THE UNIVERSITY OF MICHIGAN

INDUSTRY PROGRAM OF THE COLLEGE OF ENGINEERING

A STUDY OF AUTOMATIC SYSTEM SIMULATION PROGRAMMING AND THE ANALYSIS OF THE BEHAVIOR OF PHYSICAL SYSTEMS USING AN INTERNALLY STORED PROGRAM COMPUTER

Franklin H. Westervelt

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy in The
University of Michigan
1960

October, 1960

IP-470

PREFACE

In the relatively short time during which the high speed digital and analogue computers have been in use, many remarkable applications have been made. Particularly in the area of simulated behavior of physical systems the interest has been high and the utility great. While a great many systems have been programmed and investigated, the general application of the digital computer to system simulation has not yet been as widespread as the applications of analog computers. This has been true for many reasons despite the inherently greater flexibility and more positive error control offered by the digital machine. Perhaps the chief reason for this lies in the more difficult encoding of the analysis. The need to reduce each analysis to machine code has already resulted in several levels of machine languages that are designed to assist the user in bringing his problem to the machine.

Since many engineering problems, of which the system simulation is a good example, require extensive analysis prior to the time at which advanced languages can be of assistance, it may be expected that the need to study a variety of simulations and large systems would result in the development of methods to assist the analysis as well as the later computation of results. The approaches to the assistance in analysis have been varied. The range of methods extends from the production of a generalized system program from which a specific system of the same type may be approximated by an interpretive selection to more truly analytical programs capable of producing other programs to simulate specific systems of rather specific types.

This paper treats the development of two techniques to handle a very generalized system simulation. The first technique, referred to here as The Simulator Program, is a procedure, referred to as an algorithm, for producing programs automatically on the digital computer that are simulation programs for very general systems. The second technique, referred to here as The Stepwise Regression Program with Simple Learning, is an algorithm for producing analytical expressions and subroutines for use in representing the performance of the components of systems. The subroutines may be used by the programs produced by the simulator program or by other programs as desired. Together these techniques provide a heretofore unavailable method for undertaking the study of large systems.

The cooperation and direction given me by the members of the doctoral committee; Associate Dean G. V. Edmonson as chairman, Professors B. A. Galler, J. J. Martin, N. R. Scott, C. A. Siebert, G. J. Van Wylen and Mr. R. D. Allen of Consumers Power Company in Jackson, Michigan, has been greatly appreciated.

Special thanks are extended to both Consumers Power Company and Commonwealth Associates, Incorporated for the enabling grant that made the work possible. Thanks are also extended to the many people at the Computing Center and the College of Engineering whose assistance was most valuable. In particular, thanks are extended to Mr. J. S. Squire and Mr. M. P. Anderson for their valuable assistance.

BACKGROUND OF THE STUDY

In order to establish an orientation of the developments presented in this paper, it may prove desirable to review briefly the developments of earlier workers in pertinent areas. The simulation problem has attracted the attention of many workers. The conventional approach taken by most earlier efforts was the construction of a special purpose program designed to encompass as general a description of the system to be simulated as might be feasible. The solution of the simulation of a specific physical system then depended upon that system being representable by some subsection of the more general program. The specific system was usually selected by means of control parameters from the more general program. Experience with such techniques made obvious the inherent difficulty of representing accurately the many variations of the systems that may be suggested for study.

This deficiency led to the consideration of programs[1] that produced other programs which in turn accomplished the desired simulation. In this way, the computer began to be utilized as an analytical aid and was enabled to generate programs from a description of the system and a specified set of physical laws and relationships with which the system may be described. DYANA, for example, was able to generate programs for any system representable by a general network and by the principles contained in Kirchoff's laws and/or D'Alembert's principle.

The task of generalizing the simulation problem remained and the Simulator Program presented in this paper is a workable solution of that problem. Specifically, the generalization allowed by this technique

extends the use of the computer for the analysis of any system describable as a network and whose component parts may be characterized by the application of relationships involving the parameters of the system as determined at the node, or interconnection, points of the network. The nature of the physical laws and relationships required for the analysis does not influence the logical structure of the Simulator. Thus, the programs generated by DYANA are, in effect, members of the set of programs that may be generated by the Simulator. The relationships supplied by the user to characterize the components of his system may then be time dependent or time independent as desired.

The second development presented in this paper concerns the production of expressions for the prediction of the behavior of physical phenomena. The methods of "least squares" and multiple regression have received much attention.[9,10,11,12] The stepwise regression technique of Efroymson as extended by Dallemand offers a powerful technique to assist the Simple Learning developed in this paper to extend the treatment of data representation problems to include all orders of interaction between multiple functions of several independent variables. The stepwise regression analysis provides the independent evaluation required by the heuristic selection mechanism employed by the Simple Learning program to generate the terms to be used in the prediction equation for the data. In this extension of the techniques of "artificial intelligence" many of the objections to the earlier methods of regression analysis have been answered. Previous methods were often forced to make rather drastic simplifications of the statistical models in order to allow the problems to be solved in a reasonable time, even on the largest and fastest computers.

The incorporation of Simple Learning has allowed the regression analysis to gain access to every possible term within the scope of the functions allowed by the user for all orders of interaction in multiple independent variable problems and still obtain the desired equations in practically feasible time. Since the equations generated are often required by the components of the systems treated by the Simulator, extensions of earlier methods were made to cause the production of the resulting equations in subroutine form ready to be used.

It is important to understand that the stepwise regression portion of the analysis may be replaced by other techniques as improved methods are developed and still retain the benefits offered by the Simple Learning methods presented here. At present, however, the stepwise regression analysis as extended here is regarded as a very suitable technique. The extensions include a treatment of the truncation and roundoff errors generated during the analysis and an improved treatment of the constant term when the analysis is conducted with respect to the normal coordinate axes.

The Simple Learning techniques presented are also extended from earlier efforts.[2,3,4,5] The basic principle may be regarded as analogous to learning through reinforcement. By arranging to increment the probability of an action after encountering success and decrementing the probability after failure, earlier workers had indicated the potential ability of a mechanism to simulate learning. Some interesting observations of random mechanisms had also pointed up possible advantages of an initially random mechanism that would gradually become more nearly stepwise in its

action as the reinforcement process took place. Earlier workers also pointed out some of the pitfalls awaiting learning mechanisms. By incorporating the experiences of earliers workers and introducing a "half-life" concept of reinforcement, the mechanism presented in this paper displays rather promising properties while retaining simplicity. The learning mechanism employed by the program is termed "simple" because the modification of each portion of the selecting mechanism is controlled by a single parameter and each modification occurs individually when the success or failure of each portion of the mechanism is determined. The more complex problem of the interrelation of success and failure patterns is not treated by the present mechanism.

Thus it may be observed that the contributions of many workers in many rather diverse areas have served as the foundation upon which the present work was built. In turn, the development and applications of the methods and concepts presented here will serve to further extend this area in the future.

TABLE OF CONTENTS

LIST OF FIGURES

INTRODUCTION

The simulation of the behavior of physical systems is one of the most economically promising uses of the digital computer because the simulation process affords the user an opportunity to examine system performance without requiring a capital investment in the actual hardware of the system. The simulation of a system also is attractive theoretically because of the opportunity extended to conduct controlled investigations and to avoid the "noise" problems associated with actual systems. The simulation process is also attractive in those cases in which some degree of hazard is present since an "explosion" occurs only on paper.

In spite of the obvious advantages of simulation on the digital computer only a relatively small number of physical systems have thus far been studied in this way. The reason for this state of affairs hinges primarily on the difficulty of communicating general system problems to the computer in the form of a procedure capable of producing the desired results.

The work described in this paper attacks this problem by producing two procedures of immediate use in helping to generate system simulation programs for digital computers. The procedures have been coded and verified for the IBM 704 computer but the techniques are more generally applicable. The two procedures are the following:

1) The Simulator Program.
   This procedure produces simulation programs as algorithms in compiler language ready for translation and execution by the machine.

2) The Stepwise Regression Program with Simple Learning.
   This procedure produces predicting equations in subroutine form for the description of the behavior of the components of the system being simulated.

Both procedures produce machine translatable programs automatically, ready for immediate processing by the machine. Taken together the procedures offer for the first time a method of direct communication for general system problems to the machine for analysis and production of algorithms for the simulation of the system. The use of the machine for analysis as opposed to calculation is not yet widespread. The implementation of the techniques discussed in this paper may be of much more general interest in this area. Specifically, the machine must be presented with methods of proceeding with a problem when the best available information is only capable of indicating the relative possibility of success for the alternative paths. Methods of probabilistic choice and mechanisms for altering the likelihood of choice from "experience" also implemented in the development of the procedures. The programs discussed contain workable implementations of these methods. These methods are a first step toward more sophisticated "artificial intelligence" techniques.

In order to understand the problem encountered in the generation of algorithms by a machine, consider the simulation process itself. If a system is to be simulated, the behavior of each component of the system must be related to the other components in such a way that the physical laws and relationships pertinent to the problem are preserved. In the past, the preservation of this consistency was the responsibility of the human programmer. The result of his effort of analysis was a procedure or algorithm

by which he, or the machine, could proceed step by step from the data supplied to the results desired. In general, the analysis of the system to produce this algorithm depends on three kinds of information: 1) The System Definition, 2) The Given or Known Data, 3) The Desired or Unknown Results. The algorithm may be specialized to the defined system and designed to accept the given data and from this information produce the desired results. The algorithm must be so constructed that the solution can proceed from step to step toward the result.

The Simulator Program is designed to produce these algorithms. The same three basic classes of information are supplied to the Simulator. The Simulator Program then carries out an analysis of this information to produce an algorithm capable of the required performance (or, if it should prove to be incapable of producing the algorithm due to insufficient or otherwise inadequate information, error diagnostics are produced). Because of the possibility that several methods may exist that will yield the desired results, some interesting heuristic methods must be employed by the Simulator.

When the program for the particular system has been produced by the Simulator, it is both printed and punched on cards ready for compilation and execution. The produced program will, in general, require many special characteristics of the components of the system to be available to the program. These characteristics are usually functions of one or more parameters of the system. The Stepwise Regression Program with Simple Learning is an extension of earlier stepwise regression techniques to allow consideration of all orders of interaction in multiple independent variable problems.

The classical approach to such a problem may be shown to be quite unmanageable on present (and even projected) computers without drastic simplification. The Simple Learning mechanism employed in this program avoids such a pitfall and allows an accumulation of "experience" to be directed toward the acceleration of the generation of the predicting equation for the desired characteristic. The technique has been employed in many varied problems and has been carefully verified in many known cases. The procedure produces the predicting equation for a given problem together with a complete statistical analysis and a punched card subroutine ready to be used with the simulation program (or by any other program, as desired).

The Simulator Program together with the Stepwise Regression Program with Simple Learning allow the direct simulation of the performance of any system that may be characterized by a general network. The use of these programs to produce specialized programs of each system avoids the loss of accuracy of representation sometimes suffered in attempting to use a general representation in an interpretive mode. Since the information presented to both programs is designed to be fairly easily obtained, the man-hour cost of programming system simulation problems can be greatly reduced. At the same time, revisions of simulation programs to keep pace with design and operating changes in the actual system can be made economically feasible. Finally, the user of these programs during the design phases can allow study of a greater number of possible configurations than was previously practical.

The feature of, perhaps, greatest importance is the opportunity extended to the user to obtain increasingly accurate representations of the actual behavior of a system while studying a simulation of the system. The most recent information available on any part of the system may be incorporated immediately in the simulation by the techniques described here. Thus the problem of revising and correcting a former simulation program becomes quite secondary.

SUMMARY OF RESULTS

Two programs have been produced for assisting the representation of the performance of physical systems on the digital computer. The Simulator Program is designed to produce the analysis of physical systems in the form of an algorithm in machine translatable language. The user of the Simulator Program must supply: 1) The System Definition, 2) The Given or Known Parameters, 3) The Deisred or Unknown Parameters. The Simulator Program then attempts to construct the required algorithm for the problem. In doing so, use is made of a Library of methods pertinent to the system. The library is accessible to the user so that new methods may be easily inserted. The methods are grouped under the heading of Element Descriptions. That is, each possible element that may occur in the system is described in the Library. New elements may be easily added and older descriptions may be revised within the structure of the Simulator Program.

The second program, The Stepwise Regression Program with Simple Learning allows the generation of predicting equations for the behavior of the components of the system in a form required by the simulation program produced by the Simulator. This program allows the consideration of multiple independent variable problems with interactions of all orders allowed. Since the classical approach to this problem is of such magnitude that present and projected computers cannot adequately cope with the solution in many cases, the Simple Learning technique was developed to allow a solution to be made.

The two programs thus allow the simulation of very general physical systems from a rather basic set of information available on the systems. The resulting reduction of man-hours of programming should allow the extension of systems simulation techniques in many areas not previously practical. Furthermdre, the availability of these techniques should allow the study, and thus lead to understanding, of more complicated systems and components than those previously treated.

I. THE SIMULATOR PROGRAM

THE STRUCTURE OF A PROCEDURE TO GENERATE ALGORITHMS TO SIMULATE PHYSICAL SYSTEMS.

In order to understand a procedure that can generate algorithms to simulate physical systems, first consider the nature of the problem. In general, a physical system consists of a collection of components or elements that are inter-connected to each other in various ways.

For example, a typical vibrating mechanical system consistes of masses, springs, dampers, levers and so on. These components or elements of the system are inter-connected to each other to form the desired system. One such inter-connection might be the attachment between a mass element and a spring element, as an illustration.

The behavior of each component is determined by various physical laws and relationships that are determined by the nature of the component. In particular the behavior may be expressed in terms of the values of parameters at the points of inter-connection of the component to the other members of the system.

For the general system there may be a very large number of different components and associated with each component a large number of methods and procedures that allow the performance to be calculated. In order to select those procedures needed to produce a simulation of the system additional constraints must be imposed. These constraints consist of those parameters for which values will be supplied as initial and/or boundary conditions.

For purposes of discussion, suppose that the simulation of a system is regarded as a method or procedure that will allow the calculation of the values of the various parameters belonging to the system. The task of the simulator program is directed at the problem of determining the method of calculation just mentioned. The actual calculations of the values of the parameters will be produced by the program method produced by the simulator program. The simulator program enters the problem as an analytical rather than as a calculational method. It is this use of the computer on the level of producing programs that in turn are used to produce calculations that allows the most powerful applications of the technique. In so doing, the program has assumed a very large burden in the solution of system simulation problems. This in turn will allow the user to study more complicated and more accurately represented systems.

Specifically, the task of the simulator is not to be construed as that of determining all possible values of all possible parameters but rather that of determining the values of specific parameters subject to specific initial and/or boundary conditions. If the procedure for defining the algorithm can be made quite general then the parameters selected for display and the conditions imposed can be made quite general. The problem is always that of determining when a sufficient set of information has been supplied and of producing the procedure when a sufficient set of information is present.

The information requirements are easily set down. The determination of the sufficiency is most difficult. The requirements are: 1) The Definition of the System. 2) The Definition of the Components of the System. 3) The Specification of the Constraints to be imposed on the System.

The Definition of the System consists of the specification of the elements or components of the system and the way in which these components are inter-connected. For the purpose of this discussion let the Definition of the System be complete when all of the components of the system are defined and there are no possible inter-connection points of any component that are not connected to some other point. To secure the completeness it may be necessary to define some components to act as sources, sinks or boundaries.

The Definition of the Components of the System consists of the specification of the methods or procedures by which values of parameters at the various inter-connection points of the components may be found in terms of the values of parameters at the same and other inter-connection points of the same component. Strictly speaking, completeness of the Definition of the Components requires an exhaustive collection of all possible methods of parameter calculation. In other words, every feasible method or technique that can be applied to a given component must be made part of the collection. Otherwise, it is always conceivable that a program will not be generated by the simulator because a technique was omitted from the collection. This will seldom, if ever, be achieved in practice. Usually the most the Definition of Components can be expected to do is to embrace the most generally productive methods.

Since the decision of what constitutes general productivity is at best highly subjective, a procedure charged with constructing a calculational procedure from these methods must not be involved directly with this decision or else its utility is almost certain to be limited by the decision. If possible, the simulator procedure should allow easy extension and/or modification of the Component Definitions independently of the simulator procedure itself. That is, the method of analysis and use of Component Definitions should be independent of the contents of the Component Definitions.

The Specification of Constraints imposed on the System consists of the values of the parameters to be construed as initial and/or boundary (operating) conditions for the system. Completeness of these Specifications is generally dependent upon the completeness of the Component Definitions.

If very complete Component Definitions are available for the system then it is possible that several different sets of values of the parameters can be made to produce a given value of another parameter. For example, in the superheat region for steam the enthalpy of the steam may be found if the values of any two independent properties, such as pressure, temperature, entropy, specific volume and so on, are known. If many methods are available in the component definition for the determination of enthalpy then almost any combination of two parameters will allow the calculation. If only a few methods have been included in the component definition obviously the parameters given as constraints must be so chosen that these methods apply. In other words, if the Component Definitions are very nearly complete, then calculation procedures can be found for almost any set of constraints that may be given. If this is not the case, then the set of constaints must be large enough to include those that are needed for the calculation using the

available methods. In almost every case however there will be some minimum set of constraints required for any given desired parameter.

The interaction between the Constraints and Component Definitions for a given system is extremely complex. The determination of a minimum set of constraints for a given set of component definitions and a given system definition may be of interest in some cases but the general problem cannot usually recognize whether a failure to yield an algorithm to produce a particular result is due to lack of constraints or deficiency in component definitions. Furthermore, when several alternative calculational methods exist at one or more points in a system it is possible that a valid procedure can be found before all possible procedures have been examined.

