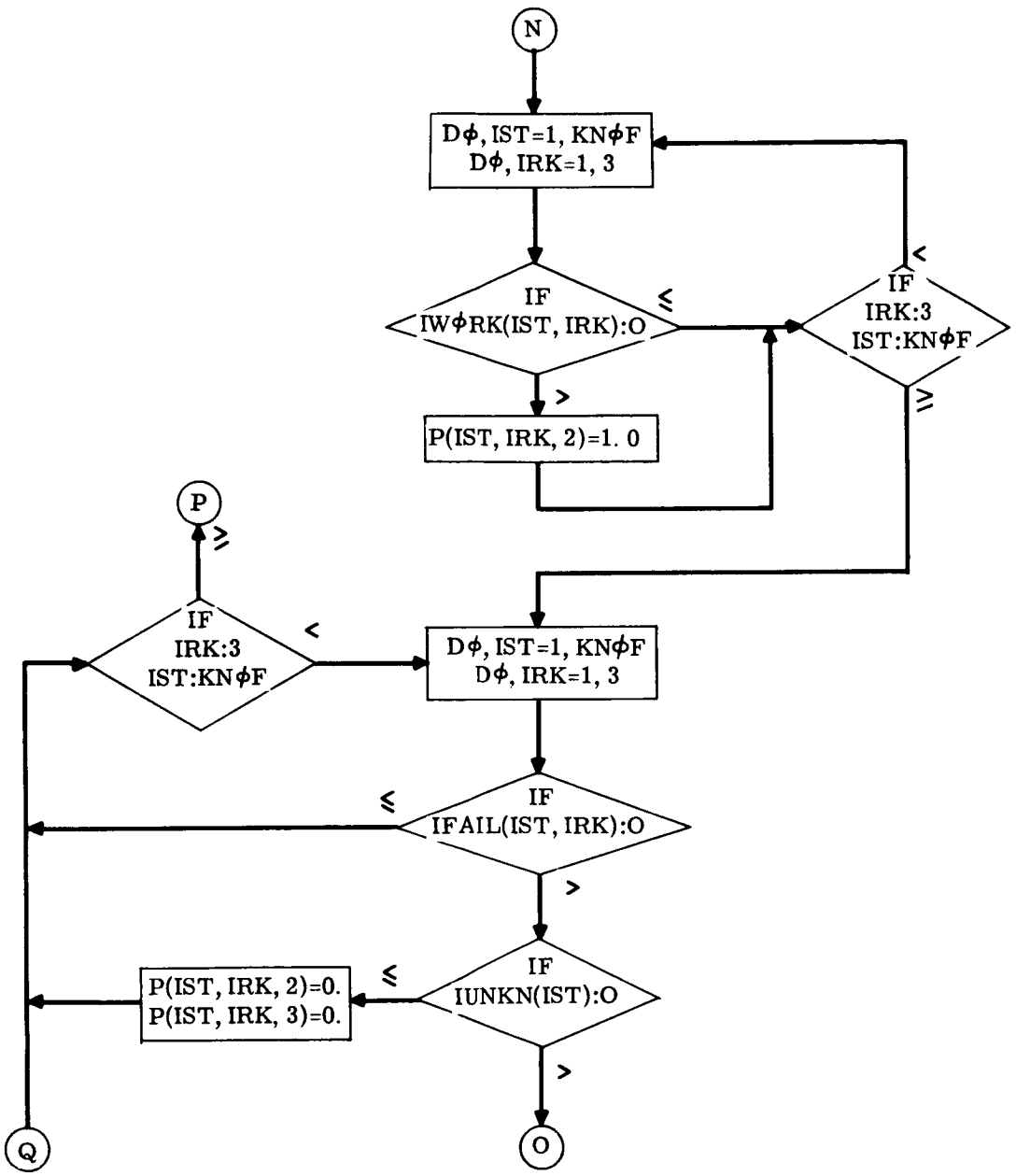

G
DO, INX=1, KNOF
IF
IRL(INX, NR):O
>
<
IF
INX:KNOF
≤
≥
LL(NR)=INX-1
LL(NR)=KNOF
IF
ITZREL:O
≤
>
B1=1. 0
B2=1. 0
B3=1. 0
LS=LL(NR)
DO M1=1, LS
LOC=IRL(M1, NR)
P(LOC, 1, 2)=EXP(P(LOC, 1, 1)*(TO-TM))
P(LOC, 2, 2)=EXP(P(LOC, 2, 1)*(TO-TM))
P(LOC, 3, 2)=EXP(P(LOC, 3, 1)*(TO-TM))
B1=B1*P(LOC, 1, 2)
B2=B2*P(LOC, 2, 2)
B3=B3*P(LOC, 3, 2)
IF
M1:LS
<
≥
R=R*(B1*B2+B1*B3+B2*B3-2.*B1*B2*B3)
IF
NR:NRT
<
≥
F
H