In addition to the three information requirements previously mentioned, the generation of a specific program must be viewed as a selection of a reduced collection of methods and their arrangement to yield specific values for certain parameters. Otherwise the simulation of a system would be required to be exhaustive and again this is, in general, not possible. To be generally useful a procedure for generating a simulation program should allow for unrestricted specification of desired information and be charged with the task of establishing a method of calculating this information within the framework of the previous three information requirements whenever possible.

The development of an algorithm to accomplish this generation must be concerned with the recognition of conditions in which it can be established that a method of calculation cannot be found. This may seem strange until

it is considered that when a method cannot be found and the condition recognized then it becomes possible to either terminate the attempt or restart the attempt from another direction. These things could not be done otherwise. If the situation can be recognized when further progress cannot be made in generating a program it is then possible to formulate a simple procedure for generating a program.

A program may be said to be "non-extendable" when there exists at least one required parameter that satisfies the following conditions: 1) The parameter is not given as a constraint and 2) there is no method in either component definition of the two components directly involved with the parameter that will yield the value of the parameter. For example, suppose that the current flow is needed through a thermistor and that no method is available that can produce the current flow value in this case. Then, if further direct progress is to be made the value must either be given or an appropriate method must be added to the Component Definition collection. If neither of these things can be done then the progress depends upon finding an alternative chain of methods that avoid the need for the current flow through the thermistor. If there have been no points earlier in the work at which alternative paths could have been chosen then there is no possibility of the extending the method beyond this point. The program is said to be "non-extendable".

If a program is non-extendable and each previously established required parameter could be found in no more than one way, then the problem may be said to be not well posed. A problem is not well posed, in general, when there does not exist any collection of methods to yield all

of the requested information subject to the imposed constraints and the definitions. It should be noted that a problem may be not well posed even when several methods exist at previous stages. The determination of the well posed condition when this occurs may require an exhaustive investigation of all possibilities.

An algorithm that uses the definition of a non-extendable program to generate a simulation program for a system is the following:

1) Check to be certain that the System Definition is complete. That is, determine if there is a Component Definition available for every element of the System Definition and if every attachment point is connected to some other point. (A Complete System Definition may not be correct but it is capable of analysis.)

2) Check each attachment point in an ordered search to locate any point at which there is requested information. A "request" for a parameter may have occurred in one of two ways, a) the value of the parameter may have been desired by the user of the program, b) the value of the parameter may have been required as an input for a method selected previously. A "request" cannot occur in any other way.

2A) If no such point can be found in the entire system an algorithm for the simulation of the program has been found. (If no such point were found in the first search, the problem is trivial but the preceding statement is still valid.)

2B) When such a point is found, remove the request for information by one of the following methods:

2B1) Matching the request with a specified constraint. That is, if a request result has been given as a

constraint value then the value of the requested result is known without any further calculation.

2B2) Finding a calculational procedure that applies at this point that will yield the requested result. If more than one method applies, pick one and indicate that a choice has been made.

2C) Whenever the request is removed by matching with a constraint no new requests are created. Whenever the request is removed by finding a method, the method may introduce new requests for information. When this is true, cause the entire step 2 to be repeated again after completing the current search.

2D) Whenever a request cannot be removed, the program may be non-extendable. Test the choice indicator and a) if no previous choices have been made, the program is not well posed, b) if previous choices have been made, cause the program generation to return to an earlier state, make another choice and try again.

3) Repeat the search indicated by step 2 until 2A is satisfied or until an upper limit of numbers of trials have been exceeded or until the problem is shown to be not well posed.

4) Whenever step 2A is satisfied, the required algorithm consists of the methods found by step 2B2 executed in the reverse order of their determination. That is, the first method determined yields the last request required of the program. The second yields the next to the last and so on. If any method introduces new requests, these results must be found before

the method can be used. This is precisely the situation that will be obtained since the methods to determine these results will be found later, the execution of the methods in reverse order will produce the results before they are needed by the method requesting them.

An essential part of the previous procedure lies in the technique of picking a method whenever more than one method is available. Obviously, if an inflexible selection is made, that is, always choosing the "best" method (no matter how "best" may be defined), repeating the generation when a program has been found to be non-extendable would always lead back to the same point. Therefore, the selection should be made flexible and, in particular, allow equal chance of selection for equally promising methods and occasionally the selection should allow choice of methods not locally "best." Thus the technique must set some scale by which the characteristic "equally promising" and, in fact, the degree of "promise" can be measured. Many such scales could be specified. One scale that is easily determined and contains some measure of the "promise" characteristic is the ratio of the number of useful results produced by a method to the number of new information requests the method will make. Since the objective is to eliminate the requests by finding methods that require only constraint information, such a scale would place greater weight on methods that make the smallest number of new requests, but would also consider the number of useful results yielded.

For example, a method that requires one new result while yielding one requested result is equally promising when compared to a method requiring four new results to yield four requested results. However, a method

yielding four requested results and requiring only two new results would be scaled twice as promising as either previous method. On this scale, a method that produces any results without requiring any new results would automatically be selected. Also a method that produces no results is automatically rejected. The important point to be understood is that the selection cannot be fixed so that the method of greatest finite weight is always selected since it is possible that one of the parameters that this method would require may not be capable of calculation due to incompleteness of Component Definitions or Constraint Specifications while a method of less weight may avoid this difficulty. However, the methods of greatest weight should tend to be selected if the program is ever to be finished. The result of these considerations is to produce a selection method that operates probabilistically in the choice.

The implementation of this procedure in the form of a program for the digital computer requires two tasks to be performed: 1) The creation of an artificial language to allow communication of the information concerning the system between the human user and the machine program, 2) the preparation of the foregoing simulator procedure as a program capable of accomplishing the translation of the artifical language into a simulation program. Since the first of these tasks is strongly associated with the human and his simulation problem, while the second task may, for the casual user, be regarded as a problem removed from his immediate consideration, the discussion of the implementation of the simulator is divided into two parts along this division. Of course, the second part is vital to the use of the first but its operation is of concern to a relatively small number of people

by comparison. It must be understood, however, that the generation of the simulation programs is accomplished by the translator. The translator is thus the procedure of greatest importance in the solution of the simulation problem on the machine.

IMPLEMENTING THE SIMULATOR

Communication of the System Information to the Program

Communication of the information concerning the system to be simulated to the simulator program is the first, and for the simulator user, the most important, step in the generation of a program to simulate the system. This transfer of information is accomplished through the medium of an artificial language that is designed to be reasonably like the user's own and, at the same time, contain a structure that is recognizable by the program so that the information content may be extracted. Thus the user may expect to use familiar alphanumeric characters and standard punctuation symbols in all but a few cases.

Since the user is often not acquainted with the detailed operation of computing machines, some effort has been made to remove restrictions in the formats for source program preparation. (The source program is the collection of punched cards containing the user's system information. The simulator program produces an object program from the source program. The object program is a machine translatable program from which the machine can produce the desired simulation when supplied with data.) In general, the source program may be punched anywhere in columns 1 through 72 on IBM cards. Statements may run over from card to card without requiring special continuation symbols. While most users will tend to place a single statement per card for convenience in checking and correcting source programs, more than one statement may be placed on a card, if the user so desires. Statements are terminated with a period (decimal point symbol) as in conventional

writing. In a few cases, notably in Element Descriptions, where the user wishes to convey a specific object program language to the simulator, format restrictions will be imposed and emphasized at that place.

Structure of the Simulator Source Language

Information must be transmitted to the simulator from three basis areas and, if desired, further implemented by a fourth "utility" area. The basic areas are 1) the System Definition, 2) the Input-Output Requirements, and 3) the Characterization of Component Performance. In every problem, the user will be involved with the first two of these areas directly. If the problem requires modification or extension of the libraries on Component Performance, the user will also be involved in the third area. The language requirements for each area are inter-related so that the user may carry over most of the structure from one area to the next.

I. The System Definition

In order to simulate a system, the specific system to be considered must first be separated from the set of all possible systems allowed by the simulator. This requires the communication of 1) the names of the actual components that are found in the system, 2) the way in which these components are attached (connected) to each other.

This information is transmitted to the simulator program by statements occurring within the range of a CONNECTIONS declaration. A Declaration does not perform any calculation but instead prepares the program to receive the information that follows. The range of any declaration begins immediately after the declaration and continues until terminated

by any other declaration or by the end of input data cards. The form of the system definition declaration is: CONNECTIONS.

Connection Statements

Connection Statements are chains of symbolic names transmitting the precise components and attachments and their interconnection to the simulator. The symbolic names are of four types:

1. Element Name

Any six or fewer alphanumeric characters may be an Element Name. The Element Name used in a Connection Statement must either agree exactly with the corresponding Element Description Name for that component or be made to agree by use of a synonym. There are no restrictions as to the order of appearance of alphabetic or numeric characters.

Examples of Acceptable Element Names

PUMP; TRBIN1
6L6; 12AT7
MASS; SPRING; DAMPER

2. Element Identification

Since more than one element of a given kind may be found within a system means must be provided to identify each different element. Since the desired effect is to convey uniqueness of the system, the user may use any six or fewer alphanumeric symbols to identify the elements of the system. If an element is identified, all occurrences of the same element must exactly agree in identification. If an element occurs only once in a system, the element identifier may be omitted.

Examples of Acceptable Identifiers

1; 2; 3
A; B; C
A1; 2B; 3C6
MAIN; SCNDRY

3. Attachment Name

Every element enters the system definition by the way in which it is connected to the rest of the system. That is, an element name cannot, except for unary and binary elements, occur without an associated attachment name. The attachment name consists of six or less alphanumeric characters and must agree with the attachment names given in the Element Description for the element associated with the attachment name. If the user desires the agreement can be obtained through the use of synonyms.

Examples of Acceptable Attachment Names

INLET; EXIT
1; 2
ENTRY3; OUTLET
GRID; PLATE; CTHODE; SCREEN

4. Attachment Identification

If more than one occurrence of an attachment name with its identified element exists in a set of CONNECTIONS statements, an ambiguous situation arises. The attachment, in effect, has been made to several different places with but one physical contact point. The user has the option of defining an element with branching or junction properties to resolve this problem or, if it is more convenient or desirable, the option of writing the element description of the component to allow for attachment identification. The attachment identifier is any alphanumeric name of 6 or less characters. A unique identified attachment of an identified element may occur only once in each program. The acceptable forms are like those of element identifiers. As with element identifiers, the user is free to create whatever symbolic attachment identifiers that may be needed.

The Connective, TO , and Connections Punctuation

The foregoing forms of symbolic names are sufficient to define a unique connection point in a system. Let the following generalized symbolic names be defined:

Let EL1 be any allowable element name.
EID1 be any allowable element identifier.
AT1 be any allowable attachment name.
AID1 be any allowable attachment identifiers.

The following forms of connection points are then allowed:

EL1(AT1) if there is only one element EL1 and only one AT1 on EL1 in the system.

EL1, EID1(AT1) if there is more than one EL1 but only one AT1 on EL1, EID1.

EL1(AT1, AID1) if there is only one EL1 but more than one AT1 on EL1.

EL1, EID1(AT1, AID1) if there is more than one EL1 and more than one AT1 on EL1, EID1.

Let CONN be any of the connection point forms above.

CONN = EL1 (AT1)
EL1, EID1 (AT1)
EL1, (AT1, AID1)
EL1, EID1 (AT1, AID1)

Then a CONNECTIONS Statement is of the form: CONN, TO, CONN.

The connective, TO, establishes the joining together of the connections. The punctuation should appear as written in the definition.

Examples of Acceptable CONNECTIONS Statements

PUMP1, A23 (OUTLET, B), TO, HEATER, 5(INLET1).
6L6, 1(PLATE), TO, XFRM, OUTPUT (TAP1, 3).
SPRING, 15B (END1), TO, LEVER, 6C, (ATT3).

The Special Cases of Unary and Binary Elements

The unary elements (one attachment) and binary elements (two attachments) allow special treatment in writing connections statements. These elements may be written without specifing attachment names since the attachment is immediately obtained form the context. If the full connection statement notation is used, no error will result but some saving in programming will be lost. The simulator program will assign the attachment names 1 and 2 to binary elements and the attachment name 1 to unary elements. The user must take care to use these names if reference is made to these elements using the complete notation.

Examples of Connection Statements with Unary and Binary Elements

```
PUMP1, 16(OUTLET), TO, PIPE, 23, TO, HOTWEL.
2N133, 1(CLLCTR, 1), TO, CAP, 3, TO, 2N132, L(BASE, 1).
```

II. The Input-Output Requirements

In addition to the System Definition, the user must state the information requirement to be imposed on the system. That is, the parameters for which values will be supplied as data for initial and/or boundary conditions and the parameters for which the user expects the program to produce values must be stated to the simulator. These parameters are listed under one or the other of the input-output declarations: 1) INPUT PARAMETERS. 2) DESIRED RESULTS. The user must give the source program the appropriate declaration followed by a list of the pertinent parameters. The list of parameters consists of 1) the name of the parameter, 2) the symbolic location of the parameter in the system.

Parameter Names

Parameter names are 6 or fewer alphanumeric characters of which the first character must be alphabetic. These names must agree exactly with the parameter names used by the element descriptions. Synonym modification is not allowed.

Examples of Acceptable Parameters (with Associated Connections)

PRESS (SUPHTR, 1(EXIT)).
VOLTS (6SN7, 3(GRID)).

Specification of More Than One Parameter at a Point

More than one parameter may be specified at a point by giving a list separated by commas and followed by the point designation. For example:

FLOW, PRESS, TEMP (TRAIN 1, 1(INLET)).
FORCE, VELCTY, ACCLRN (LINK, 3(END1)).

Parameter Range

The range of the input-output requirements of a parameter is established by the point designation. If the point designation is EL1, EID1 (AT1, AID1) the parameter requirement will have the range of exactly one point. If the designation is EL1, EID1 (AT1) and there is more than one AT1 on EL1, EID1 in the system definition, the range will be for all AT1 on EL1, EID1. If the designation is EL1 (AT1), the range is for all AT1 on all EL1. In general the range is defined to cover every connection bearing the designation. This concept is very useful for writing parameter input-output requirements but may trap the unwary user. (If for example, the user gave the designation EL1, the effect is to place the parameter requirement at every attachment on every occurrence of EL1.)

III. Characterization of Components

The simulator must have precise information concerning the way in which the physical laws or relationships are to be treated for each component in the system. Fundamentally, the problem is one of allowing the program analytical access to a large collection of possible methods. The methods are catalogued as to the input information required and the output results produced. The simulator program then searches the catalogue for the most appropriate methods to use in the generated program.

If the library of component descriptions is complete for a given system, the user will obtain a program to simulate the performance of the given system after supplying only the System Definition and the Input-Output requirements. If the component library is inadequate for any reason the user must then supply additional information concerning the component. This is done by using the declaration "ELEMENT DESCRIPTION." followed by a collection of assertions and statements conveying the information.

Assertions Within Element Descriptions

An Assertion, like a declaration, conveys special information to the simulator program but unlike the declaration does not terminate the scope of the declaration. The "ELEMENT DESCRIPTION." language has four assertions and two forms of statements.

Element Name Assertion

The element name assertion defines the six or less alphanumeric symbolic name by which the description will be recognized. This is the true

name of the element description. Let ELNAME stand for any symbolic name, then the element name assertion is: NAME OF ELEMENT ELNAME.

Examples of Allowable Element Name Assertions

NAME OF ELEMENT PUMP.
NAME OF ELEMENT 2N133.
NAME OF ELEMENT SPRING.

Parameter Scope Assertion

So that an attachment may be identified, the concept of parameter scope must be implemented. The parameter scope concept classifies parameters into Broad Scope parameters and Narrow Scope parameters by applying the following rules:

A parameter is a Broad Scope parameter if when considering this parameter at an identified attachment it is true that when the value of the parameter has been established at any one of the identified attachment points it has automatically been established for every other identified point of that attachment.

A parameter is a Narrow Scope parameter otherwise. In particular, a parameter is a Narrow Scope parameter when a requirement for a value of this parameter at an attachment point automatically requires individual determinations of the value of the parameter at each identified point of the attachment.

The user may declare a parameter to be of Broad Scope with the assertion: BROAD SCOPE PARAM1, PARAM2, PARAM3. where PARAM1, PARAM2, PARAM3 stand for any symbolic parameter names.

The simulator program will assume a parameter to be of Narrow Scope unless otherwise specified in every component description using the

parameter contained in either the Permanent or the Temporary Library. A parameter not used by an Element Description is thus excluded from this assumption. Failure to properly assert the scope of parameters may result in failure to generate programs that should lie within the scope of the simulator, but programs generated will yield correct results even though redundant calculations were programmed.

Examples of Parameter Scope Assertions

BROAD SCOPE PRESS, TEMP.
BROAD SCOPE VOLTS.

Typical broad scope parameters are pressure, temperature, voltage, since if these parameters are established at any identification of any attachment all other identifications of the same attachment must have the same value for the parameter. Typical narrow scope parameters are flow, current and similar parameters since the value for the attachment requires the values at every identification of the attachment.

Library Status Assertion

The elevation of an Element Description to Permanent Library Status should be made only after the Element Description is throughly checked and very generally useful. When the user feels that these requirements are met the status may be made permanent by giving the assertion: PERMANENT.