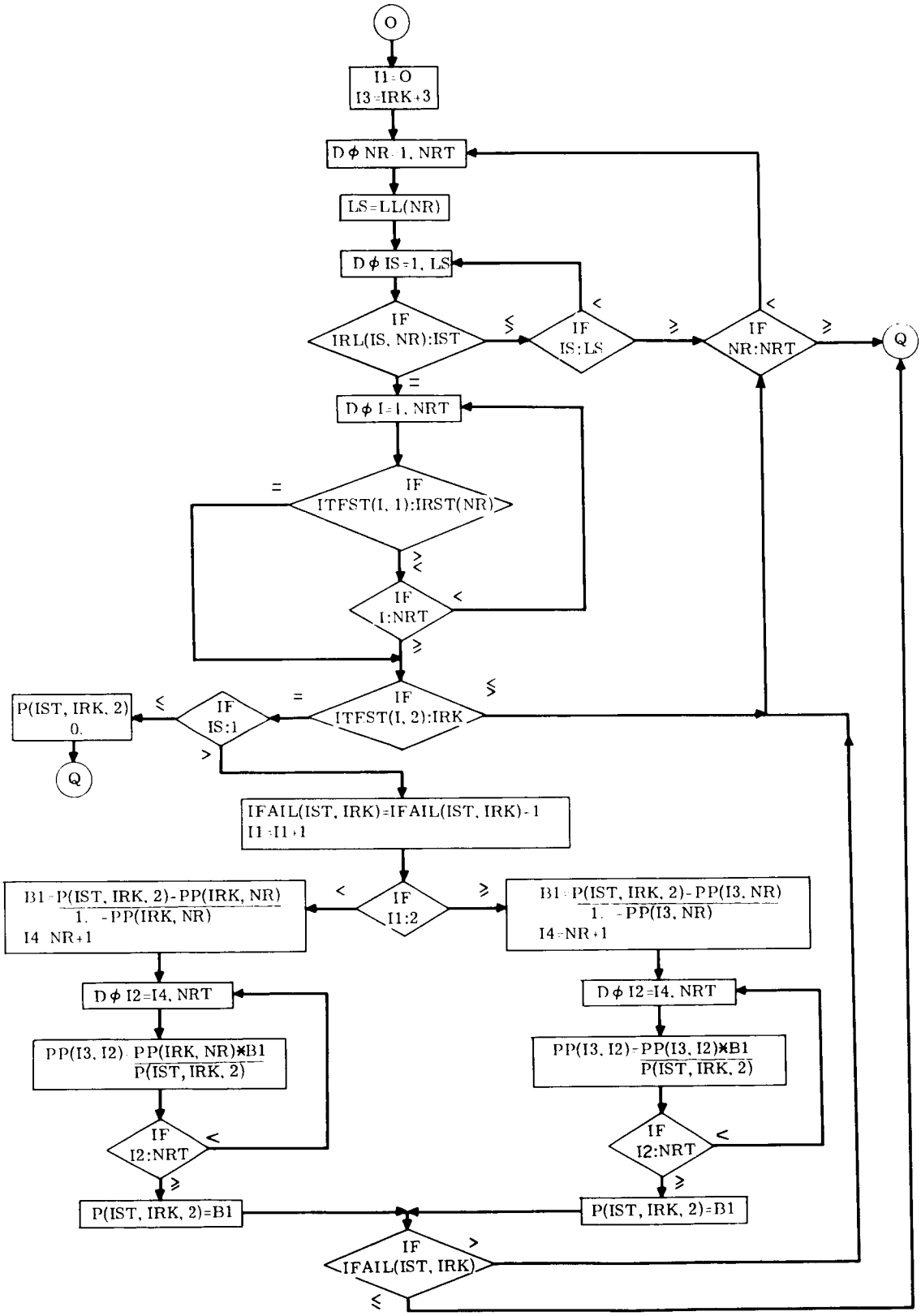

H
DϕIST=1, KNϕF
DϕIRK=1, 3
P(IST, IRK, 2)=EXP(P(IST, IRK, 1)*(TO-T1))
P(IST, IRK, 3)=EXP(P(IST, IRK, 1)*(T1-TM))
IF
IRK:3
IST:KNϕF
<
≥
DϕIT=1, NTP
DϕNR=1, NRT
IF
IRST(NR):I TEST(IT)
≤
>
IF
NR:NRT
<
≥
=
LS=LL(NR)
IF
IT:NTP
<
≥
L
J
IF
I TEST(IT, 2):O
≤
>
L=ITEST(IT, 2)
DϕIS=1, LS
LϕC=IRL(IS, NR)
IFAIL(LϕC, L)=IFAIL(LϕC, L)+1
IF
IS:LS
≥
<
K
DϕIS=1, LS
LϕC=IRL(IS, NR)
IWϕRK(LϕC, 1)=1
IWϕRK(LϕC, 2)-1
IWϕRK(LϕC, 3)=1
IF
IS:LS
<
≥

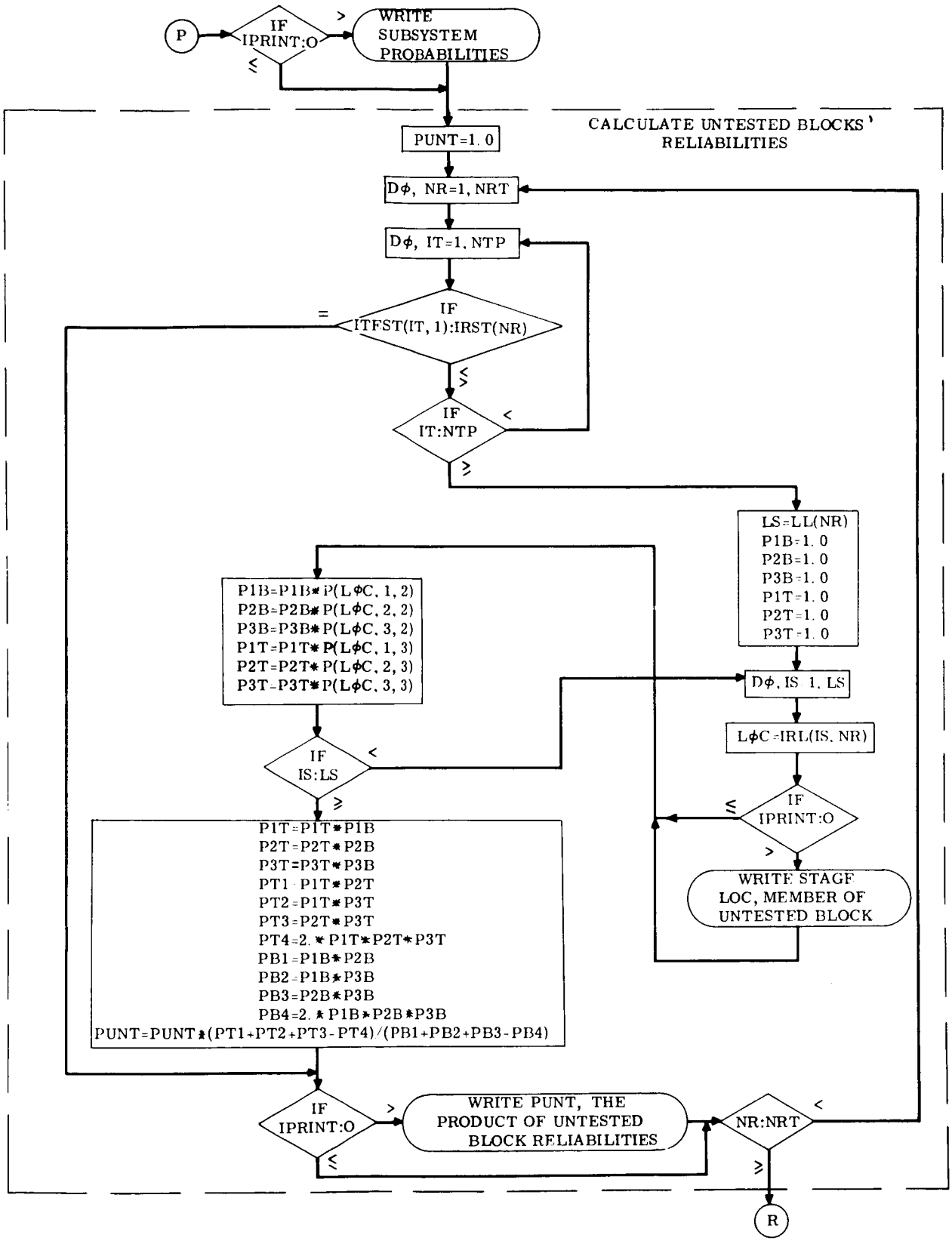

K
IF
L:2
=
<
>
Dϕ, IS=1, LS
LϕC=IRL(IS, NR)
IWϕRK(LϕC, 1)=1
IWϕRK(LϕC, 3)=1
IF
IS:LS
<
≥
Dϕ, IS=1, LS
LϕC=IRL(IS, NR)
IWϕRK(LϕC, 2)=1
IWϕRK(LϕC, 3)=1
IF
IS:LS
<
≥
Dϕ, IS=1, LS
LϕC=IRL(IS, NR)
IWϕRK(LϕC, 1)=1
IWϕRK(LϕC, 2)=1
IF
IS:LS
<
≥
J
L
Dϕ NR=1, NRT
Dϕ IT=1, NTP
IF
ITEST(IT, 1):IRST(NR)
=
≤
≥
IF
IT:NTP
<
≥
LS=LL(NR)
Dϕ IS=1, LS
LϕC=IRL(IS, NR)
IUNKN(LϕC)=1
IF
IS:LS
<
≥
IF
NR:NRT
<
≥
M