After an ELEMENT DESCRIPTION is entered in the Permanent Library it may be removed only by rewriting the entire Permanent Library. Once entered in the Permanent Library the Element Description does not have to appear with the source program deck to generate programs using the element.

The preceeding three assertions may be made in any order but if given, must follow the Element Description declaration, and preceed the first Statement Collection assertion for a given element description.

Statement Collection Assertion

The assertion Statement Collection prepares the simulator so that the statements following the assertion will be processed to form the element description capability. The statement collection assertion has the form: STATEMENT COLLECTION.

Collection Capability Statement

Immediately following the statement collection assertion, the collection capability is stated. The capability language conveys the input parameter requirements and the result capability of the collection. The parameters are given in exactly the parameter language form of the Input-Output requirements. The Collection Capability requirements add three special words to the simulator language; 1) WITHOUT, 2) THEN, 3) ESTIMATE.

The Connection Implication Then

The word then set off by commas separates and identifies for the simulator the output results of a statement collection. This may be best illustrated by example.

Suppose the user wishes to convey to the simulator the capability of an element description that would allow the determination of enthalpy at a point if pressure and entropy were known. This capability might be expressed: PRESS, ENTRPY(OUTLET), THEN, ENHLPY(OUTLET).

A capability statement may involve any number of input parameters and any number of output results but only one capability statement may be given by the user for each statement collection.

The Restrictive Without

To avoid the problems associated with having many similar element descriptions for components that are basically alike but have different attachments the user is permitted to restrict a Statement Collection to apply only to those elements of the type that are without certain attachment points. For example, this allows one element description to be written for a turbine stage and treat both stages with an extraction point and stages without an extraction point. The collection handling a turbine stage without an extraction might use the capability statement:

FLOW (INLET), WITHOUT (EXPT), THEN, FLOW (OUTLET).

Clearly the word without conveys the information required to prevent the use of the statement collection that would follow the capability statement in the case of the turbine stage with an extraction point.

ESTIMATE, The Iterative Solution Collection Indicator

Often physical systems are simulated most conveniently through iterative algorithms. That is, the program is so structured that an initial estimate is improved by repeated calculation. If the user wishes to present a collection of MAD statements to the simulator library to allow the use of such a technique, the form is such that useful results are apparently produced without requiring any input information. Such a collection should be used only if the parameter produced by the collection has been calculated

independently by some other method. In order to inform the simulator that a statement collection is iterative in form the word ESTIMATE is given followed by the list of parameters to be estimated (and later calculated). When this is done, the simulator will allow the use of the iterative collection only when the parameter has been calculated by some other means later in the program.

An acceptable capability statement for the ESTIMATE control word is: ESTIMATE, FLOW (EXTRCT).

Restricted Collections

Certain types of statement collections that are extremely useful in writing element descriptions are such that if they occur in a program they may not be used more than once at any given attachment point. As an example of such a collection consider the continuity equation for mass flow applied at a branching junction. Suppose, for illustration, that there is a single inlet stream designated at the attachment INLET but five outlet streams at the attachments EXIT, 1; EXIT, 2; EXIT, 3; EXIT, 4 and EXIT, 5. If the flow were found at the inlet and at exits 1, 2, 4, 5 then continuity allows the flow at 3 to be found by the relation:

$$\mathrm{FLOW}_3 = \mathrm{FLOW}_{\mathrm{INLET}} - \sum_{\substack{i=1 \\ i \neq 3}}^{5} \mathrm{FLOW}_{\mathrm{EXIT},\, i}$$

Clearly this is a most useful collection form but its use must be restricted to one occurrence at any given attachment. Of course, the collection may be used once and only once at any pertinent attachment point in the system and, therefore, the collection may appear several times in a program, each time at a different attachment.

The collection capability statement for this type of collection uses an identified attachment name for at least one of the attachment names that apply. The capability statement for the preceeding illustration is:

FLOW (INLET), FLOW (EXIT, i), THEN, FLOW (EXIT)

or the equally correct forms

FLOW (INLET), FLOW (EXIT), THEN, FLOW (EXIT,j)

or

FLOW (INLET), FLOW (EXIT, i), THEN, FLOW (EXIT,j)

Note that the symbols i and j are completely arbitrary and therefore open to the user's choice.

Only one form is not recognized as meaningful by the simulator. That form omits both occurrences of the identified attachment on the opposite sides of the implication THEN. The previous illustration written incorrectly is:

FLOW(INLET), FLOW(EXIT), THEN, FLOW(EXIT)

This form effectively says that the exit flow is known if the exit flow is known. This is not meaningful to the simulator because of the apparent redundancy.

The Collection of Statements

Immediately following the capability statement the user must supply the program statements that will produce the results claimed. This simulator program adopts the Michigan Algorithm Decoder language as the medium for the program statements. The user must write the Statement Collection conforming exactly to the format and coding restrictions and conventions of that language. Complete details and manuals for

the M.A.D. language are available from the Computing Center of The University of Michigan. It is presumed here that the user is familiar with the M.A.D. language.

The simulator program allows the complete structure of the M.A.D. language to be employed in implementing statement collections. Two special symbols are the multiple punch symbols plus zero ($\overset{+}{0}$) and minus zero ($\tilde{0}$). The plus zero is punched by depressing and holding the multiple punch key on the IBM 026 keypunch and then striking the plus sign and the numeric zero. The resulting combination of holes in the IBM card is 12-0. (The equivalent 12-2-8 combination will not be read properly by the peripheral 714 Card Reader.) The minus zero is produced by depressing and holding the multiple punch key and then striking the minus (not the dash) and the zero. The resulting combination of holes is 11-0. (The equivalent 11-2-8 will not be read properly by the 714 Card Reader).

These symbols function as special brackets or parenthesises for the simulator. Since the user should have no need for these symbols in his M.A.D. statements their use is specifically restricted to convey information from the M.A.D. statements to the simulator.

Meaning and Use of the Symbol $\tilde{0}$

The symbol $\tilde{0}$ is used to delimit two types of statement segments; 1) function substitutions, 2) floating statement labels.

The function substitution use allows modifications to be made in the actual program generated by simulator at the time of the execution of the generation. For example, suppose that the element description for the element TRBINE has need for the stage efficiency of the turbine stage and

the different turbine stages require different functions to describe the efficiency.

The statement collection might be written:

$$EFF=\bar{0}ETA\bar{0}.(\overset{+}{0}\ FLOW(INLET)\overset{+}{0})$$

and at execution of the program, if this collection were required a check would be made to determine whether a substitution of names was desired for ETA. Thus the program might be made to produce ETA1 for ETA when stage one used, ETA2 for ETA when stage two is used, and so on. If no substitution is given at execution time, the program will use ETA. The $\bar{0}$ symbols are deleted from the object program.

The floating statement label allows the use of statement labels in statement collections. Since an element may appear any number of times in a system and the same statement collection might therefore occur any number of times in a program simulating the system, the user must not use fixed statement labels within a statement collection. If this were done the program generated might be ambiguous. Floating statement labels allow the simulator to generate unique statement labels and thus eliminate any ambiguity in labels.

The user wishing to use a floating statement label writes:

$$\bar{0}**XXXXX\bar{0}$$

Where the X represent any six or less alphanumeric character statement label the user may care to use in his statement collection. The floating statement label is initiated by the $\bar{0}$ followed by <u>two</u> asterisks (*) and is closed by the $\bar{0}$. The simulator will generate a unique statement label, replace the entire floating statement label by the unique label and use

this unique label whenever the same floating statement label is found in this local statement collection. Should the collection appear again in the object program another unique statement label will be generated.

Example of Floating Statement Label

```
      THROUGH Ō**AŌ, FOR I = 1, 1, I.G.10
      .
      .
      WHENEVER T.G. TMAX, TRANSFER TO Ō**AŌ
      .
      .
Ō**AŌ  CONTINUE
```

The Meaning and Use of the Symbol $\overset{+}{O}$

The symbol $\overset{+}{O}$ is used to delimit the parameter and attachment names for which the simulator must generate unique variable names of 6 or less alphanumeric characters. If it is possible to extract the first three characters of the parameter name and produce a unique symbol, this will be done. In this way the mnemonic significance will be preserved so far as possible. If conflicts occur, the simulator will generate a unique symbol for the parameter and use this symbol throughout the object program. In generating the unique symbol the procedure is to first build up a three non-blank alphabetic character symbol and if conflict still exists modify this symbol with probability 0.10 of changing the first character to a random alphabetic, probability 0.30 of changing the second character to a random alphanumeric, and probability 0.60 of changing the third character to a random alphanumeric. The process continues until a unique symbol is generated. A maximum of 70 different parameters may occur at each attachment in any system to be simulated.

Let PARAM represent any true parameter name; ATTCH represent any true attachment name; AID represent any attachment identifier, then the following forms are allowed within the scope of a pair of $\overset{+}{0}$ symbols:

$\overset{+}{0}$ PARAM (ATTCH, AID) $\overset{+}{0}$

$\overset{+}{0}$ PARAM (ATTCH) $\overset{+}{0}$

$\overset{+}{0}$ (ATTCH, AID) $\overset{+}{0}$

$\overset{+}{0}$ ATTCH, AID $\overset{+}{0}$

$\overset{+}{0}$ (ATTCH) $\overset{+}{0}$

$\overset{+}{0}$ ATTCH $\overset{+}{0}$

No other forms are permitted within the scope of $\overset{+}{0}$.

The attachment within the scope of the $\overset{+}{0}$ will cause a unique three digit attachment point number to be determined from the System Definition. The unique parameter name code and three digit attachment code are combined to form a six character variable name for use by the M.A.D. translator. In case no parameter occurs in the $\overset{+}{0}$ scope, as in the latter four allowable forms, the parameter code is generated as three blank columns. The user may use this feature in whatever way may be logically useful in his statement collection.

Because of possible ambiguity, the user may not use more than one identified attachment with a given statement. Any number of occurrences of the same identified attachment may appear with the same or different parameter names within the same statement. When an identified attachment is encountered, the simulator will produce a copy of the statement for every identified attachment occurring in the System Definition for the identified element concerned with this statement collection.

The only exception to this rule occurs when a statement collection involves an identified attachment name agreeing exactly with the point of attachment in the system that caused the collection to be chosen. For example, suppose that the point OUT is identified on some element and that a parameter, say PRESSR, occurring at OUT has caused a collection to be selected in which the flow at OUT is treated for identified attachments named OUT. In this case, a copy of the statement will be produced for every identified attachment OUT except the current one for which the parameter PRESSR was required. The "all except the current point" rule thus becomes "all points" if the current point is not of the same name. Furthermore, if the attachment name within the $\overset{+}{0}$ scope is <u>not</u> identified the rule is to use the current point designation if the name agrees with the attachment name, otherwise use the first occurrence of the attachment name on the current element encountered in the System Definition.

These rules allow the user of the Element Description Library to write simple but very powerful statements involving identified attachments. Summarizing, the user may:

1. Write a collection referring to all identified attachments of an element.

2. Write a collection referring to all identified attachments except the current one, if the current point name agrees with the identified attachment.

3. Write a collection referring only to the first occurrence of an attachment name or to the current point if it has the same name, with priority awarded to the current point.

Examples of the Use of $\overset{+}{0}$.

```
  EXECUTE  TR BIN1.(+OPRESS(INLET)+O,+OENHLPY(INLET)
1 +O,+OPRESS(OUTLET)+O,+OENHLPY(OUTLET)+O,-OETA-O
2 FLOW(OUTLET)+O
```

The numeric 1 and 2 in column 11 are continuation marks exactly as in the usual M.A.D. language.

The following group of statements might be used to sum the current flow at an attachment that may be identified.

```
CURRNT =0.
CURRNT =+OCRRNT(BASE,1)+O+CURRNT
```

If K other components were attached to BASE by using attachment identification in the System Definition the result of the two previous ELEMENT DESCRIPTION statements will be K+1 object program statements that will produce the total current flow at the BASE attachment.

All other characters occurring outside the scope of $\overline{0}$ and/or $\overset{+}{0}$ symbols, including blanks, are automatically passed directly to the object program. Correct M.A.D. card formats are automatically produced, as are continuation cards if needed. Remark cards are automatically produced, as in M.A.D. by placing a character R in Column 11.

End of Element Description Declaration

As many statement collections can be given as may be required to completely state the performance characteristics of the component. When the description has been completed the process is terminated by giving the declaration: DESCRIPTION FINISHED.

This declaration must be given. Failure to do so will cause a simulator error that will finally throw the job off the computer.

Example of an Element Description

The following element description is intended to illustrate the Element Description Language and would undoubtedly require more capability to be of general use. The increased capability may be obtained by adding as many more statement collections as needed.

```
ELEMENT DESCRIPTION. NAME OF ELEMENT TRBIN. BROAD
SCOPE PRESS, TEMP. PERMANENT.
STATEMENT COLLECTION.
FLOW (INLET), FLOW(EXPT), THEN, FLOW (EXIT).
     FLOW1=0
     FLOW1=FLOW1+ FLOW (INLET, 1)
     FLOW1=FLOW1- FLOW (EXPT,1)
     FLOW(EXIT)=FLOW1
STATEMENT COLLECTION.
     PRESS(INLET) FLOW(EXIT), THEN, PRESS(EXIT),PRESS(EXPT).
     PRESS(EXIT)=PRESS(INLET)*PRATIO.(FLOW1(EXIT))
     WHENEVER FIRST, READ FORMAT DATA(1), PDPOEXPT
     PRESS(EXPT)=(1.-PDPOEXPT)*PRESS(EXIT)
STATEMENT COLLECTION.
PRESS, ENHLPY, FLOW (INLET), PRESS(EXIT),
THEN, ENHLPY (EXIT), KWH(SHAFT).
     FLOW=0.
     FLOW=FLOW+FLOW(INLET,1)
     EXECUTE TRBIN1.(PRESS(INLET),ENHLPY(INLET)
    1 ),PRESS(EXIT),ETA.(FLOW1), FLOW,  ENHLPY
    2 (EXIT),KWH(SHAFT))
       EQUIVALENCE (PRESS(INLET),PRESS(INLET,1))
       EQUIVALENCE(PRESS(EXIT),PRESS(EXIT,1))
STATEMENT COLLECTION.
FLOW(INLET), WITHOUT(EXPT), THEN, FLOW(EXIT).
       FLOW(EXIT)=0.
       FLOW(EXIT)=FLOW(EXIT)+FLOW(INLET,1)
DESCRIPTION FINISHED.
```

IV. Utility Declaration

The foregoing three areas of information constitute a necessary and sufficient amount of information for the simulator program to accomplish the generation of programs. There are some additional features of the simulator language which are not strictly essential but which allow the user to produce better programs in some cases and to produce programs more easily in others. Still other declarations are for use in "housekeeping", that is, for making the job of Library Maintenance easier. These declarations and their associated statements are called the "utility" declarations. The declarations are:

1) FUNCTION SUBSTITUTIONS.
2) SYNONYMS.
3) NEW ELEMENT TAPE.
4) NEXT SET OF DATA.

FUNCTION SUBSTITUTIONS. Declaration

When the author of the library Element Descriptions so desires the descriptions may be written so that minor changes can be made at the time of the program execution. The proper method of writing this capability into Element Descriptions was treated in that section. The remaining task is that of allowing the user to exploit this capability in his programs. This is done by giving a FUNCTION SUBSTITUTIONS. declaration followed by function substitution statements. The declaration form is: FUNCTION SUBSTITUTIONS.

Function Substitution Statements

The function substitution statements convey the substitution to be made and the scope of the substitution to the simulator. Any six or

less alphanumeric character word may be substituted for any other word enclosed in $\bar{O}$ in a statement collection. The form of the statement is defined as follows:

Let NUWORD be any six or less character new word.
OLDWRD be any six or less character old word.

Let EL be any symbolic element name.
EID be any symbolic element identification.

Then the allowable function substitution forms are:

NUWORD, OLDWRD.
NUWORD, OLDWRD(EL).
NUWORD, OLDWRD(EL,EID).

The effect of this statement is to replace OLDWRD when it occurs within $\bar{O}$, and at the element specified, by NUWORD. The scope of the substitution is controlled as follows:

1) For the statement form, NUWORD, OLDWRD(EL,EID).
   The substitution will take place only for statements generated for EL, EID.
2) For the statement form, NUWORD, OLDWRD(EL).
   The substitution will take place for statements generated for any occurrence of EL.
3) For the statement form, NUWORD, OLDWRD.
   The substitution will take place for any object program statements generated.

If function substitution declaration is not used, the original contents of the $\bar{O}$ in the statement collection will be used. If the user misspells the OLDWRD, the substitution will not occur. If the user misspells the NUWORD, the object program will contain the error.

Examples of Function Substitutions

Suppose that the element description TRBIN contains the statement: DELTAH=DELTAH*$\bar{O}$ETA$\bar{O}$.($\overset{+}{O}$FLOW(INLET)$\overset{+}{O}$)

and several components TRBIN occur in the system. The characteristics of

each turbine may be different and the user may obtain different functions to represent these characteristics by using the following statements:

```
FUNCTION SUBSTITUTIONS.
ETA1, ETA(TRBIN,1).
ETA2, ETA(TRBIN,2).
ETA3, ETA(IRBIN,3).
```

When this is done the object program statements will use the functions ETA1, ETA2, and ETA3 in place of ETA.

SYNONYM. Declaration

The synonyms declaration allows the use of different symbolic names in writing connection points. This declaration also may be used to condense the connection point symbol strings to a single word or a few words. Both of these uses are introduced for the convenience of the user. It is not necessary to use synonyms to write programs, but the use of synonyms may be of considerable assistance.