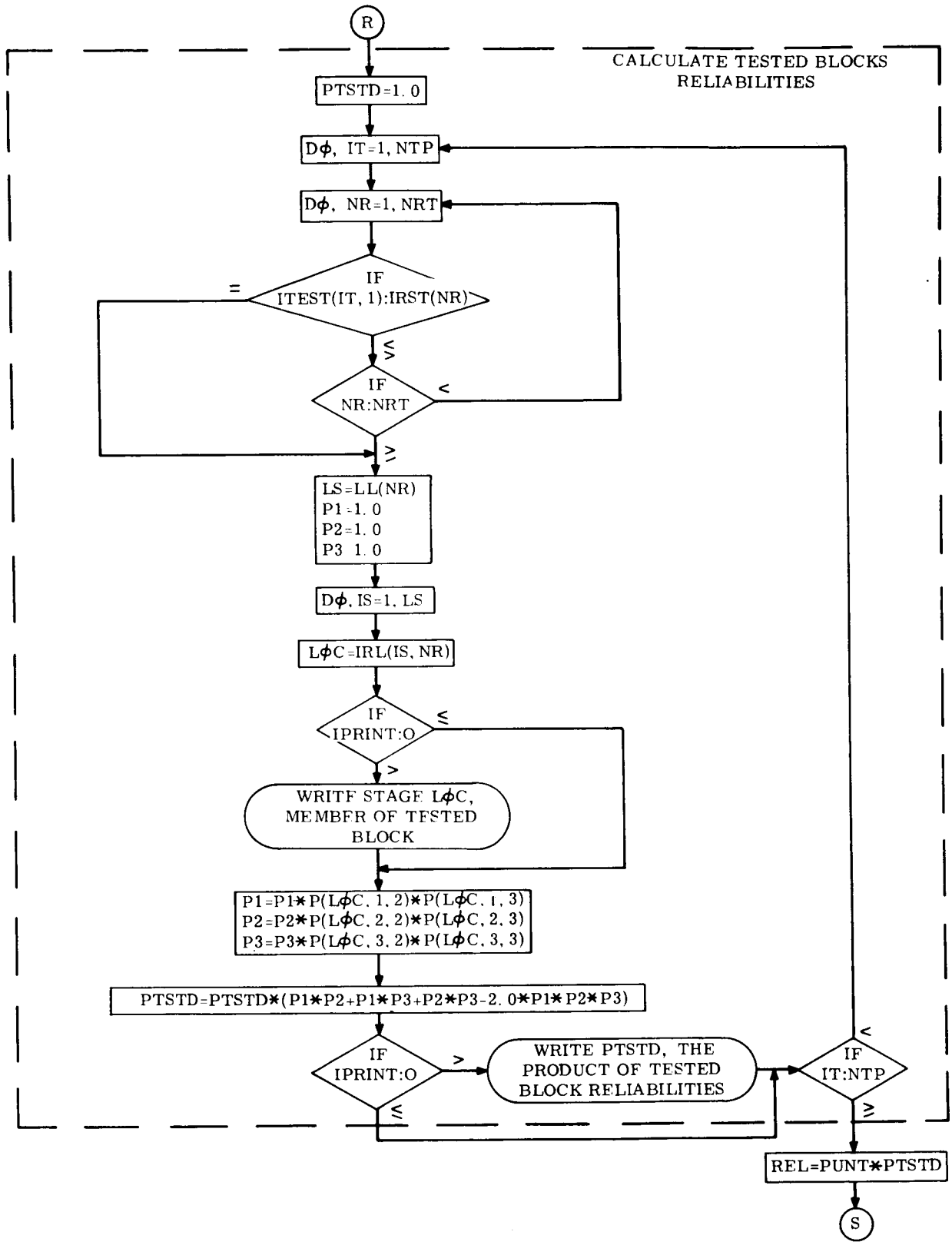

M
DΦ IST=1, KNOF
DΦ IRK=1, 3
IF
IFAIL(IST, IRK):O
≤
>
IF
IWORK(IST, IRK):O
≤
>
IFAIL(IST, IRK)=O
IF
IRK:3
IST:KNΦF
<
IF
IPRINT:O
≤
>
WRITE IUNKN
WRITE IWΦRK
WRITE IFAIL
I