The form of the declaration is: SYNONYMS.

The synonyms declaration is followed by any number of synonym statements.

Synonym Statements

Synonym statements convey to the simulator the true symbolic name or string and the symbols that are synomynous with the true name or string. The only restriction is that the true name or string must be first part of the statement. Equal sign symbols are used to separate the true part of the statement and any synonymous parts. If any portion of a connection symbol string is written as a single zero (0) that portion will be left untouched. Let the following symbols be defined:

Let ELT be any true element name;
EIDT be any true element identification;
ATT be any true attachment name;
AIDT be any true attachment identification;
ELS be any element name synonym;
EIDS be any element identification synonym;
ATS be any attachment name synonym;
AIDS be any attachment identification synonym.

Then the allowable forms of synonyms are:

Type I

$ELT=ELS_1=ELS_2=\cdots=ELS_n.$
$0,EIDT=0,EIDS_1=0,EIDS_2=\cdots=0,EIDS_n.$
$(ATT)=(ATS)_1=(ATS)_2=\cdots=(ATS)_n.$
$(0,AIDT)=(0,AIDS)_1=(0,AIDS)_2=\cdots=(0,AIDS)_n.$

The single zero must be punched as indicated. The substitution of the true name will occur unconditionally for any synonym on the right.

Type II

$ELT,\ EIDT=ELS_1=\cdots=ELS_n.$
$ELT,\ EIDT(ATT)=ELS_1=\cdots=ELS_n.$
$ELT,\ EIDT(ATT,AIDT)=ELS_1=\cdots=ELS_n.$
$ELT(ATT)=ELS_1=\cdots=ELS_n.$
$ELT(0,AIDT)=ELS_1=\cdots=ELS_n.$
$(ATT,AIDT)=(ATS)_1=\cdots=(ATS)_n.$

In the preceeding group, the occurrence of the single synonym symbol will cause the use of the entire true symbol string.

Type III

$ELT(ATT)=(ATS)_1=\cdots=(ATS)_n.$
$ELT(0,AIDT)=(0,AIDS)_1=\cdots=(0,AIDS)_n.$
$ELT(ATT,AIDT)=(ATS)_1=\cdots=(ATS)_n.$

In the preceding group, the occurrence of the synonyms are restricted to apply only to the true attachment names and/or attachment identification associated with the true element name independent of element identification.

Type IV

$0,EIDT(ATT)=(ATS)_1=\cdots=(ATS)_n.$
$0,EIDT(ATT,AIDT)=(ATS)_1=\cdots=(ATS)_n.$
$0,EIDT(0,AIDT)=(0,AIDS)_1=\cdots=(0,AIDS)_n.$

In the preceding group, the synonym stustitution occurs for all elements identified EIDT regardless of the element.

Type V

ELT, EIDT(ATT,AIDT)=ELS,ELDS(ATS,AIDS)$_1$=···=EL,
EIDS(ATS,AIDS)$_n$.

In this type of synonym, the synonymous groups are replaced by the true names in a one to one substitution.

No other synonym forms are allowed.

The utility of these groups is best illustrated by example.

1) Suppose the synonym statement is given: PUMP1=PMP1=PMP=P.

Whenever the user writes PMP1, PMP, or P as an element name in a connections or input-output statement the name PUMP1 will be used as the true name.

2) Suppose the synonym statement is given: PUMP1, MAIN=PMP1.

Whenever PMP1 is used as an element name in a connections or input-outpput statement, the element name PUMP1 and the element identification MAIN will be used as the true names.

3) Suppose the synonym statement is given: PUMP1 (OUTLET,PRIMRY)=(X1).

Whenever (X1) is used as an attachment with PUMP1, regardless of element identification, the symbols OUTLET and PRIMRY will be used as true names.

As the user gains familiarity with the synonym capability, occasionally large reductions in the amount of punching required for connections and input-output requirements may be obtained. But it must be emphasized that this is completely a matter of convenience and does not increase the capability of the simulator.

NEW ELEMENT TAPE Declaration

The declaration NEW ELEMENT TAPE, if given, must precede the entire collection of Element Descriptions that are to be used in the program, and in particular precedes the group of statements known as the Prologue and Epilogue. In this way, new collections of elements can be made and new prologue and epilogue statements can be produced. The form of the declaration is: NEW ELEMENT TAPE.

Immediately following this declaration the user must give a set of prologue and epilogue statements. This collection of statements will be common to every program generated using this element tape. The prologue is charged with bringing in the input parameters, making certain initializations, testing for the completion of the calculation and printing the desired results. The epilogue is charged with transferring the program back to the prologue for testing. The epilogue section is entered automatically at the end of the statements generated to simulate the system. The prologue automatically precedes the statements generated to simulate the system.

Rules for Writing Prologue and Epilogue Collections

The rules for writing the Prologue and Epilogue collections are the same as for Element Descriptions with three exceptions:

1. The symbol $\overset{+}{0}$ when enconnectered for the first and second time triggers the repetative generation of statements containing the input parameters. Multiple copies of the statement will be generated, the copies will differ only in the parameters and the parameters will be grouped by attachment point. After the completion of this task, a complete dictionary of the input parameters will be produced on Remark Cards.

The symbol $\overset{+}{0}$ when encountered for the third time triggers the repetative generation of statements containing the desired results parameters. Multiple copies of the statment are again generated, one statement for each desired attachment point as before. On the completion of this generation, a second dictionary of Remark Cards is produced for the desired results.

This is the only permitted use of the symbol $\overset{+}{0}$ and must occur in the Prologue. The symbol $\overset{+}{0}$ is not permitted in the Epilogue. The use of three $\overset{+}{0}$ symbols thus allows 1) reading in the input parameters, 2) printing of the input parameters for verification, 3) printing the desired result parameters as solutions.

The symbol $\overset{+}{0}$ as used in the Element Description is not affected in any way by the Prologue and Epilogue rules.

2. The minus sign occurring in Column 1 must appear twice in the Prologue. The first occurrence marks the point after which the program has completed testing and is ready to produce the desired results printing. The second occurrence marks the end of the Prologue. Both minus signs in column 1 must appear in the Prologue. The minus signs in column 1 must not appear on Remark Cards.

3. The plus sign, occurring in column 1, must appear once at the end of the Epilogue. The occurrence terminates the processing of the Prologue and Epilogue. The plus sign is located on the last actual executable statement card, and must not appear on a Remark Card.

Typical Prologue and Epilogue Collection

The following prologue and epilogue collection is offered as an example of a generally useful collection. Many of the basic features of this collection would be common to any collection of prologue and epilogue statements. The statements themselves must conform to M.A.D. formats and restrictions.

NEW ELEMENT TAPE.

```
Column
 1        11
           R SYSTEM SIMULATION PROGRAM
           R PROLOGUE BEGINS
  START      READ FORMAT ICARD, NOTRYS
             VECTOR VALUES ICARD=$7I10*$
             TRYCNT=1
             FIRST=1B
             REPEAT=0B
             INTEGER NOTRYS, TRYCNT
             BOOLEAN FIRST, REPEAT
             READ FORMAT WORDS, DATA(1)...DATA(12)
             VECTOR VALUES WORDS=$12C6*$
             READ FORMAT DATA(1),0
             PRINT FORMAT DATA,0
             DIMENSION DATA(12)
             VECTOR VALUES DATA=$1H0$
             TRANSFER TO BEGIN
  BACK       WHENEVER TRYCNT.L.NOTRYS. AND. REPEAT
             REPEAT=0B
             FIRST=0B
             TRYCNT=TRYCNT+1
             TRANSFER TO BEGIN
             END OF CONDITIONAL
             WHENEVER REPEAT, PRINT FORMAT REMARK,
           1 NOTRYS
             VECTOR VALUES REMARK=$18HONO CONVERGENCE IN I5,
-          1 8H TRIALS.*$
             READ FORMAT WORDS, DATA(1)...DATA(12)
             PRINT FORMAT DATA(1),0
             TRANSFER TO START
           R END OF PROLOGUE
-BEGIN        CONTINUE
           R END OF GENERATED SIMULATION PROGRAM
           R EPILOGUE BEGINS
             TRANSFER TO BACK
           R END OF EPILOGUE
+            END OF PROGRAM
```

Program Continuation Declaration

If there is more than one system to be simulated in one approach to the computer, a statement is needed to signal the end of one problem and set signals to return for further problems upon completion of the current one. The declaration accomplishing this signalling is: NEXT SET OF DATA.

This statement must be contained on the card preceding the first card of the next problem. Otherwise, the first card of the next problem will be skipped in processing. If there is no next problem, there is no NEXT SET OF DATA. declaration and return is made to terminate the simulator program.

General Simulator Problem Considerations

The statements, assertions and declarations of the Simulator Language may usually be presented without particular concern for ordering and grouping of the statements. That is, CONNECTIONS declarations and statements may be intermixed with INPUT PARAMETERS, DESIRED RESULTS, FUNCTION SUBSTITUTIONS and SYNONYMS. Only those statements pertaining to the Libraries are somewhat restricted in order. NEW ELEMENT TAPE must precede the Prologue and Epilogue and all ELEMENT DESCRIPTION declarations, assertions and statements. The statements in ELEMENT DESCRIPTION are restricted somewhat. Reference should be made to that section of this paper for the exact restrictions. Beyond this restriction it should be noted that while no error will result to prevent execution of the simulator considerable savings in time of processing can be made by placing all Element Descriptions containing the assertion PERMANENT before those without this

assertion. If this is not done, the Temporary Library must be saved, the new Permanent Library entry made and the Temporary Library restored after each Permanent entry. This is not a fast procedure at best but will be done if required for processing.

The current simulator is restricted in the size of system that can be simulated. The version for 704 Electronic Data Processing Machines with 8192 word core storage, 8192 word drum storage and 6 tapes will accomodate 200 connection statements. Simple revisions for 32768 word core storage machines would accomodate 1000 connection statements. Each connective TO produces one connection statement.

Provisions are made for 125 Element Descriptions. Each element is allowed a maximum of 20 different kinds of attachments. If attachment identifiers are used, no limit is placed on the number of identifiers for each attachment. Each attachment and the system may treat up to a total of 70 different parameter types. That is, each attachment is involved with the same set of parameter types and the total number of different types in the system may not exceed 70. The number of INPUT PARAMETERS may not exceed 200. The same limit applies to DESIRED RESULTS. The number of SYNONYMS may not exceed 400. The number of FUNCTION SUBSTITUTIONS may not exceed 400. The number of card images allowed in one statement generated for the object program is limited by M.A.D. to the first card and up to nine continuation cards. The number of statements in a statement collection as well as the number of statement collections in an element description is limited only by the length of a magnetic tape reel. No practical limitation is expected in this area for some time.

There is no limit on the number of times an element may appear in a system. The only restriction is that each unique element attachment may be used only once. This is a restriction to prevent ambiguity in the system definition and not a size limitation.

The scope of a parameter is assumed to be narrow unless the parameter is specifically defined to be a broad scope in every element description in both libraries that refers to the parameter.

With the exception of the declaration NEXT SET OF DATA and the statements for Element Descriptions the simulator statements may be prepared anywhere within columns 1 through 72 of IBM cards and may run from card to card or contain more than one statement per card. No continuation marks are used but every simulator statement (but not the M.A.D. statements in element descriptions), assertion and declaration terminates with a period (decimal point, punched 12-3-8).

Remark Cards for M.A.D. that will be produced in every object program carry an R in column 11. Remark Cards for the simulator current job carry a division slash / in column 1. Any card with / in column 1 is ignored, but printed, by the simulator.

THE STRUCTURE OF THE SIMULATOR TRANSLATOR

After a language for communication of information concerning a system to be simulated is established, the job of the simulator program remains. That job is the translation of the various statements allowed by the language into an algorithm or solution procedure for the system simulation requested.

This is accomplished by several sections of program. The sections and their function are:

1) Preprocessing

2) Desired Result Reduction

3) Program Generation

The preprocessing phase consists of decomposing, analyzing and regenerating the information from the source program statements in a form more easily handled by the machine. Input Parameters and Desired Results are saved in a very condensed form. Since each attachment point may have up to 70 parameters and these may fall into two groups (input parameters and desired results), each point must retain information on 140 items. Thus 200 attachments require 28000 items to be stored. These items are, fortunately, Boolean constants. In particular, the Boolean constant for an Input Parameter is 1(True) if the parameter has been stated to be given. The constant is 0(False) otherwise. For desired results, the constant is 1 if the parameter is required as an output and 0 otherwise. Since the 704 computer is a binary machine it is possible to identify each of the 36 binary digits in

the 704 word with a specific parameter and thus save the status of 36 parameters in a single storage location. The entire parameter status is compressed into four words for each attachment by the Preprocessing section.

A second task of the Preprocessing section is to generate an image of the system to be simulated within the machine. This is done by forming a connection matrix. This array retains the nature of each attachment point. Each attachment is entered as the joint between two elements. Four items are required to specify uniquely an attachment on an element and thus eight locations specify a connection. The result is an 8 x n matrix, where n is the number of connections in the system. The matrix entries are the true names of the elements, attachments and identifiers either as supplied directly by the connection statements or as replaced by synonyms.

The connection matrix thus generated may be quite disordered so far as efficient processing is concerned. After the matrix is completely entered in the machine a sorting is done using an indirect list address array to arrange the matrix in the order of occurence of the Element Descriptions on magnetic tape. The indirect address lists allow the matrix to remain stationary in memory while the effective order is completely changed. Since the finding of information on magnetic tape is the most time consuming of all the operations every effort is used to save tape movement. The ordering also provides for an effective method for locating all occurences of an element in the connecting array without searching the entire array. This is done as follows:

Let the connection array be denoted:

$$EL_{1L}EID_{1L}AT_{1L}AID_{1L}EL_{1R}EID_{1R}AT_{1R}AID_{1R}$$

$$EL_{2L}EID_{2L}AT_{2L}AID_{2L}EL_{2R}EID_{2R}AT_{2R}AID_{2R}$$

$$EL_{nL}EID_{nL}AT_{nL}AID_{nL}EL_{nL}EID_{nL}AT_{nL}AID_{nL}$$

Where

EL is any true element name
EID is any true element identifier
AT is any true element attachment name
AID is any true attachment identification
and the subscripts iL and iR denote the attachment point occurring to the "left" and to the "right" and the i-th such attachment point.

The ordering procedure is then:

1) Order the matrix by the EL_{iL} (and group each EL and EID) according to the arrangement of descriptions on the magnetic tape. Call this ordering vector "L".

2) Order the matrix by the EL_{iR} in the same way and call this ordering vector "R".

Let i be incremented from 1 through the number of connections, say n. Let Li (Ri) be the value of the i-th location in the L (R) vector. Then the i-th member of the EL_L (EL_R) is found in EL_{Li} (EL_{Ri}). This type of list addressing is known as "indirect" addressing.

3) Construct a vector for the vector L whose values are the locations of the first occurrence of each element addressed through L in the vector R. If no such occurrence exists in R, then insert the negative of the first occurrence of the element in L itself. Call this vector "L TO R".

4) Construct a similar vector for the vector R relating the occurrences in R to L. Call this vector "R TO L".

With these vectors the task is simplified for finding the entire set of occurrences of any identified element. The location of all occurrences is accomplished in the following way: (The method is given for L but is equally valid, with appropriate changes, for R)

1) If L TO R at a point is negative, the element does not occur in R. The first occurrence in L is found by taking the absolute value of L TO R.

2) Each entry in L agrees with the first as long as the corresponding L TO R corresponds to the first L TO R.

3) If L TO R is positive, the first L occurrence is found by going first to the first occurrence of the element in R and then back to L by using the associated R TO L value. The same test as in 2 applies to equivalent identified elements.

4) If L TO R is positive, the value gives the first R occurrence. The succeeding values in R are for the same identified element so long as R TO L for each succeeding value agrees with the first R TO L value.

The remaining task of Preprocessing is to save all input-output requirements and function substitutions for the Program Generation. In addition, should the library complement be incomplete, the Preprocessing must construct the library entries. Each Element Description is saved as two files on the tape known as ELTAPE. The first saves the contents of each collection capability in 80 word blocks. This allows two words for input parameters and two words for desired results at each of the 20 allowable attachments. The words are in the Boolean form previously described. Since two words allow for 72 parameters and only 70 parameters are allowed, the remaining bits are available for special use. In particular, the last bit in the desired result word is used to signal that this capability must apply to elements <u>without</u> this attachment.

The second file contains the M. A. D. statements to generate the capability contained in the first file. The first capability group in the first file corresponds to the first collection of M. A. D. statements in the second file and so on.

Upon completion of the Preprocessing, the status of the storage is as follows:

1) The connection matrix is in and contains only true names. The matrix is ordered and the input-output requirements have been packed in Boolean parameter words.

2) The Element Descriptions are processed and saved in groups of two files per description on tape ELTAPE, with all permanent descriptions first.

3) The function substitutions and input-output parameters in complete notation are saved for the program generation phase on an erasable tape.

At this point, control is passed to the Desired Result Reduction section. This section is charged with the actual generation of the algorithm for simulating the system. The procedure for accomplishing this task is almost the reverse of the usual procedure used by humans in attempting the same task. The human approach, largely because of the extremely large storage capacity of the human brain, is a search that proceeds from the known parameters and is directed toward the desired results. This approach could be implemented in the machine but because of storage limitations may become quite unworkable. The difficulty is that the machine program cannot reject a method until it can be shown to be unnecessary in the program to obtain the desired results. Thus the program would be forced to enumerate all the possible methods available from the input parameters plus the results

of the first set and so on. The number of methods available grows rapidly and if the problem is well-posed the desired results will eventually be encompassed. However, this constitutes an exhaustive search with only a small fraction of the methods actually of use.