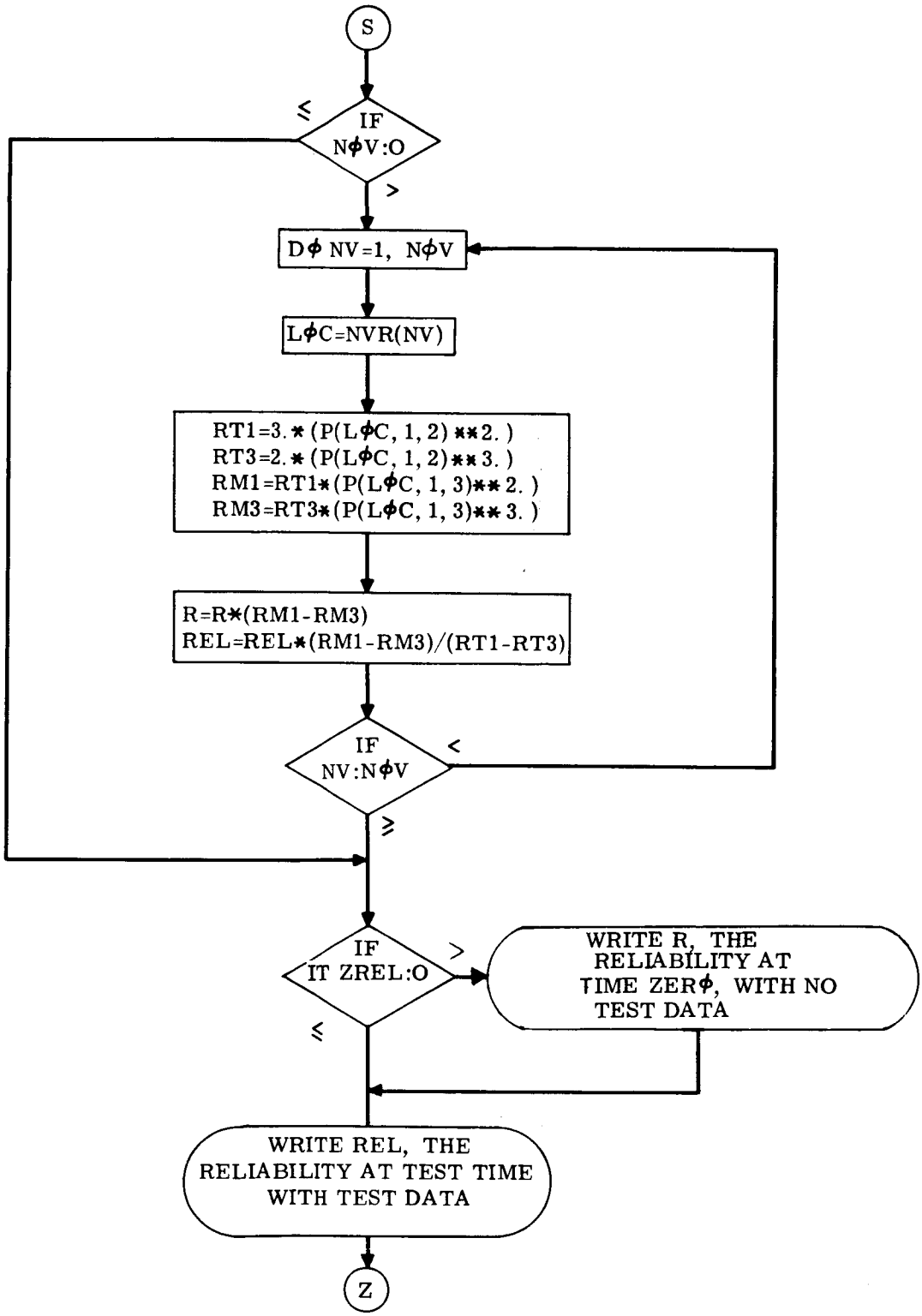

I
DΦ, NR=1, NRT
LS=LL(NR)
PP(1, NR)=1.
PP(2, NR)=1.
PP(3, NR)=1.
DΦ IS=1, LS
LΦC=IRL(IS, NR)
IF
IFAIL(LΦC, 1):O
>
PP(1, NR)=PP(1, NR)*P(LΦC, 2, 2)
≤
IF
IFAIL(LΦC, 2):O
>
PP(2, NR)=PP(2, NR)*P(LΦC, 2, 2)
≤
IF
IFAIL(LΦC, 3):O
>
PP(3, NR)=PP(3, NR)*P(LΦC, 3, 2)
≤
<
IF
IS:LS
≥
<
IF
NR:NRT
≥
N


N
Dϕ, IST=1, KNϕF
Dϕ, IRK=1, 3
IF
IWϕRK(IST, IRK):O
≤
>
P(IST, IRK, 2)=1. 0
IF
IRK:3
IST:KNϕF
<
≥
P
≥
IF
IRK:3
IST:KNϕF
<
Dϕ, IST=1, KNϕF
Dϕ, IRK=1, 3
≤
IF
IFAIL(IST, IRK):O
>
≤
IF
IUNKN(IST):O
P(IST, IRK, 2)=0.
P(IST, IRK, 3)=0.
>
Q
O


O
I1=O
I3=IRK+3
Dϕ NR=1, NRT
LS=LL(NR)
Dϕ IS=1, LS
IF IRL(IS, NR):IST
IF IS:LS
IF NR:NRT
Q
Dϕ I=1, NRT
IF ITFST(I, 1):IRST(NR)
IF I:NRT
P(IST, IRK, 2)=0.
IF IS:1
IF ITFST(I, 2):IRK
Q
IFAIL(IST, IRK)=IFAIL(IST, IRK)-1
I1=I1+1
B1=P(IST, IRK, 2)-PP(IRK, NR) / 1.-PP(IRK, NR)
I4=NR+1
IF I1:2
B1=P(IST, IRK, 2)-PP(I3, NR) / 1.-PP(I3, NR)
I4=NR+1
Dϕ I2=I4, NRT
PP(I3, I2)=PP(IRK, NR)*B1 / P(IST, IRK, 2)
IF I2:NRT
P(IST, IRK, 2)=B1
Dϕ I2=I4, NRT
PP(I3, I2)=PP(I3, I2)*B1 / P(IST, IRK, 2)
IF I2:NRT
P(IST, IRK, 2)=B1
IF IFAIL(IST, IRK)


P
IF IPRINT:O
>
≤
WRITE SUBSYSTEM PROBABILITIES
CALCULATE UNTESTED BLOCKS' RELIABILITIES
PUNT=1.0
Dϕ, NR=1, NRT
Dϕ, IT=1, NTP
=
IF ITFST(IT, 1):IRST(NR)
≤
IF IT:NTP
<
≥
LS=LL(NR)
P1B=1.0
P2B=1.0
P3B=1.0
P1T=1.0
P2T=1.0
P3T=1.0
P1B=P1B*P(LϕC, 1, 2)
P2B=P2B*P(LϕC, 2, 2)
P3B=P3B*P(LϕC, 3, 2)
P1T=P1T*P(LϕC, 1, 3)
P2T=P2T*P(LϕC, 2, 3)
P3T=P3T*P(LϕC, 3, 3)
Dϕ, IS 1, LS
LϕC=IRL(IS, NR)
IF IS:LS
<
≥
IF IPRINT:O
≤
>
WRITE STAGE LOC, MEMBER OF UNTESTED BLOCK
P1T=P1T*P1B
P2T=P2T*P2B
P3T=P3T*P3B
PT1=P1T*P2T
PT2=P1T*P3T
PT3=P2T*P3T
PT4=2.*P1T*P2T*P3T
PB1=P1B*P2B
PB2=P1B*P3B
PB3=P2B*P3B
PB4=2.*P1B*P2B*P3B
PUNT=PUNT*(PT1+PT2+PT3-PT4)/(PB1+PB2+PB3-PB4)
IF IPRINT:O
>
≤
WRITE PUNT, THE PRODUCT OF UNTESTED BLOCK RELIABILITIES
NR:NRT
<
≥
R


R
CALCULATE TESTED BLOCKS RELIABILITIES
PTSTD=1. 0
Dϕ, IT=1, NTP
Dϕ, NR=1, NRT
IF ITEST(IT, 1):IRST(NR)
=
≤
>
IF NR:NRT
<
≥
LS=LL(NR)
P1=1. 0
P2=1. 0
P3 1. 0
Dϕ, IS=1, LS
LϕC=IRL(IS, NR)
IF IPRINT:O
≤
>
WRITE STAGE LϕC, MEMBER OF TESTED BLOCK
P1=P1*P(LϕC, 1, 2)*P(LϕC, 1, 3)
P2=P2*P(LϕC, 2, 2)*P(LϕC, 2, 3)
P3=P3*P(LϕC, 3, 2)*P(LϕC, 3, 3)
PTSTD=PTSTD*(P1*P2+P1*P3+P2*P3-2. 0*P1*P2*P3)
IF IPRINT:O
>
WRITE PTSTD, THE PRODUCT OF TESTED BLOCK RELIABILITIES
≤
IF IT:NTP
<
≥
REL=PUNT*PTSTD
S


S
IF
NϕV:O
≤
>
DϕNV=1, NϕV
LϕC=NVR(NV)
RT1=3.*(P(LϕC, 1, 2)**2.)
RT3=2.*(P(LϕC, 1, 2)**3.)
RM1=RT1*(P(LϕC, 1, 3)**2.)
RM3=RT3*(P(LϕC, 1, 3)**3.)
R=R*(RM1-RM3)
REL=REL*(RM1-RM3)/(RT1-RT3)
IF
NV:NϕV
<
≥
IF
IT ZREL:O
>
≤
WRITE R, THE RELIABILITY AT TIME ZERϕ, WITH NO TEST DATA
WRITE REL, THE RELIABILITY AT TEST TIME WITH TEST DATA
Z

VI. FORTRAN IV. PROGRAM LISTING

The following pages contain a complete FORTRAN IV listing of the reliability analysis program.

```
00100     1.   C      POST TEST
00100     2.   C-----PROGRAM TO ESTIMATE REDUNDANT SYSTEM RELIABILITY AFTER TEST
00101     3.         DIMENSION IRL(150,75),IFR(200),ITO(200),IRST(75),NFB(150)
00103     4.         DIMENSION NI(150),IBR(150),IBRX(150),LL(75),NVR(75),IUSD(200)
00104     5.         DIMENSION ITEST(75,2),IFAIL(150,3),IWORK(150,3),PP(6,75)
00105     6.         DIMENSION IUNKN(150),P(150,3,3)
00106     7.    400  READ(5,301)LCT,NOF
00112     8.         READ(5,301)(IFR(IT),IT=1,LCT)
00120     9.         READ(5,301)(ITO(IT),IT=1,LCT)
00126    10.         READ(5,305)T0,T1,TM
00133    11.         READ(5,301)ITZREL
00136    12.         READ(5,301)NTP
00141    13.         READ(5,301)((ITEST(NR,NFR),NFR=1,2),NR=1,NTP)
00152    14.         READ(5,301)IEQULR
00155    15.         KNOF=2*NOF
00156    16.         IF(IEQULR)675,675,676
00161    17.    676  READ(5,307)(P(IST,1,1),IST=1,KNOF)
00167    18.         DO 677 IST=1,KNOF
00172    19.         P(IST,2,1)=P(IST,1,1)
00173    20.    677  P(IST,3,1)=P(IST,1,1)
00175    21.         GO TO 678
00176    22.    675  READ(5,307)((P(IST,IRK,1),IRK=1,3),IST=1,KNOF)
00207    23.    678  READ(5,301)NOV
00212    24.         IF(NOV)601,601,801
00215    25.    801  READ(5,301)(NVR(NV),NV=1,NOV)
00223    26.    601  READ(5,301)IPRINT
00226    27.         WRITE(6,309)
00230    28.         WRITE(6,310)(IFR(I),ITO(I),I=1,LCT)
00237    29.         WRITE(6,311)NOF
00242    30.         WRITE(6,312)T0,T1,TM
00247    31.         WRITE(6,313)((ITEST(I,J),J=1,2),I=1,NTP)
00260    32.         WRITE(6,314)(N,P(N,1,1),P(N,2,1),P(N,3,1),P(N+NOF,1,1),P(N+NOF,2,1
00260    33.        1),P(N+NOF,3,1),N=1,NOF)
00260    34.   C-----CLEAR STORAGE
00274    35.         DO 108 LL1=1,KNOF
00277    36.         IBR(LL1)=0
00300    37.         IBRX(LL1)=0
00301    38.         IRST(LL1)=0
00302    39.         NI(LL1)=0
00303    40.         NFB(LL1)=0
00304    41.         IUSD(LL1)=0
00305    42.         DO 108 NR=1,NOF
00310    43.         IRST(NR)=0
00311    44.    108  IRL(LL1,NR)=0
00314    45.         DO 800 IST=1,KNOF
00317    46.         IUNKN(IST)=0
00320    47.         DO 800 IRK=1,3
00323    48.         IWORK(IST,IRK)=0
```

```
00324    49.     800  IFAIL(IST,IRK)=0
00327    50.          R=1.
00327    51.    C-----COMPILE RESTORER LIST
00330    52.          NRT=0
00331    53.          DO 102 IIJ=1,LCT
00334    54.          IF(ITO(IIJ)-NOF)102,102,103
00337    55.     103  IF(ITO(IIJ)-1000)104,102,102
00342    56.     104  DO 101 IJ=1,NOF
00345    57.          IF(ITO(IIJ)-IRST(IJ))105,102,105
00350    58.     105  IF(IRST(IJ))106,106,101
00353    59.     106  IRST(IJ)=ITO(IIJ)
00354    60.          NRT=NRT+1
00355    61.          GO TO 102
00356    62.     101  CONTINUE
00360    63.     102  CONTINUE
00360    64.    C-----COMPILE FAN-IN LIST,NFB,WITH NO. OF INPUTS,NI
00362    65.          I10=1
00363    66.          LCT1=LCT-1
00364    67.          DO 908 I1=1,LCT1
00367    68.          IF(IUSD(I1))901,901,908
00372    69.     901  I3=I1+1
00373    70.          DO 906 I2=I3,LCT
00376    71.          IF(ITO(I2)-ITO(I1))906,902,906
00401    72.     902  IF(NI(I10))903,903,904
00404    73.     903  NI(I10)=2
00405    74.          GO TO 905
00406    75.     904  NI(I10)=NI(I10)+1
00407    76.     905  IUSD(I2)=1
00410    77.     906  CONTINUE
00412    78.          IF(NI(I10))908,908,907
00415    79.     907  NFB(I10)=ITO(I1)
00416    80.          I10=I10+1
00417    81.     908  CONTINUE
00421    82.          NFI=I10-1
00421    83.    C-----BEGIN COMPILING ISOLATED BLOCKS
00422    84.     111  DO 70 NR=1,NRT
00425    85.          NMBR=IRST(NR)
00426    86.     9    DO 3 I=1,LCT
00431    87.          IF(ITO(I)-NMBR)3,2,3
00434    88.     3    CONTINUE
00436    89.     2    DO 15 II=1,KNOF
00441    90.          IF(IRL(II,NR)-IFR(I))14,25,14
00444    91.     14   IF(IRL(II,NR))16,16,15
00447    92.     16   IRL(II,NR)=IFR(I)
00450    93.          GO TO 4
00451    94.     15   CONTINUE
00453    95.     4    IF(IFR(I)-NOF)5,5,109
00456    96.     109  IF(IFR(I)-1000)25,110,110
00461    97.     110  IRL(II,NR)=0
00462    98.          GO TO 25
00463    99.     5    DO 7 I1=1,NFI
00466   100.          IF(NFB(I1)-IFR(I))7,8,7
00471   101.     7    CONTINUE
00473   102.          NMBR=IFR(I)
00473   103.    C-----IFR IS NOT A FAN-IN STAGE
00474   104.          GO TO 9