Therefore a different approach is used. Essentially, the algorithm is produced in reverse by working from the desired results toward the input parameters. In this way every step generated is necessarily of use in the program. The program is, of course, backward, in that the first statement collection specified is the last one needed and so on but this is easily taken care of by the Program Generation section. The method of production of the algorithm is the following:

1) Inspect the "Desired Result" Boolean words for each connection point in the matrix. Whenever no Desired Result bits can be found in the entire matrix the algorithm is completed.

2) Whenever a connection is found for which results are desired, steps must be taken to satisfy the request for results.

2A) The requested results may be input parameters. If this is so, remove the corresponding desired result bits.

2B) The requested results may occur at any identified attachment and be of broad scope. If this is so and the result (as an input parameter,) can be found at any of the identified attachments, remove the corresponding desired result bit.

2C) If requested results still remain after steps "2A" and "2B", then some additional program must be added to obtain the results.

2C1) Find all of the statement collections for both elements that occur at this attachment point that are useful in obtaining the requested results. That is, ignore any collections that are "without" attachments specified for this element or collections that do not happen to produce any of the desired results.

2C2) Check each useful collection to determine its effectiveness. The effectiveness is the ratio of the number of requested results the collection produces to the number of new requests for results the collection will produce. A new request for results will occur if any of the parameters required by the collection is not an input parameter or already requested by previous statements.

If the number of new requests for results is zero, the collection is always inserted in the algorithm. This collection produces results without requiring any new information.

(The only exception to this rule occurs with the iterative ESTIMATE collection. In this case, no new information request is apparent, however, the ESTIMATE collection is restrained from inclusion in the program until the parameter in question is found by at least one independent calculation method.)

Otherwise, retain the effectiveness ratio as the weight of the collection.

2C3. When all of the collections have been examined for effectiveness and if desired results still remain, select a set of the collections that will produce the requested results.

At this point, the methods in which a parameter could be found using a method only once at a given attachment point are checked and discarded if already used. Otherwise, these methods are simply placed in competition with any other techniques available.

2C3A) First check to be certain that every requested result can be found in at least one way. If any result cannot be so found, the problem may not be well posed. The problem is not well posed if no "branches" have occurred previously in the generation. A "branch" occurs when a choice is made between more than one method of determining a requested result.

2C3B) Whenever there is exactly one method for producing a result, this method must be included at this point in the algorithm. The method is inserted, the results produced by the method have the corresponding bits removed and any new requests occurring anywhere in the matrix have the corresponding bits inserted.

2C3C) After all single method results are taken care of there remain only results for which there are several methods of calculation. Since only one method will be

used for each result the selection will constitute a "branch" in the algorithm generation, if the method selected is not always forced to be the same one.

If one considers the available methods, each with its associated weight, the simulator should tend to choose the method of greatest weight. However, the simulator should be allowed to select the method on the basis of the probability of selection being proportional to the weight. If this is not done, one may anticipate that in some case the method of greatest weight may contain a parameter that is incapable of calculation (considering the input parameter) and therefore the program could not be generated. If, however, the simulator makes the selection probabilistically, the method of greatest weight is most likely to be selected but other methods may be selected in its place. The probabilistic selection is automatically made and the "branch" Boolean constant is set to one. In this way, if later there should arise a case in which no method is available the simulator may make another trial and possibly work out a satisfactory algorithm by having the chance to choose another method at this point. This is a situation in which the locally "best" method is not always the globally "best" but tends to be so.

3) The steps 1 and 2 are repeated over and over. Each time the requested results are satisfied a new set of requests are generated except when the request matches an input parameter. If the problem is well posed then a sequence of methods may be found such that all desired results are satisfied, through the sequence, by input parameters. When this has been done, an algorithm for the simulation of the system has been produced.

The algorithm produced tends to be optimal since at each state the method of greatest weight was most likely to be employed but the simulator cannot, with limited storage, view the generation of the algorithm beyond a single step. Thus occasionally the simulator may produce several steps that might be condensed if more information were available. In particular, it may happen that identical sets of statements may be produced in the algorithm at different stages of the generation. This redundancy is easily detected and the final algorithm will contain only the first occurrence of the set.

The method of probabilistic selection is also used to discard the least likely method should there be found too many methods to apply at a given point.

The method of probabilistic selection for picking an item from a group of n weighted items is the following:

Let $W_i > 0$ be the weight of the i-th item from a goup of n total items.

$W = \sum_{i=1}^{n} W_i$ be the total weight of the group

N_0^W be a random number selected from a uniformly distributed set of random numbers on the interval 0, W.

Then the K-th member of the group of n items will be selected for the smallest K such that

$$\sum_{i=1}^{K} W_i \geq N_0^W$$

N_0^W is most likely to fall in the subinterval such that W_i is maximum but may fall anywhere in the interval. An modification of this method to pick the least likely item (for discard, etc.) consists of defining a new set of weights $\rho = 1/W_i$ and make the selection using ρ in place of W. In particular, it should be noted that equally probable alternatives receive equal chances and every alternative, no matter how small its weight may be, receives some consideration and may be chosen at any time. This method should find many applications in future programs.

Since there is no way to predict either the number of parameters that may be needed at a point or the number of methods available for any parameter it is necessary to allow an extremely flexible storage assignment so that the storage may be completely used. This is done by means of an "associative memory" list for the storage region. This list functions as follows:

1) Associated with each parameter at the attachment point is a storage location whose value is:

1A) Zero if the parameter is not required.

2B) Minus one if the parameter is not required and no method has yet been found to yield the parameter.

1C) Otherwise, the value is a positive integer giving the location of the beginning of the list of methods for this parameter in the "associative memory".

2) Each entry in the associative memory list gives the location of the next (associated) entry. The final entry is denoted by a minus sign.

3) New entries are made by consulting the associative memory list beginning at the zeroth location. The value of this location is the next available location in the memory. An addition to the end of any list is made in the available location, the value that was in this location is stored in the zeroth location and the former list end is changed to refer to the new list end.

4) Whenever a result collection is selected, the storage space is reassigned as available storage by placing the starting location for the list in the zeroth location, and the value formerly in the zeroth location at the end of the list being removed. In this way the entire list is made available with only two storage reassignments.

5) If the capacity of the storage is exceeded before all the parameters have been treated space can be created by selecting the parameter with the greatest number of methods and picking the method least likely using the probabilistic section technique. The location thus chosen is made available by giving its address to the zeroth location and reassigning the preceeding list location to skip this location and refer to the next item in the list.

Upon completion of the removal of all the desired results and those created during the removal of others, the algorithm is completed and written in reverse order on magnetic tape. This type of storage is used because it is not possible to predict the storage required for a program in advance. This is somewhat unfortunate since the program is generated in the form of a "push down" list. A push down list is a list such that each entry occurs at the beginning (rather than the end) of the list and thus moves the former first item to second place, the former second to third and so on. Thus the items are "pushed down" on the list. Removal of items from the list occurs from the beginning of the list with the last item entered. Thus the effective order of the list is reversed. This is precisely the action that must occur in the program generated since the last statement collection found must be the first used in the program and the first collection found is the last one used in the program. The difficulty is resolved by moving the tape backward two records and forward one working from the last record written toward the first.

As this process is begun after the completion of the algorithm the simulator is at this point generating the simulation program using the Program Generator. The first output of the Program Generator is the Prologue and its associated input-output statements. The Prologue is followed by the program. The algorithm is stored in a short code giving the connection matrix row number and index in the L vector so that the unique connection could be located by the program generator, the element description name given by position in the element name vector and tagged with a plus (+) sign if the element involved was the left most (and a minus (-) sign if the

right most) and finally the statement collection number. The program generator section moves to the second file in the desired element description and next to the appropriate statement collection. Finally the statement collection is processed and produced both on cards and in print to form the desired simulation program. The rules by which the processing takes place have been stated in the section describing Element Descriptions. Briefly, the M. A. D. statements are written using floating statement labels, and special codes for function substitutions and for parameter and attachment codes. The Program Generator assigns unique fixed statement labels for the floating statement labels. Any function substitution is checked for possible modification. If a substitution has been requested, the substitution is made, otherwise the original text is retained. The parameter and attachment codes are reduced to a six character variable name code for each parameter-attachment combination occurring. Some of the six characters may be blanks. Non-identified attachment parameters are immediately coded and inserted in the output statement. Identified attachments cause multiple copies of the statement to be generated. One copy is made for each different identifier. To avoid possible embiguity, only one attachment may be identified in each statement. However, this attachment may occur any number of times with any number of parameters within one statement. In this way the effect of a special junction element is produced without specifically requiring such an element.

As noted in the Collection Capability section, there is one exception to this rule. Namely, when an attachment is identified and the current point in the connection matrix agrees exactly with this attachment name, a copy of

the statement is produced for all occurrences except the current one. Finally, if an attachment is not identified the current attachment point will be selected if the name agrees with the name occurring in the collection statement. Otherwise, the first occurrence of the name is used in the code.

Upon completion of the program generation the control is returned to the Preprocessing Section to process any other system simulation problems that may be waiting. Since the output of this program is a program in M. A. D. code and on punched cards the simulation program may be used as it is produced or modified easily before using it to simulate the system. To use the program as it is generated, the user need only supply the data and special subroutines needed and the special cards needed by the executive system for the data processing system. The program will be translated into machine code and executed using the data supplied.

II. STEPWISE REGRESSION PROGRAM WITH SIMPLE LEARNING

The representation of the characteristic performance of the various components of a system is vital to the simulation problem. It is not sufficient to obtain a relation which merely fits the available data, if the relation is to be used for predictive purposes, because such a relation may bear only superficial resemblance to the actual performance at other points. A much more desirable relation would consist of terms suggested by the nature of the physical laws governing the component performance but using only those terms which may be shown to be substantiated by the available data. The Stepwise Regression program was written to establish this relation and produce, in addition to the analysis of the data as just described, the actual M.A.D. statements needed for the predicting equation subroutine.

The use of simple learning by the program allows the program to deal with a much more general solution of the predicting equation problem than has been previously possible. The usual engineering problem consists of many independent variables (pressure, temperature, load, etc.) which affect the behavior of the dependent variable (e.g., efficiency, loss, etc.) that is to be predicted. In addition, these variables are usually found to enter in nonlinear manners, (e.g., raised to powers or roots or even more complicated forms). Also, in the usual problem the dependent variable performance is often affected by interaction between the independent variables and functions of the independent variables. The simplest sort of example of such an interaction is the Perfect Gas relation:

$$PV = MRT$$

In this case, the variables P and V interact so that T may not be determined by a relation consisting of terms using P alone plus terms using V alone but may be found by using a term involving the interaction between P and V.

Thus the size of an engineering problem of several independent variables, each of which may be represented by several functions, grows very rapidly when all possible interactions are allowed. An illustration will indicate the magnitude of the problem.

A common selection of twenty functions for a single independent-variable problem, that is, a problem which may be expressed:

$$Y = \sum_{i=1}^{20} b_i F_i(X)$$

will require about 30 seconds to solve on the IBM 704 using conventional stepwise regression techniques. If an apparently only slightly more complicated problem involving three independent variables, each of which has twenty functions, were to be attempted, considering all interactions, the number of terms to be considered increases from 20 to 9260, the size of the matrix involved grows from 21 by 21 to 9261 by 9261, and the IBM 704 time becomes approximately 2500 machine hours.

It is clear that, without a technique capable of reducing this problem by several orders of magnitude, the general problems encountered in engineering will have to be treated in strongly simplified terms. Indeed, this has been precisely the motivation for earlier linearized system models.

The simple learning mechanism developed for use with the Stepwise Regression Program has been used successfully to produce predicting relations in much less time than required for more conventional methods. The following discussion will describe, first, the Stepwise Regression Program and, second, the Simple Learning mechanism employed by the program. This program represents one of the first applications of "artificial intelligence" in an area of immediate practical interest.

Discussion of Stepwise Regression

The following describes a computer program useful in determining the relationships existing among a group of up to 60 variables or functions of variables at each program pass. Taking one of the variables to be a dependent variable, the program results in a linear predicting equation using the current set of predictor variables or terms and selecting from this set a "minimal" set. The program allows simple learning to occur concerning the most satisfactory types of terms, thereby extending the usefulness in determining equations that take account of possible variable interactions of all orders. The program further allows the generation of equations using either stepwise buildup or stepwise purification at the discretion of the user.

This discussion concerns some extensions carried out by the author of the work originated by Mr. M. A. Efroymson of Esso Standard Research and Engineering Co., and carried forward by Mr. J. E. Dallemand of General Motors Research Staff. The problem considered is that of determining a predicting equation from a collection of data. The method of analysis deals with the situation which arises when data have been

collected on many variables, of which one is regarded as a dependent or response variable and the remainder of the set is regarded as a set of independent or predictor variables. It may be anticipated that the method will be useful in experimental situations involving unknown complicated interactions between many variables and complicated relationships (functions) of the variables. In particular, when the data are already available, or where it is difficult to control variables systematically, or where the conduct of a systematic experiment would disrupt the normal operation of a system too severely, this method will be useful.

Specifically, this method is useful in obtaining answers to questions like the following:

(1) What linear combination of the independent variables, or functions of the independent variables, or interactions (cross products) of independent variables and functions of independent variables best explains the data on the dependent variable?

(2) How good is this relationship (obtained in (1))?

(3) What is the linear relationship between the "best" single independent variable (function of an independent variable, or interaction) and the dependent variable? Also, what is the relation for the "best" two, three, or other subset of the possible predictor terms?

(4) For each subset of (3), how good is the relationship?

(5) What is the smallest set of predictor terms that will make statistically significant contributions toward explaining the statistical variation in the dependent variable? (The user may set the level of significance.)

(6) How good is the relationship in (5) and how good is the prediction?

(7) How much of the behavior of the dependent variable is still unexplained by the equation? (This is the Standard Error of Estimate.)

(8) If there is theoretical justification for suggesting certain terms to explain the behavior of the dependent variable, what is the "best" relationship for this set?

(9) How good is this relationship (8) and what can be done to explain the behavior not explained by the present theory?

I. The Stepwise Regression Method

The Stepwise Regression analysis deals with a set of p independent variables denoted X_1, X_2, ..., X_p and a single dependent variable Y. Let N be the number of observations made on each of these p+1 variables yielding N*(p+1) data in all. The objective of the analysis is to generate a relation of the form

$$Y = b_o + b_1X_1 + b_2X_2 + \cdots\cdots + b_pX_p \quad . \qquad (1)$$

The b_i, i = 0, 1, 2, ... , p are the coefficients or multipliers of the various X_1. It should be clear that one could not distinguish between the previous case of p independent variables and the case of p linearly independent functions of a single independent variable or any other combination of numbers of independent variables and function choices for these variables totalling p terms in all. Therefore, the discussion here treats the problem as if there were p independent variables without loss of generality.

To focus these statements on a physical problem, consider the following: Suppose that measurements have been made of the electrical losses of a hydrogen cooled generator. Figure 1 shows the general behavior of the variables and indicates that at least three factors must be considered. It is assumed that measurements or observations are available of (1) gross electrical load on the generator, (2) hydrogen pressure, and (3) power factor as well as the corresponding electrical loss.

The formal relation (1) might be interpreted as the linear relation:

$$\text{GENLOS} = b_0 + b_1 * \text{GKW} + b_2 * \text{HPRESS} + b_3 * \text{PFCTOR}$$

where

GENLOS = generator electrical loss
GKW = gross generator load
HPRESS = hydrogen pressure
PFCTOR = power factor .

However, Figure 1 indicates that such a linear relation may not represent the actual behavior. More complicated analytical models may be suggested to the Stepwise Regression Program by making appropriate definitions of some pseudo-variables.