00474   105.    C-----IFR IS A FAN-IN STAGE
00475   106.     8    NI(I1)=NI(I1)-1
00476   107.          DO 10 I2=1,LCT
00501   108.          IF(ITO(I2)-IFR(I)) 10,11,10
00504   109.     11   DO 12 I3=1,KNOF
00507   110.          IF(IBR(I3)-ITO(I2))18,19,18
00512   111.     18   IF(IBR(I3))20,20,12
00515   112.     20   IBR(I3)=ITO(I2)
00516   113.     19   IBRX(I3)=I2
00517   114.          I=I2
```

```
00520   115.          GO TO 2
00521   116.    12    CONTINUE
00523   117.    10    CONTINUE
00523   118.   C-----END OF BRANCH---SEARCH FOR NEXT BRANCH
00525   119.    25    DO 26 I5=1,KNOF
00530   120.          MI5=KNOF+1-I5
00531   121.          IF(IBR(MI5))26,26,27
00534   122.    27    DO 28 I6=1,NFI
00537   123.          IF(NFB(I6)-IBR(MI5))28,29,28
00542   124.    29    NI(I6)=NI(I6)-1
00543   125.          IF(NI(I6))30,30,22
00546   126.    30    IBR(MI5)=0
00547   127.          IDX=IBRX(MI5)+1
00550   128.          IBRX(MI5)=0
00551   129.          GO TO 31
00552   130.    22    IDX=IBRX(MI5)+1
00553   131.    31    DO 32 I7=IDX,LCT
00556   132.          IF(ITO(I7)-NFB(I6))32,33,32
00561   133.    33    IF(IBR(MI5))35,35,34
00564   134.    34    IBRX(MI5)=I7
00565   135.    35    I=I7
00566   136.          GO TO 2
00567   137.    32    CONTINUE
00571   138.    28    CONTINUE
00573   139.    26    CONTINUE
00575   140.          DO 39 INX=1,KNOF
00600   141.          IF(IRL(INX,NR))38,38,39
00603   142.    38    LL(NR)=INX-1
00604   143.          GO TO 502
00605   144.    39    CONTINUE
00607   145.          LL(NR)=KNOF
00610   146.    502   IF(ITZREL)70,70,503
00613   147.    503   B1=1.0
00614   148.          B2=1.0
00615   149.          B3=1.0
00616   150.          LS=LL(NR)
00617   151.          DO 504 M1=1,LS
00622   152.          LOC=IRL(M1,NR)
00623   153.          P(LOC,1,2)=EXP(P(LOC,1,1)*(T0-TM))
00624   154.          P(LOC,2,2)=EXP(P(LOC,2,1)*(T0-TM))
00625   155.          P(LOC,3,2)=EXP(P(LOC,3,1)*(T0-TM))
00626   156.          B1=B1*P(LOC,1,2)
00627   157.          B2=B2*P(LOC,2,2)
00630   158.    504   B3=B3*P(LOC,3,2)
00632   159.          R=R*(B1*B2+B1*B3+B2*B3-2.*B1*B2*B3)
00632   160.   C-----MAKE IRL LIST FOR NEXT RESTORER ON LIST
00633   161.    70    CONTINUE
00633   162.   C-----COMPUTE PROBS. FOR ALL UNITS
00635   163.          DO 250 IST=1,KNOF
00640   164.          DO 250 IRK=1,3
00643   165.          P(IST,IRK,2)=EXP(P(IST,IRK,1)*(T0-T1))
00644   166.    250   P(IST,IRK,3)=EXP(P(IST,IRK,1)*(T1-TM))
00644   167.   C-----FIND ALL TESTED BLOCKS-PUT UNITS IN IWORK OR IFAIL
00647   168.          DO 401 IT=1,NTP
00652   169.          DO 402 NR=1,NRT
00655   170.          IF(IRST(NR)-ITEST(IT,1))402,403,402
00660   171.    402   CONTINUE
00662   172.    403   LS=LL(NR)
00663   173.          IF(ITEST(IT,2))404,404,405
00666   174.    405   L=ITEST(IT,2)
00667   175.          DO 410 IS=1,LS
00672   176.          LOC=IRL(IS,NR)
00673   177.          IFAIL(LOC,L)=IFAIL(LOC,L)+1
00674   178.    410   CONTINUE
00676   179.          GO TO 406
00676   180.   C-----NO FAILURES IN THIS STRING
```

```
00677   181.     404  DO 408 IS=1,LS
00702   182.          LOC=IRL(IS,NR)
00703   183.          IWORK(LOC,1)=1
00704   184.          IWORK(LOC,2)=1
00705   185.     408  IWORK(LOC,3)=1
00707   186.          GO TO 401
00710   187.     406  IF(L-2)414,415,416
00710   188.    C-----FAILURE IN RANK ONE
00713   189.     414  DO 409 IS=1,LS
00716   190.          LOC=IRL(IS,NR)
00717   191.          IWORK(LOC,2)=1
00720   192.     409  IWORK(LOC,3)=1
00722   193.          GO TO 401
00722   194.    C-----FAILURE IN RANK TWO
00723   195.     415  DO 420 IS=1,LS
00726   196.          LOC=IRL(IS,NR)
00727   197.          IWORK(LOC,1)=1
00730   198.     420  IWORK(LOC,3)=1
00732   199.          GO TO 401
00732   200.    C-----FAILURE IN RANK THREE
00733   201.     416  DO 421 IS=1,LS
00736   202.          LOC=IRL(IS,NR)
00737   203.          IWORK(LOC,1)=1
00740   204.     421  IWORK(LOC,2)=1
00742   205.     401  CONTINUE
00742   206.    C-----FIND ALL UNTESTED BLOCKS--PUT STAGES IN IUNKN
00744   207.          DO 417 NR=1,NRT
00747   208.          DO 418 IT=1,NTP
00752   209.          IF(ITEST(IT,1)-IRST(NR))418,417,418
00755   210.     418  CONTINUE
00757   211.          LS=LL(NR)
00760   212.          DO 419 IS=1,LS
00763   213.          LOC=IRL(IS,NR)
00764   214.     419  IUNKN(LOC)=1
00766   215.     417  CONTINUE
00766   216.    C-----ALL UNITS  ARE ON PROPER LISTS
00766   217.    C-----COMPARE FAILED AND WORKING LISTS
00770   218.          DO 422 IST=1,KNOF
00773   219.          DO 422 IRK=1,3
00776   220.          IF(IFAIL(IST,IRK))422,422,423
01001   221.     423  IF(IWORK(IST,IRK))422,422,424
01004   222.     424  IFAIL(IST,IRK)=0
01005   223.     422  CONTINUE
01010   224.          IF(IPRINT)621,621,620
01013   225.     620  WRITE(6,317)
01015   226.          WRITE(6,320)(IST,IUNKN(IST),IST=1,KNOF)
01024   227.          WRITE(6,319)
01026   228.          WRITE(6,321)(I1,IWORK(I1,1),IWORK(I1,2),IWORK(I1,3),I1=1,KNOF)
01037   229.          WRITE(6,318)
01041   230.          WRITE(6,321)(I1,IFAIL(I1,1),IFAIL(I1,2),IFAIL(I1,3),I1=1,KNOF)
01041   231.    C-----ALL LISTS COMPARED AND COMPLETED
01041   232.    C-----COMPUTE PROB.PROD.FOR EACH IRL LIST
01052   233.     621  DO 251 NR=1,NRT
01055   234.          LS=LL(NR)
01056   235.          PP(1,NR)=1.
01057   236.          PP(2,NR)=1.
01060   237.          PP(3,NR)=1.
01061   238.          DO 251 IS=1,LS
01064   239.          LOC=IRL(IS,NR)
01065   240.          IF(IFAIL(LOC,1))254,254,253
01070   241.     253  PP(1,NR)=PP(1,NR)*P(LOC,1,2)
01071   242.     254  IF(IFAIL(LOC,2))257,257,255
01074   243.     255  PP(2,NR)=PP(2,NR)*P(LOC,2,2)
01075   244.     257  IF(IFAIL(LOC,3))251,251,258
01100   245.     258  PP(3,NR)=PP(3,NR)*P(LOC,3,2)
01101   246.     251  CONTINUE
01101   247.    C-----BEGIN COMPUTING RELIABILITIES
```

```
01101   248.   C-----COMPUTE WORKING UNITS RELIABILITIES
01104   249.          DO 433 IST=1,KNOF
01107   250.          DO 433 IRK=1,3
01112   251.          IF(IWORK(IST,IRK))433,433,434
01115   252.    434   P(IST,IRK,2)=1.0
01116   253.    433   CONTINUE
01116   254.   C-----COMPUTE FAILED UNITS RELIABILITIES
01121   255.          DO 438 IST=1,KNOF
01124   256.          DO 438 IRK=1,3
01127   257.          IF(IFAIL(IST,IRK))438,438,439
01127   258.   C-----ONE OR MORE FAILED LISTS
01132   259.    439   IF(IUNKN(IST))441,441,442
01132   260.   C-----NOT ON UNTESTED LIST
01135   261.    441   P(IST,IRK,2)=0.
01136   262.          P(IST,IRK,3)=0.
01137   263.          GO TO 438
01137   264.   C-----ONE OR MORE FAILED LISTS AND UNTESTED LIST
01140   265.    442   I1=0
01141   266.          I3=IRK+3
01142   267.          DO 515 NR=1,NRT
01145   268.          LS=LL(NR)
01146   269.          DO 514 IS=1,LS
01151   270.          IF(IRL(IS,NR)-IST)514,510,514
01154   271.    510   DO 511 I=1,NRT
01157   272.          IF(ITEST(I,1)-IRST(NR))511,512,511
01162   273.    511   CONTINUE
01164   274.    512   IF(ITEST(I,2)-IRK)515,513,515
01167   275.    513   IF(IS-1)522,522,523
01172   276.    522   P(IST,IRK,2)=0.
01173   277.          GO TO 438
01174   278.    523   IFAIL(IST,IRK)=IFAIL(IST,IRK)-1
01175   279.          I1=I1+1
01176   280.          IF(I1-2)517,518,518
01201   281.    517   B1=(P(IST,IRK,2)-PP(IRK,NR))/(1.-PP(IRK,NR))
01202   282.          I4=NR+1
01203   283.          DO 519 I2=I4,NRT
01206   284.    519   PP(I3,I2)=(PP(IRK,NR)*B1)/P(IST,IRK,2)
01210   285.          P(IST,IRK,2)=B1
01211   286.          GO TO 521
01212   287.    518   B1=(P(IST,IRK,2)-PP(I3,NR))/(1.-PP(I3,NR))
01213   288.          I4=NR+1
01214   289.          DO 520 I2=I4,NRT
01217   290.    520   PP(I3,I2)=(PP(I3,I2)*B1)/P(IST,IRK,2)
01221   291.          P(IST,IRK,2)=B1
01222   292.    521   IF(IFAIL(IST,IRK))438,438,515
01225   293.    514   CONTINUE
01227   294.    515   CONTINUE
01231   295.    438   CONTINUE
01234   296.    449   IF(IPRINT)623,623,622
01237   297.    622   WRITE(6,322)
01241   298.          WRITE(6,323)(IST,P(IST,1,2),P(IST,2,2),P(IST,3,2),P(IST,1,3),P(IST
01241   299.         1,2,3),P(IST,3,3),IST=1,KNOF)
01255   300.          I1=0
01256   301.          WRITE(6,324)
01256   302.   C-----COMPUTE RELIABILITY OF UNTESTED STRINGS (BLOCKS)
01260   303.    623   PUNT=1.0
01261   304.          DO 450 NR=1,NRT
01264   305.          DO 451 IT=1,NTP
01267   306.          IF(ITEST(IT,1)-IRST(NR))451,450,451
01272   307.    451   CONTINUE
01274   308.          IF(IPRINT)625,625,624
01277   309.    624   I1=I1+1
01300   310.          WRITE(6,325)I1
01303   311.    625   LS=LL(NR)
01304   312.          P1B=1.0
01305   313.          P2B=1.0
```

```
01306   314.          P3B=1.0
01307   315.          P1T=1.0
01310   316.          P2T=1.0
01311   317.          P3T=1.0
01312   318.          DO 452 IS=1,LS
01315   319.          LOC=IRL(IS,NR)
01316   320.          IF(IPRINT)627,627,626
01321   321.     626  WRITE(6,326)LOC
01324   322.     627  P1B=P1B*P(LOC,1,2)
01325   323.          P2B=P2B*P(LOC,2,2)
01326   324.          P3B=P3B*P(LOC,3,2)
01327   325.          P1T=P1T*P(LOC,1,3)
01330   326.          P2T=P2T*P(LOC,2,3)
01331   327.     452  P3T=P3T*P(LOC,3,3)
01333   328.          P1T=P1T*P1B
01334   329.          P2T=P2T*P2B
01335   330.          P3T=P3T*P3B
01336   331.          PT1=P1T*P2T
01337   332.          PT2=P1T*P3T
01340   333.          PT3=P2T*P3T
01341   334.          PT4=P1T*P2T*P3T*2.0
01342   335.          PB1=P1B*P2B
01343   336.          PB2=P1B*P3B
01344   337.          PB3=P2B*P3B
01345   338.          PB4=P1B*P2B*P3B*2.0