Suppose that the pseudo-variables X_i are defined:

X_1 = GKW
X_2 = GKW^2
X_3 = GKW^3
X_4 = HPRESS
X_5 = HPRESS^2
X_6 = HPRESS^3
X_7 = PFCTOR
X_8 = PFCTOR^2
X_9 = PFCTOR^3

and, of course, the list may be longer and as complicated as needed to describe the physical problem. The "standard" types of terms automatically

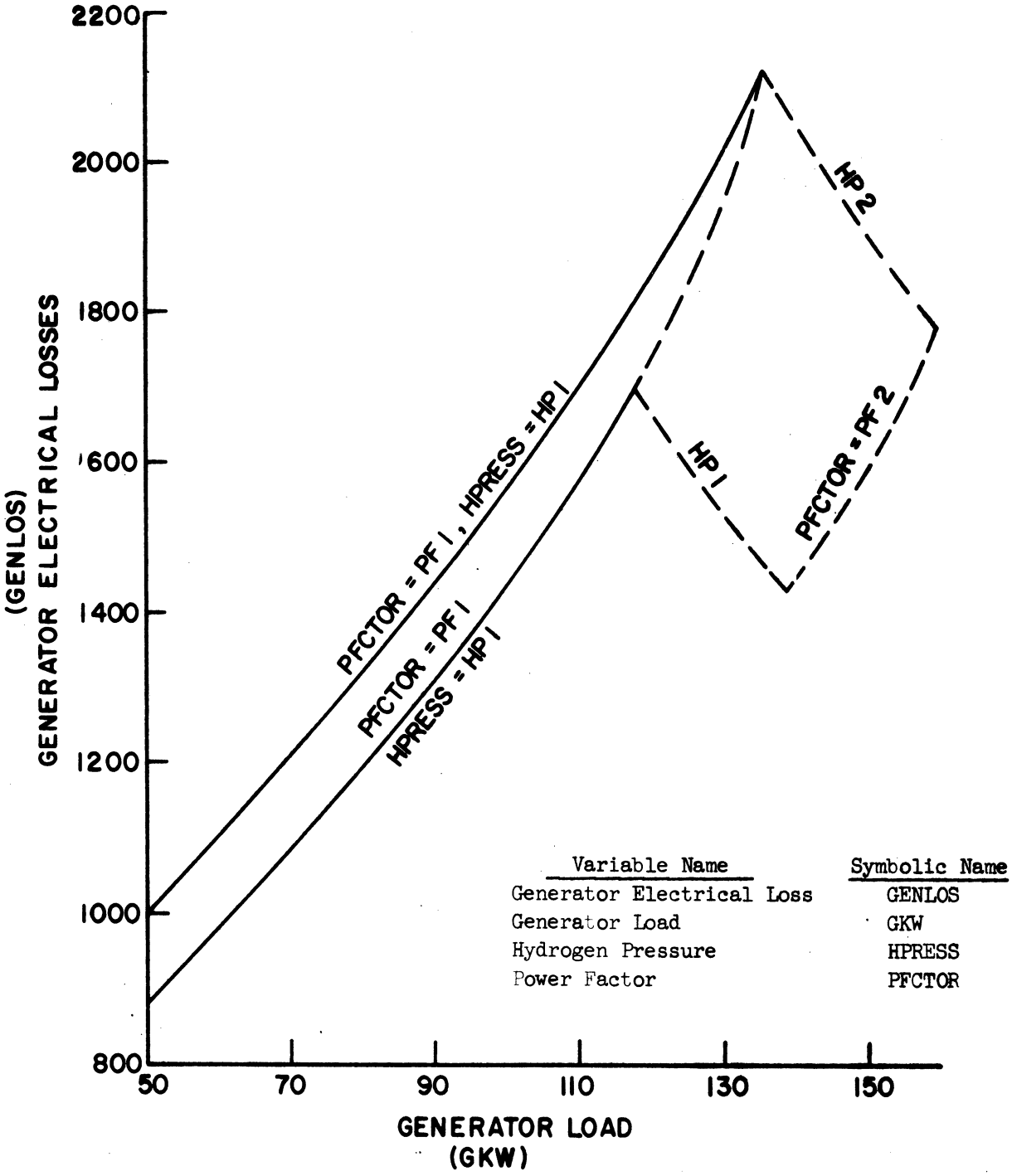


Figure 1. Generator electrical losses as a function of load, hydrogen pressure, and power factor.

available to every problem include integer powers, integer roots, and the reciprocals of these terms. Provision is made to insert any other special terms desired as well (such as logarithms, exponentials, etc.). Then a relation of the form (1) is:

$$LOSS = b_0 + b_1 * X_1 + b_2 * X_2 + \cdots + b_9 * X_9$$

or its equivalent

$$LOSS = b_0 + b_1 * GKW + b_2 * GKW^2 + \cdots + b_9 * PFCTOR^3$$

Again, it often happens that interaction may occur between the variables and the functions of variables. Once again a relation of form (1) may result by defining:

$$\begin{aligned}
Z_1 &= X_1 = GKW \\
Z_2 &= X_2 = GKW^2 \\
&\vdots \\
Z_9 &= X_9 = PFCTOR^3 \\
Z_{10} &= X_1 * X_4 = GKW * HPRESS \\
Z_{11} &= X_1 * X_5 = GKW * HPRESS^2 \\
&\vdots \\
Z_{36} &= X_6 * X_9 = HPRESS^3 * PFCTOR^3 \\
Z_{37} &= X_1 * X_4 * X_7 = GKW * HPRESS * PFCTOR \\
&\vdots \\
Z_{63} &= X_3 * X_6 * X_9 = GKW^3 * HPRESS^3 * PFCTOR^3
\end{aligned}$$

The formal relation (1) is now

$$LOSS = b_0 + b_1 * Z_1 + b_2 * Z_2 + \cdots + b_{63} * Z_{63}$$

or its equivalent

$$LOSS = b_0 + b_1 * X_1 + b_2 * X_2 + \cdots + b_{63} * (GKW^3 * HPRESS^3 * PFCTOR^3).$$

The problem consists of finding those Z's which contribute to the explanation of the dependent variable (LOSS) with sufficient importance, as indicated by the measured data, to allow their retention in a predicting

equation. And, having found the set of Z's meeting the importance criterion, the problem continues to the determination of the best possible estimates for the b's. In this way, a minimal relation is generated which may be used to predict LOSS for given values of GKW, HPRESS, PFCTOR. This relation is automatically generated by the Stepwise Regression Program. In addition, the Stepwise Regression Program produces on punched cards the M.A.D. function corresponding to the generated relation and having any arbitrary function name desired. In this case, suppose that the desired function name is GENLOS. The program would produce the M.A.D. External Function

GENLOS. (X1, X2, X3)

where X1, X2, and X3 are now symbolic names for the arguments GKW, HPRESS, PFCTOR, in machine translatable form ready for inclusion as part of a simulation program (or any other application). Thus one may later write the relation

NETKW = GKW - GENLOS. (GKW HPRESS PFCTOR)-MECLOS

as a M.A.D. statement to be used in a simulation program, and the result will be the net power generated (NETKW). It is clear that no loss of generality has resulted by considering the formal relation (1). It should also be clear that the X terms in (1) may represent either the actual measurements of the independent variables or that they may represent functions of these measurements without requiring any change in technique. In the remainder of this discussion, the symbol X will be used and the meaning may be understood in its most general sense.

The b_i in (1) are determined in such a way that, if one forms the sum of the squares of the differences between the observed values of Y and the predicted values of Y arising from the use of (1), then that sum will

be minimum. Notice that the process of squaring the differences insure that all errors, positive and negative, contribute toward increasing the sum. This is commonly referred to as the method of "least squares."

The importance of the Stepwise Regression method lies in the process of "building" the expression (1) a term at a time, always insisting that the terms be inserted in order of their relative importance to the explanation of the behavior of Y. Furthermore, checks are made continually regarding the continued importance of terms in the equation and only those terms will be inserted into (or removed from) the equation (1) that meet certain significance tests which can be controlled by the user. Thus the final equation will comprise a "minimal" set of terms. Since terms may be removed from the equation as well as inserted into the equation, the method of Stepwise Regression also allows the generation of a relationship by "purifying" an initially large set of terms with very little added burden to the user. Experience indicates that the purification process occasionally produces valuable additional information in certain problems.

A number of statistics are computed before the task of building the predicting equation begins. These statistics may be printed out to help give further insight into inter-relationships in the data and are used by the program for executing the task. Included among these statistics are the mean (average value) for each variable, the standard deviation (a measure of variability) for each variable and the correlation coefficient for each pair of variables. The correlation coefficient measures the linear relationship existing between the pair of variables, and ranges from +1.00 (perfect direct relationship) to 0.0 (no relationship) to -1.00 (perfect inverse relationship).

Figure 2 shows the interpretation of the standard deviation and the mean. If the scatter of data is due to random uncontrollable error, then the Gaussian distribution will model the variability with respect to the predicting equation. Taking the mean or average value to be that indicated most likely by the data, the width of plus and minus one standard deviation will embrace an interval about the mean within which the expectation of the true value is 68%. As indicated by the figure, if the interval is doubled, the expectation grows to more than 95%, and if the interval triples, the expectation is 99.8%. In other words, based on the data measured on the physical component in question, one may expect to encounter a true value of the dependent variable lying more than three standard deviations away from the predicting relation value with a long term frequency of 1 in 500.

Figure 3 illustrates this discussion with respect to a predicting equation. If the predicting equation produces the estimate of the true value of the predicted variable indicated by the central heavier curve, then the bands to either side may be understood to indicate the range within which the true value may be expected to lie with the stated frequencies. Thus a predicting equation with very small standard error of estimate will more accurately represent the true behavior of the variable than will a predicting equation with large standard error of estimate.

I. Generation of a Predicting Equation

Consider a simple example; suppose that an experiment has been made consisting of a set of observations of six variables. Regarding one of the six as a dependent variable and the remaining five as predictor

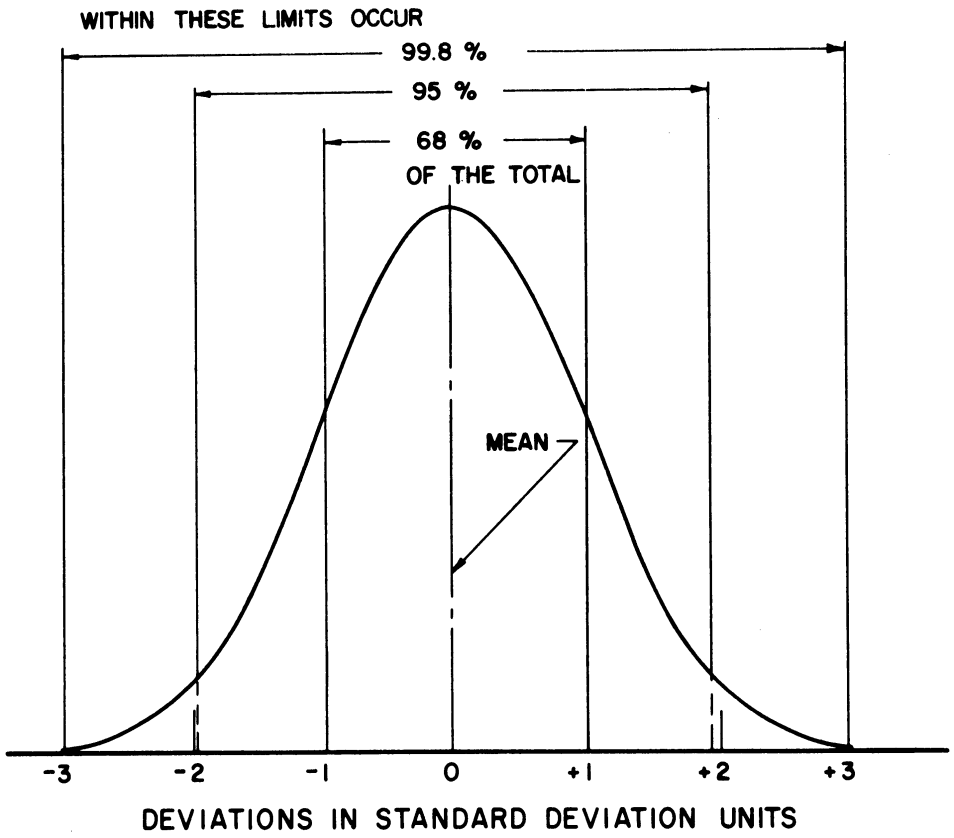


Figure 2. The Gaussian distribution.

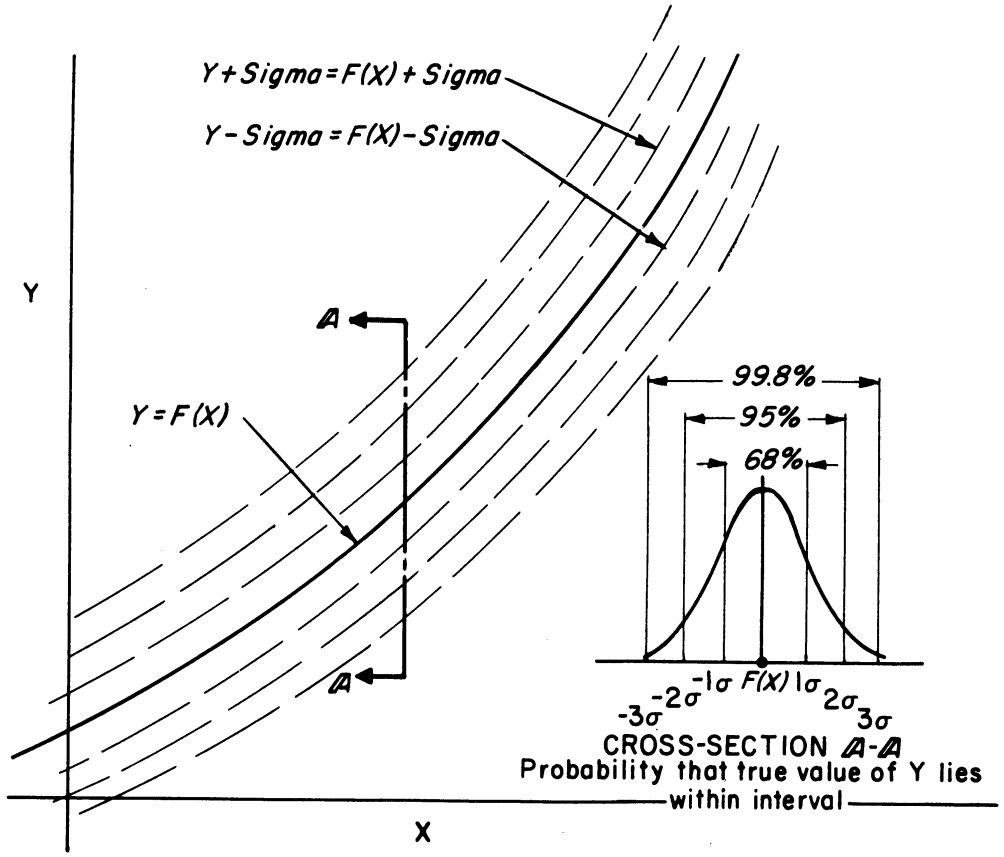


Fig. 3. Predicting equation Y = F(X) as an approximation to true values of Y.

or independent variables, the analysis determines the "minimal" set of variables which may be used in a relation of form (1), where, in this case, $p = 5$.

The first step is to find that variable X_i which best predicts Y. This is done by correlating each of the X_i to Y and the selecting that X_i which has the greatest "correlation coefficient"* in absolute value. If more than one X_i shares the largest value, take the X_i with the lowest subscript i. That is, take the first such X_i encountered. Suppose that in this instance that best i is 4. The first predicting equation is then

$$Y = b_0 + b_4 X_4 \quad . \tag{2}$$

The b_0 and b_4 satisfy the least-squares criterion.

Succeeding steps are of slightly different form. First, the X_i are sorted into two subsets $X_{i,1}$ and $X_{i,2}$. The set $X_{i,1}$ consists of all those variables that are in the predicting equation at the time of sorting. The set $X_{i,2}$ consists of all those variables that are not yet in the predicting equation.

---

* The correlation coefficient is defined as the product-moment coefficient of correlation: Let

$$X_i X_j = \sum_t W_t X_{it} X_{jt} - \frac{(\sum_t W_t X_{it})(\sum_t W_t X_{jt})}{\sum_t W_t}$$

where t = number of observations
n = number of independent variables
$j = i, i+1, i+2, \cdots, n+1$
$i = 1, 2, \cdots, n+1$

Then let

$$\alpha_i = \sqrt{X_i X_i} \quad i = 1, 2, \cdots, n+1$$

and the correlation coefficient r is then

$$r_{ij} = \frac{(X_i X_j)}{(\alpha_i)(\alpha_j)}$$

with the properties

$$r_{ji} = r_{ij} \qquad i, j = 1, 2, \cdots, n+1$$
$$r_{ii} = 1.000$$
$$-1 \leq r_{ij} \leq 1 \quad .$$

For each of the members of $X_{i,1}$ the analysis computes an "importance factor"** which is a measure of the relative contribution of the variable to the predicted equation. The smallest of these importance factors is isolated. If the variable associated with this factor is less important than the user requires for the variable to be retained in the equation, then that variable is removed from the equation before continuing.

The scale used to determine whether a variable meets the "importance" criterion is simply the probability or chance that the user is willing to take that a variable may be left in the predicting equation that should have been removed. Figure 4 illustrates the nature of this "importance" scale. The F-test measures the extent to which a variable will contribute toward explaining the dependent variable behavior, and tests this contribution against a purely chance correlation by comparing the variance with and without the term. The hypothesis tested is that the variance is equal in both cases and that any difference is due only to chance. Thus, the term will be used only when the difference in variance cannot be explained by chance alone. Thus, in Figure 4, if one selects a probability of committing an insertion error (that is, inserting a term into the predicting equation that really does not belong in the equation) and finds the number of degrees of freedom (roughly the number of weighted observations), the value of F indicated by the surface is such that if the value of F displayed by the "best" term exceeds the value on the surface, then the risk is less than the probability chosen. As might be expected, the value of F on the "threshold" surface goes to zero as the probability goes to 1 (certainty of committing an error). In that case, any nonnegative value of F equals or exceeds the "threshold" and the result would be to insert every term whether correlated

or not. Conversely, if one goes toward zero probability (certainty of not committing an error) the threshold value grows, approaching infinity in the case of zero probability. Thus no value of F can exceed this threshold and so no terms can be inserted. For any reasonable probability, the effect of the number of data can be assessed. As the number of data grows large, the "threshold" value approaches a constant dependent only on the probability. As the number of data approach zero, the risk of error is held constant by requiring larger and larger F values with infinity as the value corresponding to "no data." With such a test, the Stepwise Regression Program can control the generation of a predicting equation so that each term possesses a maximum risk of appearing incorrectly. Of course, many, perhaps most, terms actually appearing in the final equation exceed the threshold by substantial amounts and thus represent greatly reduced risks. The test insures that every term is at least as good as the risk specified.

NOTE: If $V_i > 0$, then X_i is not yet in the regression equation and the $V_i > 0$ may be regarded as the relative contribution by the respective X_i in explaining the as yet unexplained variance in the dependent variable Y.

If $V_i < 0$, then X_i is currently in the regression equation. The $|V_i|$ for all $V_i < 0$ may be regarded as the relative contribution by the respective X_i to the regression prediction of Y.

As each term is added or deleted from the regression equation, the regression matrix a_{ij} is modified to contain the corresponding effect.

** The "importance factor" is found by using the variance contribution for each variable. Initially, the correlation matrix r_{ij} defined earlier is equal to the regression matrix a_{ij}

$$a_{ij} = r_{ij}; \quad i, j = 1, 2, \cdots, n+1 \quad .$$

Then the variance contribution for the i-th variable is:

$$V_i = \frac{a_{iy}a_{yi}}{a_{ii}} \quad i = 1, 2, \cdots, n$$

and where the subscript y is understood to be the dependent variable subscript (n+1).

If the user takes a probability of error for removing variables of 0.05 then the odds are 1 in 20 that a term may be left in the predicting equation incorrectly. Obviously, if the user wants to make this error very rarely, he may set the probability of that error very low, say 0.01 or 0.001. This situation requires one additional remark. When one asks that the chance of committing an error be made small, the chance of committing the converse error must become large. In this case, one increases the risk of removing variables that really belong in the equation by decreasing the risk of leaving variables that do not belong in the equation. If the chance of leaving a variable incorrectly were set by the user at 1 in 10000, it is also possible that insufficient data may have been accumulated to allow the retention of any variables in the equation, and the analysis can do no more than predict the average value of the dependent variable Y' by the appropriate b_0. The remedy is clear: if one wishes to set high standards, the price is additional experimentation to produce additional evidence to support the case.

If and when all the importance factors exceed the minimum value required for retention of the set $X_{i,1}$, the analysis then examines the set $X_{i,2}$. For each element of this set a "potential importance factor" (as defined earlier), is determined and the largest of these isolated. These factors measure the relative contribution which each variable not presently in the predicting equation might make to the equation if it were put in. The largest of these is associated with the "best" variable at this stage. Once again a comparison is made to insure that the risk of inserting a variable incorrectly is in agreement with the significance of the "best" variable before the insertion is allowed to occur. Again, the user specifies the risk he is willing to take of a variable being incorrectly inserted into the

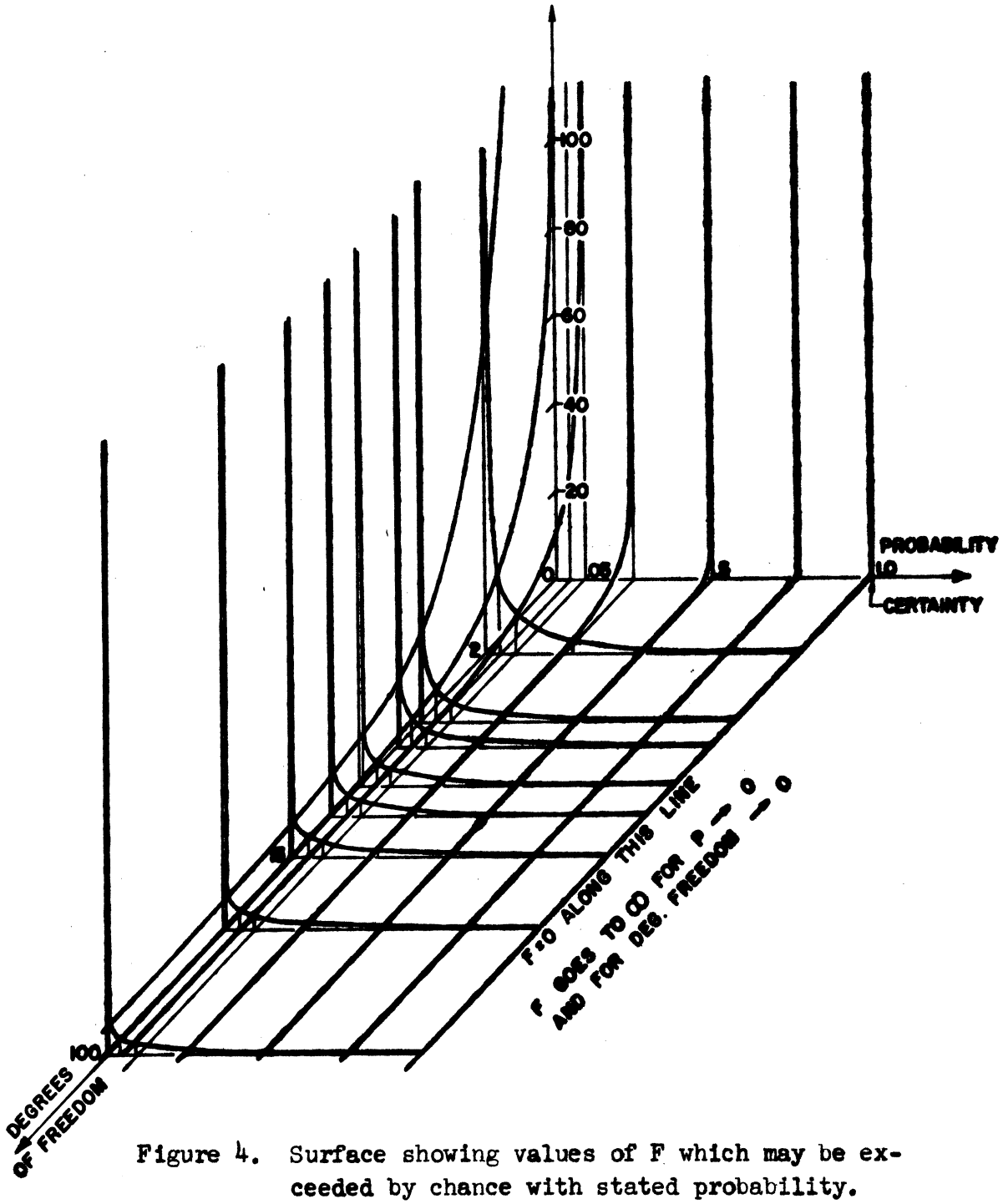


Figure 4. Surface showing values of F which may be exceeded by chance with stated probability.

predicting equation, understanding .05 to mean 1 chance in 20 of the error occurring and recognizing that reducing the chances of incorrectly inserting variables increases the chances of omitting correct variables from lack of evidence.

Suppose that, in the example considered, X_4 has been retained, and that of X_1, X_2, X_3, X_5, the variable X1 best explains the behavior of Y not explained by X_4 . If there is sufficient evidence to support the insertion of X_1, then a new predicting equation is formed by least squares:

$$Y = b_0^{(1)} + b_1^{(1)}X_1 + b_4^{(1)}X_4 \qquad (3)$$

The superscripts on b_0 and b_4 indicate that these coefficients have undergone one modification in the process and are new values. At this point the variables are again sorted and checked for importance and the procedure repeated. The analysis ceases when either all the X variables have been inserted into the predicting equation or none of the X variables that remain as possible candidates for the equation is sufficiently important to allow insertion.

Continuing the example, suppose that on the third step X_5 is introduced, yielding:

$$Y = b_0^{(2)} + b_1^{(1)}X_1 + b_4^{(2)}X_4 + b_5^{(2)}X_5 \; . \qquad (4)$$

Further suppose that X_1 and X_5 behave together in such a way that the results is like having X_4 in the equation twice. In such a case, the importance of X_4 might be considerably reduced. Suppose that this is the case and that the importance of X_4 falls below the limit set by the user. Then X_4 is removed and the equation becomes:

$$Y = b_0^{(3)} + b_1^{(2)}X_1 + b_5^{(1)}X_5 \; . \qquad (5)$$

In step (5) suppose that X_2 is added giving:

$$Y = \overset{(4)}{b_0} + \overset{(3)}{b_1X_1} + \overset{(2)}{b_2X_2} + b_5X_5 \quad . \tag{6}$$

Now suppose that neither X_3 nor X_4 are sufficiently significant to allow their insertion. The final prediction equation produced is (6). The analysis makes available a number of statistics at each step which may be interpreted as a measure of goodness of fit or prediction as well as the b values and the importance level for the term considered at that step.

2. The Statistical Model

Suppose that the physical system giving rise to the preceding example was such that it could be hypothesized that the system could be characterized or described by the mathematical model:

$$Y = B_0 + B_1X_1 + B_2X_2 + B_3X_3 + B_4X_4 + B_5X_5 + E \quad , \tag{7}$$

where the $B_i (i=0, 1, 2, \ldots, 5)$ are unknown and possibly some of them may be zero. E is a random error variable term which accounts for the inability to obtain strictly reproducible data when observing the physical system. Setting aside the consideration of E for the moment, the problem is that of obtaining the best estimates of the B_i. It may be observed immediately that this is the problem just considered, resulting in Equation (6), and that the B_i are estimated by b_i, respectively. The best estimates of B_3 and B_4 are zero.

Turning attention once again to E in (7), it is clear that Equation (6) is not quite complete. The randomness of E makes the prediction of E impossible. What is possible is the determination of the likelihood of E being inside a range of values. In other words, because of E the measurements obtained are not exactly repeatable even if all the X's could be set

at exactly their former values. Therefore, the estimates are possibly, but not necessarily, in error due to the influence of E. A more nearly complete treatment of (7) would

(1) estimate the B_i as before;

(2) estimate the possible errors in the B_i; and

(3) estimate the variability of E.

The Stepwise Regression Program automatically estimates each of the three items desired. The estimate of the B_i has already been discussed. The possible errors in the B_i are indicated by quantities S_{Bi} called the "Standard Error of the Coefficient" for each i. These values are such that if one forms the interval

$$B_i - S_{Bi} \leqq B \leqq B_i + S_{Bi} \tag{8}$$

then the "true" value of B may be expected to be included by this interval in about 68% of all cases (see Figure 2). If one extends the interval to form

$$B_i - 2S_{Bi} \leqq B \leqq B_i + 2S_{Bi} \ , \tag{9}$$

then this interval should include the true value in about 95% of all cases.

The variability of E is measured by a statistic called "The Standard Error of Estimate." This is roughly the standard deviation of the E. Adding additional terms to the predicting equation usually results in reducing the standard error of estimate. The amount of reduction is a measure of the contribution made by that variable toward the explanation of Y. When the analysis is completed, this statistic measures the behavior of Y not explained by the predicting equation and reflects the remaining observational errors and, of course, possible errors in the hypothetical model. The precision of the predicting equation is reflected by the

magnitude of the Standard Error of Estimate (Sy) such that if one uses the predicting equation (6) to estimate Y and then forms a band about the curve predicted by (6) of plus and minus Sy (i.e., the band is 2Sy in width and centered on the curve from (6)), then the "true" value of Y may be expected to be included by this band in about 68% of all cases (see Figure 3). Again, doubling the band width to ± 2Sy raises the expectation to about 95% of all cases. In other words, when enough experimental observations of a physical system are made accurately on good instruments so as to minimize observational errors, and when the hypothetical model correctly describes the physical system, then Sy will be small and the predicting equation may be used to estimate Y with a measure of the precision of this estimate interpreted as indicated.*

The analysis produces two other valuable statistics at each step of the estimation process. The "Coefficient of Determination"** is interpreted as the proportion of the total variation in Y that is explained by the predicting equation. The possible values lie in the range from +1.00 (perfect prediction) to 0.0 (no prediction). Statisticains familiar with the "Multiple Correlation Coefficient," which is the positive square root of the Coefficient of Determination, will find it displayed also.

II. Artificial Intelligence Applied to the Stepwise Regression Method

Section I of this discussion treated the use of the Stepwise Regression Method as it applied to those cases in which the entire set of

---

* It should be mentioned in passing that the interpretation of Sy and S_{Bi} should be as stated here and that it is not correct to say that about 68% of all observed values will lie within the intervals indicated for ± S and so on.

** The Coefficient of Determination (R^2) is found by subtracting the regression matrix element a_{yy} (which measures the dependent variable variance) from unity. That is, $R^2 = 1. - a_{yy}$.

variables and functions of variables can be represented by a single collection of small enough size to allow complete retention within the memory of the machine. In the case of the IBM 704 with 8192 word core storage, the size of the problem is limited to 60 variables, which require, in addition to several linear arrays of 60 elements, a matrix 61 by 61 or 3721 locations. While some expansion might be realized by adroit programming, a little study of the nature of the problem indicates that an expansion in capacity of several orders of magnitude together with a new concept of programming will be required to handle problems of the types commonly encountered in research.

To understand the nature of the problem encountered, consider the following example. Suppose that, as in the example of Section I, an experiment has been made consisting of a set of observations of six variables. Once again we regard one of the variables as a dependent variable and the remainder as predictor or independent variables. Assuming for the moment that only linear behavior is to be expected from any variable (a drastic simplification), it is apparent that the formal relation (1)

$$Y = b_0 + b_1X_1 + \ldots + b_pX_p \qquad (1)$$

is not completely descriptive of even this simplified case. This is because of the possible existence of interactions between variables. Extending the example proposed to include interaction requires the inclusion of sets of terms of the forms 1) X_i, 2) X_iX_j, 3) $X_iX_jX_k$, 4) $X_iX_jX_kX_\ell$ and in this case the single term $X_1\ X_2\ X_3\ X_4\ X_5$ as possible candidates for the predicting equation. The number of such terms is found in the following way.

Let there be K groups of n_i distinct objects $(i = 1, 2, \ldots, k)$ and let there be selected j objects

$$j \leq \sum_{i=1}^{K} n_i$$

at a time to form combinations. The number of such combinations is readily obtained for the case $n_i = 1$ for all i (the present example case). The number is (for $n_i = 1$)

$$N_j = K!/(K - j)!\, j!$$

and the case of K = 5 produces the table.

TABLE OF N_j FOR $n_i = 1$ AND K = 5

j	N_j	$\sum N_j$
1	5	5
2	10	15
3	10	25
4	5	30
5	1	31

The table shows that even the simple example chosen has expanded the required storage capacity from a 6 + 1 square matrix of 49 locations to a 32 + 1 square matrix of 1089 locations. Furthermore, the usual problem does not permit the assumption of $n_i = 1$. The more general case may be determined if

(1) n_i is constant for all i;

(2) selection occurs always between groups and not within groups.

Then

$$N_j = [K!/(K - j)!\, j!]n_i^j \qquad (13)$$

Condition (1) is not unreasonable and condition (2) simply requires that n_i be large enough to include whatever terms might be desired generated within the smaller group. That is, if one considers X^2 and X^3 and wishes also to consider $X^5 = X^2 * X^3$, then condition (2) requires that X^5 be made a member of the n_i (and not generated from X^2 and X^3).

Suppose that in the example 10 function choices are suggested for each of the five variables (the use of 20 or more is not uncommon in problems concerning a single independent variable). Neglecting interactions the problem requires 52 * 52 locations. Considering interactions and using (13), one obtains the table:

TABLE OF N_j FOR $n_i = 10$ AND $K = 5$

j	N_j	N_j
1	50	50
2	1000	1050
3	10000	11050
4	50000	61050
5	100000	161050

Obviously this is outside the range of even projected computers since the matrix now requires $(161052)^2$ locations. The cost of solution by conventional methods is also prohibitive since the solution of a 3-variable problem with 20 function choices per variable (which requires $(9260)^2$ locations) has been estimated to require 2500 hours on the 704.

Conventionally, work has progressed in this field by the expedient of setting the coefficients of all but a very few of these terms identically equal to zero. The formal relation (1) is such a reduction. This method, while enabling some attack to be made on otherwise nearly hopeless problems, suffers greatly for several reasons. First of all, the choice of omitted terms is a process of discarding thousands of terms to retain one. Secondly, the usual practice of relying on apparent fit to select terms before the

regression process begins may result in the omission of exactly the terms needed.*

A procedure is needed to conduct a search through thousands of possible terms engaging only a few dozen at a time to produce the predicting equation. To be as effective as possible, it would be very desirable to use each experience with the problem, whether successful or not, to learn more about the nature of the terms that are generally useful and thereby accelerate the search. Such techniques as "learning" and the "acquiring of experience" are generally associated with nonmechanistic organisms. Since it is proposed that these techniques be simulated by the 704 computer, this is the application of artificial intelligence to the problem.

The program has been written so that the machine is not presented with the condensed subset, as usually happens, but instead is given access to all possible terms and interactions within the bounds of the number of variables considered and the number of function choices per variable allowed. As usual in problems of this type, no straight-forward procedure can be given to proceed to the solution that does not also appear economically prohibitive. It is not a matter of instructing the machine how to solve the problem, but instead of instructing the machine how to "learn" to solve the problem. Specifically, the machine must "learn" how to select terms so

---

* Experience with the regression program on single independent variable problems indicates that the terms added successively to the predicting equation bear little relation after the first step to their partial correlation coefficients with respect to the dependent variable. This is because the added terms are always charged with explaining the as yet unexplained variation in the dependent variable. Consequently, if the first term entered explains the dependent variable behavior quite well, the next term may be of quite different character in order to explain what is left by the first term.

that the set of terms chosen contain those terms needed to produce predicting equations of high precision. Much remains to be done in this new and vital area. The present effort contains only the most rudimentary learning but is written in such a way that more sophisticated learning models can be inserted fairly easily. Experience with the simple learning mechanism has been extremely encouraging.