01346   339.          PUNT=PUNT*(PT1+PT2+PT3-PT4)/(PB1+PB2+PB3-PB4)
01347   340.          IF(IPRINT)450,450,628
01352   341.     628  WRITE(6,327)PUNT
01355   342.     450  CONTINUE
01355   343.    C-----COMPUTE RELIABILITY OF TESTED STRINGS (BLOCKS)
01357   344.          PTSTD=1.0
01360   345.          IF(IPRINT)630,630,629
01363   346.     629  I1=0
01364   347.          WRITE(6,328)
01366   348.     630  DO 453 IT=1,NTP
01371   349.          DO 454 NR=1,NRT
01374   350.          IF(ITEST(IT,1)-IRST(NR))454,455,454
01377   351.     454  CONTINUE
01401   352.     455  IF(IPRINT)632,632,631
01404   353.     631  I1=I1+1
01405   354.          WRITE(6,329)I1
01410   355.     632  LS=LL(NR)
01411   356.          P1=1.0
01412   357.          P2=1.0
01413   358.          P3=1.0
01414   359.          DO 456 IS=1,LS
01417   360.          LOC=IRL(IS,NR)
01420   361.          IF(IPRINT)634,634,633
01423   362.     633  WRITE(6,326)LOC
01426   363.     634  P1=P1*P(LOC,1,2)*P(LOC,1,3)
01427   364.          P2=P2*P(LOC,2,2)*P(LOC,2,3)
01430   365.     456  P3=P3*P(LOC,3,2)*P(LOC,3,3)
01432   366.          PTSTD=PTSTD*(P1*P2+P1*P3+P2*P3-2.0*P1*P2*P3)
01433   367.          IF(IPRINT)453,453,635
01436   368.     635  WRITE(6,330)PTSTD
01441   369.     453  CONTINUE
01441   370.    C-----COMPUTE SYSTEM RELIABILITY AFTER TEST
01443   371.          REL=PUNT*PTSTD
01444   372.          IF(NOV)460,460,457
01447   373.     457  DO 458 NV=1,NOV
01452   374.          LOC=NVR(NV)
01453   375.          RT1=3.*(P(LOC,1,2)**2.)
01454   376.          RT3=2.*(P(LOC,1,2)**3.)
01455   377.          RM1=RT1*(P(LOC,1,3)**2.)
01456   378.          RM3=RT3*(P(LOC,1,3)**3.)
01457   379.          R=R*(RM1-RM3)
```

```
01460   380.     458  REL=REL*(RM1-RM3)/(RT1-RT3)
01460   381.    C-----PRINT RESULTS
01462   382.     460  IF(ITZREL)703,703,505
01465   383.     505  WRITE(6,315)R
01470   384.     703  WRITE(6,316)REL
01473   385.          GO TO 400
01474   386.     200  FORMAT(10I10)
01475   387.     301  FORMAT(14I5)
01476   388.     305  FORMAT(3F10.0)
01477   389.     307  FORMAT(6E12.6)
01500   390.     309  FORMAT(31H1POST-TEST RELIABILITY ANALYSIS///)
01501   391.     310  FORMAT(24H SYSTEM INTERCONNECTIONS7/3X,47HFROM TO   FROM TO   FROM
01501   392.         1 TO   FROM TO   FROM TO/3X,47H-------   -------   -------   -----
01501   393.         2-   -------/(5(I6,I4)))
01502   394.     311  FORMAT(21H0THIS SYSTEM CONTAINS,I3,1X,9HFUNCTIONS/)
01503   395.     312  FORMAT(12H TIME ZERO =F9.0/12H TEST TIME =F9.0/15H MISSION TIME =F
01503   396.         16.0//)
01504   397.     313  FORMAT(10H TEST DATA/3X, 8HRESTORER,4X,4HRANK/4X,6HTESTED,4X,6HFAI
01504   398.         1LED/(I8,I9))
01505   399.     314  FORMAT(19H0UNIT FAILURE RATES//11X,30H----------FUNCTIONS--------
01505   400.         1--,8X,30H----------RESTORERS----------/2X,5HSTAGE,4X,6HRANK 1,6X,
01505   401.         26HRANK 2,6X,6HRANK 3,8X,6HRANK 1,6X,6HRANK 2,6X,6HRANK 3/(I5,2X,E1
01505   402.         31.6,2(E12.6),E14.6,2(E12.6)))
01506   403.     315  FORMAT(68H0BEFORE TEST,SYSTEM MISSION RELIABILITY (TIME ZERO TO MI
01506   404.         1SSION END)= E12.6)
01507   405.     316  FORMAT(68H0AFTER TEST, SYSTEM MISSION RELIABILITY (TEST TIME TO MI
01507   406.         1SSION END)= E12.6)
01510   407.     317  FORMAT(26H1STAGES IN UNTESTED BLOCKS//)
01511   408.     318  FORMAT(29H1UNITS IN TESTED FAILED RANKS//)
01512   409.     319  FORMAT(30H1UNITS OBSERVED AS OPERATIONAL//)
01513   410.     320  FORMAT(22H    STAGE   1=YES,0=NO/(I7,I11))
01514   411.     321  FORMAT(22H    STAGE    --RANK---/13X,9H1   2   3/(2I7,2I4))
01515   412.     322  FORMAT(24H1SUBSYSTEM PROBABILITIES//)
01516   413.     323  FORMAT(11X,30HPROBABILITY OF OPERATION AT T1,10X,31HPROBABILITY OF
01516   414.         1 OPERATION AT TM,/52X,28HGIVEN THAT OPERATIONAL AT T1/6H STAGE, 7X
01516   415.         2,6HRANK 1,4X,6HRANK 2,4X,6HRANK 3,14X,6HRANK 1,4X,6HRANK 2,4X,6HRA
01516   416.         3NK 3//(I4,5X,3F10.6,10X,3F10.6))
01517   417.     324  FORMAT(28H1UNTESTED BLOCKS RELIABILITY/)
01520   418.     325  FORMAT(19H0UNTESTED BLOCK NO.,I3,31H  CONTAINS THE FOLLOWING STAGE
01520   419.         1S//)
01521   420.     326  FORMAT(I25)
01522   421.     327  FORMAT(55H PRODUCT OF RELIABILITIES OF ALL UNTESTED BLOCKS ABOVE=E
01522   422.         111.6)
01523   423.     328  FORMAT(26H1TESTED BLOCKS RELIABILITY/)
01524   424.     329  FORMAT(17H0TESTED BLOCK NO.,I3,31H  CONTAINS THE FOLLOWING STAGES/
01524   425.         1/)
01525   426.     330  FORMAT(53H PRODUCT OF RELIABILITIES OF ALL TESTED BLOCKS ABOVE=E11
01525   427.         1.6)
01526   428.          END

      END OF LISTING.      0 *DIAGNOSTIC* MESSAGE(S).
PHASE 1 TIME =    1 SEC.
PHASE 2 TIME =    0 SEC.
PHASE 3 TIME =    1 SEC.
PHASE 4 TIME =    0 SEC.
PHASE 5 TIME =    1 SEC.
```