Turning to the example of 5 variables and 10 functions per variable, the following discussion describes the nature of the learning scheme used by the program. Suppose that no knowledge of the nature of the more likely terms nor of the relative importance of the various term classifiers are known a priori. A "term classifier" is one of the set of (1) interaction order identifier, (2) variable identifier, or (3) function identifier, and is used to classify terms as to the degree of interaction, variables involved, and functions of variables involved in the term. If such knowledge is presumed known before commencing the solution, means are provided to suggest either the initial set of terms to try or the initial distribution of weight among the term classifiers or both or neither. In the present case, neither are assumed to be supplied so the discussion may be understood for any other case where more information is given initially.

Since term classifiers are not assumed to be supplied, the program assumes no previous experience with the problem and accordingly sets the relative likelihood of all terms equal. This is accomplished by considering each of the classifiers as an array the elements of which are the lengths of the components of a vector. Each component is initially set to a unit length.

Next the initial set of terms must be generated by the program. Each of the 161050 possible terms in this example are equally likely at this stage. The program uses a pseudo-random number generator to select (1) an interaction classifier, (2) variables to satisfy the interaction selected, and (3) a function for each variable chosen for the interaction. As each term is selected, a check is made to be sure that it is not a duplicate of an earlier term chosen for the current pass. When the number of terms (less than 60) requested by the user for each pass have been chosen and entered in a term matrix, the program calls upon an editor program to examine the data and the term matrix and thereby generate the set of edited data required by the regression analysis program. The editing process consists of operating on the raw data by referring to the term matrix for the definitions of the terms and to subroutines to carry out the generation of the terms. Each raw observation is converted into the edited data and a magnetic tape recording of the result is made. When all the data have been edited, the program turns to the Stepwise Regression Program to carry out the analysis exactly as before with respect to the set of terms chosen by the program. Upon completion of the Regression Program for this selection of terms, a check of the generated predicting equation is made to see if:

(1) the Coefficient of Determination is as large as the user specified,

(2) the Standard Error of Estimate is as small as the user specified,

(3) the number of passes executed have not exceeded the limit by the user.

If further work is allowed as the result of these checks, the program proceeds to examine the results of the pass just completed and in so doing acquires "experience" conerning the types of terms most suitable for future use.

This "experience" is acquired by the student program as follows. Each term is checked against the list of terms included in the predicting equation. If a term has been successfully used in the relation, the student (1) retains the term to be used again, and (2) increases the probability of trying similar terms by incrementing the lengths of the vector components of the classifier arrays that chose the term. If the term was not successful, the student decrements the lengths of the vector components that selected the term. By modifying the vectors by amount proportional to their current size, no term will ever be reduced to zero probability but may have its probability made arbitrarily small but positive. In this way the arbitrary setting of huge blocks of coefficients to zero is avoided and any term may at any time be used successfully and thereby become a member of the predicting equation until supplanted by a still better term.

After the student program completes the study of the previous run, the "experience" gained is utilized to select a new set of trial terms for the next pass. That is to say, the previously successful terms are retained from the former pass and the term matrix is filled out with terms chosen by using the modified classifier arrays and the random selection process. Since the classifier arrays have been modified, the selection of new terms no longer occurs with equal probability for all interactions, variables, and functions. Thus the search is less random and becomes more

nearly stepwise as success and failure direct the modification of the classifier arrays. So long as it is possible to retain terms used successfully on the previous pass and still select some additional term or terms, the program retains the previously successful terms. In this way, the new pass will always be at least as "successful" as the last pass. If, however, a new pass is called for and there is no room for additional terms, the program has encountered a "traffic jam" since a new pass would not be requested if the old selection had been good enough. In this situation, a fresh start is needed but old "experience" may still provide valuable assistance in the selection of terms. The student program discards the old selection of terms (printout of the discarded set is automatic so that human study can be made of it) and selects a complete new set while retaining the "experience" imbedded in the classifier arrays. In this case the machine is completely able to handle the "traffic jam" without outside help.

Another pitfall which might be encountered by the program concerns the case in which the solution has progressed to a locally maximally successful predicting equation. In this case, any change appears to make the predicting equation less useful and yet the present predicting equation is not good enough. An interesting property of the Stepwise Regression method for choosing the most desirable terms results in the ability of the program to work itself out of such a situation. In fact, several instances have been observed in which the program accepted somewhat poorer overall fits for one or two trials in order to retain particularly good terms and on a succeeding trial found the fitted predicting equation to be several times better than the best previous equation.

In any case the process repeats itself, studying, grading, selecting, editing, fitting, until the conditions on the goodness of fit are met or until the desired number of passes have been used whichever comes first. While one cannot be certain that the very best predicting equation possible has been found after any predetermined number of passes (a characteristic of iterative processes generally), the procedure insures that the best solution to date is preserved and that all trials contribute to the improvement of the selection process.

The learning scheme employed by the student program embodies many of the principles discussed by Friedburg, Dunham, and North in their articles on "Learning Machines" in the I.B.M. Journal. The student program extends these ideas and incorporates the advantages of both random search and stepwise search. Initial passes search rather randomly looking for promising leads. As evidence accumulates, the mode of search becomes increasingly stepwise as the number of "good" terms retained grow. Thus the search narrows itself into promising areas and progress is made toward solution until either a solution is found or the allowable number of passes is exceeded or until either a solution is found or the allowable number of passes is exceeded or until a "traffic jam" forces the random search to begin again. Random searching of the early stages is most promising since a poor start does not inhibit progress. Later stages have experienced some success and therefore the modifications are less drastic to allow the previous leads to be followed as far as they may prove to be profitable.

During the solution of any particular problem, it may happen that, when the data are operated upon by the editor program to produce the edited data, the size of the numbers generated may overflow or underflow the size of the IBM 704 word. In floating point arithmetic this may

occur whenever the editor produces a non-zero number with absolute value outside the range 10^{-18} to 10^{+18} because of a later production of the sums of the squares and cross products of the terms by the Stepwise Regression Program. In these circumstances, the student program cannot experience "learning " for those terms of correct size since they have not yet been tried for the actual curve fitting, but the student program must "learn" about the selection of terms acceptable to the 704. Occasionally, it has been observed that the terms suggested by the curve fitting process and the terms acceptable in size to the 704 may not agree. The present learning mechanism is capable of correcting itself in this case without requiring human intervention.

Some final remarks may be of assistance in understanding the analysis. First of all, a given set of data may result in more than one predicting equation of a specified goodness of fit. This corresponds to the existence of several mathematical models of the system. Classically, this situation leads to the development of experiments capable of distinguishing between the models and the retention of those models which best describe the greatest variety of consistent circumstances accurately. By randomly restarting the problem this possibility may be investigated. If the program produces different predicting equations upon random restarting, more evidence is needed. Failure to produce different equations, however, does not guarantee freedom from such difficulty but decreases the probability of this difficulty. Secondly, if previous experience with a problem is available, prudence usually dictates that the initial pass make full use of it. The program provides ready means for saving previous results and for

restarting with any or all of the previous classifier arrays and term selections intact. This same philosophy may be extended to initial runs in which the user's training and experience or previous encounters with similar problems may serve to generate an initial selection and/or weighting. The penalty for a poor guess is an increased number of passes, but a good guess results in considerable saving.

The Stepwise Regression Program with Simple Learning has been used successfully on many test problems and actual physical component modeling problems. In addition, interest in this technique has been generated in many diverse areas of the physical and social sciences. The ability to know precisely the worth of each and every term in a predicting equation, as well as the worth of the equation as a whole, as it is supported by actual evidence, should enable extensions of knowledge in many fields.

IMPLEMENTING THE STEPWISE REGRESSION PROGRAM

WITH SIMPLE LEARNING

Communication of the Problem to the Program

As it was in the case of the Simulator Program, the immediate concern of the user of the Stepwise Regression Program is to communicate the problem to be solved to the program. Since the problem is essentially computational, the link is established through the use of a set of control cards.

The program is designed to be very flexible in the analysis of the problem. Thus, the user must select the specific operations to be performed and the constraints to be imposed. The user must supply, in addition to the observed data, the following control cards:

1. Title card
2. Problem control parameter card
3. Solution control parameter card
4. Output control card
5. Simple Learning control card
6. Core and Tape Layout card
7. Initial Random Number card

Depending on the contents of the Problem control parameter card, one, several or all of the following groups of cards may be required:

8. Ordered Term Insertion cards
9. Data deck
   - 9A. Format specification card
   - 9B. Observed data deck
   - 9C. End of data card
10. Accumulated Learning deck
11. Initial pass terms deck
12. Output Function Name card.

In the foregoing list, items 8, 10, and 11 may be present or absent from the input deck depending on the contents of the Problem control parameter card. The remainder must be present in every input deck. The order of the deck follows exactly the order of the list.

Title Card

The title card allows the user to present any title that may be desired to be printed at the beginning of a new problem. Only one card may be used and the title may appear anywhere within columns 1 thru 72. Ordinarily, the user will place the number one in column one so that the printing for the problem will begin on a new page. Whatever appears in column one is the printer carriage control character. If a blank card is used, the printer will simply single space the paper.

Problem Control Parameter Card

The function of the parameters on this card is to allow the specific problem being treated to be handled in accordance with the user's wishes. The format of the card is (I5, 3F10.5, 7I5, I2). The parameters, in order, are:

1) Problem Number, an integer modulo 32768. (cols. 1 thru 5)

2) Tolerance for division and round off error. A floating point bound such that if the magnitude of any divisor is less than this value no division will occur. This value is also used to limit round off error in the matrix manipulation. Typical values are 0.0001 to 0.0005. (cols. 6 thru 15)

3) Probability of insertion error. A floating point number in the open interval from 0. to 1.. The value is the probability allowed by the user that the least significant term inserted into the predicting equation is erroneous. A value of 0.05 represents a risk of 1 chance in 20, a value 0.01 represents a risk of 1 in 100 and so on. (cols. 16 thru 25)

4) Probability of a deletion error. A floating point number in the open interval from 0. to 1.. The value is the probability allowed by the user that a term removed from the predicting equation for lack of support should have been allowed to remain in the equation.

The probability of a deletion error must not exceed the probability of an insertion error. If it does, the program may reject every term offered. (cols. 26 thru 35)

5) Number of independent (predictor) variables. An integer less than or equal to 59. (cols. 36 thru 40). This value plus one is the total number of variables in the problem.

6) Number of functions to be considered for each independent variable. An integer less than or equal to 60. (cols. 41 thru 45). The choice of functions to be used by the program is determined by the subroutine PFNCT in the Editor Program Core ((No. 4) and the output section of the Program Generator Core (No. 8) should obviously be made to agree with these functions). The user is free to replace PFNCT if the need arises in any particular problem. The "standard" version provides automatically integer powers, integer roots and their reciprocals. The extent of the set so generated depends on the number of functions. The order of these functions is: (in MAD notation)

Function No. 1 ..... X(I).P.1
2 ..... X(I).P.-1
3 ..... X(I).P.2
4 ..... X(I).P.-2
5 ..... X(I).P.1/2
6 ..... X(I).P.-1/2

and so on, repeating the pattern of functions 3, 4, 5, and 6 above for each increasing integer.

Typical values for this parameter are; 10 (yielding functions thru X(I).P.-1/3), 22 (yielding functions thru X(I).P.-1/6), 38 (yielding functions thru X(I).P.-1/10).

7) Number of terms to be tried at each solution pass. An integer less than or equal to 59. (cols. 46 thru 50)

8) Number of terms whose order of insertion is specified in the input data. An integer less than or equal to the item (7). Usually this variable is set to zero, but may be set positive and thus force complete control over the order of insertion of terms by the user. (cols. 51 thru 55)

9) Number of terms initially defined by the user. An integer less than or equal to item (7). Defined terms under this control will be used subject to the statistical analysis of the program unless overridden by item (8). If less than the total terms in (7) are defined by the user, the program will attempt to generate enough new terms to satisfy (7). (cols. 56 thru 60)

10) Parameter controlling the type of regression analysis executed by the program. An integer (cols. 61 thru 65) operating as follows:

10A) If greater than zero, the data is treated with respect to the coordinate axes and the constant term is always suppressed to zero.

10B) If equal to zero, the data is treated with respect to a set of axes translated to the means of the variables. The constant term is not suppressed.

10C) If less than zero, the data is treated as in (A) but the constant term is not suppressed. The constant term is treated

just like every other term, except that the constant is always inserted as the first term the relation and held until the next term is tried. After this point, all constraints are removed.

(The type C is most useful in dealing with physical data. Type A is most useful when other information dictates a zero constant. Type B is most useful when dealing with data that tends to group itself about the means, (Biological and sociological problems).)

11) Parameter indicating whether the data is all of unit weight (parameter value not equal to zero), or weighted individually (parameter value equal to zero). (cols 66 thru 70). If the parameter is zero, each set of data must carry a value of its weight.

12) Parameter indicating whether the program has previous "experience" with the problem. If not equal to zero (or blank) the program assumes that the accumulated learning deck is present. Integer variable in columns 71-72.

Solution Control Card.

The solution control card communicates to the program the conditions under which the program is to cease calculation. The format is (2F10.5, I5). The parameters, in order, are:

1) Estimated Coefficient of Determination. Floating point variable in the range of 0. to 1.0. This parameter is the user's estimate of the expected goodness of fit between the predicting equation surface and the data. Perfect agreement is represented by 1.0, no agreement is represented by 0.0. Typical values for physical problems run from .95 to .999. (cols. 1 thru 10)

2) Standard Error of Independent Variable. Floating point variable whose value is the user's estimate of the standard error of the independent variable represented in this data. The value reflects the probable errors present in the data in units of the same kind as the data. (cols. 11 thru 20)

3) Number of Passes allowed for this problem. Integer variable whose value is the allowed number of complete passes to be made on the problem. Typical values run from 1 to 10. (cols. 21 thru 25)

The program will run until both conditions (1) and (2) are met, or until the passes are used up, which ever comes first. Condition (1) is met when the program has found an equation whose Coefficient of Determination exceeds or equals the specified value. Condition (2) is met when the program has found an equation whose Standard Error of the Independent Variable is less than or equal to the specified value. Both conditions must be met in order to terminate the program before the allowed number of passes are used.

Output Control Card

The program must perform a variety of subsidiary calculations during the equation generation process. The output control card allows the user to suppress those extra calculations and printing for which he has no need. If a blank card (no suppression) is used, all calculations will be printed. Since, for most problems, this represents a very sizeable volume of printing the user is cautioned to select only those items of real interest. Punching a numeric 1 in the column corresponding to the item number given below will suppress printing of the calculation. Either a numeric 0 or a blank allows the printing to occur.

The suppressable output parameters, in order by column number, are:

Column No. 1) Raw Sums of Squares andCross Products
2) Average (Mean) values
3) Residual Sums of Squares and Cross Products
4) Standard Deviations
5) Partial Correlation Coefficients
6) Intermediate Steps in Regression process
7) Predictions using the intermediate step equations
8) Predictions using the final equation
9) Values of terms for each set of observations

If all of the above are suppressed, the output will consist of:

1) Listing for verification of all raw data.

2) Definitions of terms used for each pass.

3) Final equation found for each pass, with pertinent statistics. That is; the F level of the last term treated, the standard error of the independent variable, the coefficient of determination, the multiple correlation coefficient, the constant term, if any, and the coefficients and their standard errors for all terms finally retained in the equation.

4) The diagonal elements of the regression matrix.

5) The equation produced by the last pass in M. A. D. subroutine form both printed and punched on cards ready for processing.

6) The final status of the "learning" mechanism punched on cards for use in restarting future problems.

7) The terms to be used for the next presentation of the problem to the program (on cards).

8) A pseudo-random number card to allow the random number generator to continue the sequence.

For most applications, the automatically produced output is sufficient. The next most generally interesting results are the items (6) and (8) in the first group. If (4), (7) and (8) are suppressed, some calculation is also suppressed thus speeding execution time.

The user is cautioned again that the request for all of this printing will, in general, produce a very sizeable output.

Simple Learning Control Card

The user may, at this stage in the development of artifical intelligence programs, control the characteristics of the learning mechanism. Use of the external function structure for the program allows fairly easy modification of the various parts of the program. With the "standard" learning mechanism as it is now used, data is accumulated concerning three kinds of selections:

1) Order of Interaction
2) Variables Entering Interaction
3) Functions of the Variables.

A term is generated by selecting an interaction order, next the variables to be concerned in the interaction and finally the functions of the variables selected. The term is the cross product of the functions of the variables selected. The program must "learn" which interactions are most useful in explaining the data, which variables are most useful, and which functions of these variables are most useful. The program "learns" by trying to use terms selected by the program to explain the data. If the mechanism has selected a term which is supported by the data and retained by the regression analysis, the meachanism that selected that term is

modified so as to be more likely to select terms of a similar character. On the other hand, if a term is not supported by the data and is, thus, of no apparent utility in the equation it is cast out and the mechanism is adjusted to be less likely to select terms of similar character. Since the probability of selection of any component of any allowed term should be bounded positive, the program uses a "half-life" constant to modify the probabilities. In this way, the relative probability of any term may be made arbitrarily small but remains positive. The usual card is of format (I5, 4E16.8). If the mechanism is modified to require more constants, succeeding cards (up to 2) are of format (E21.8,3E16.8).

The parameters are:

1) Number of constants used by "learning" mechanism. Integer in cols. 1 thru 5. (The standard mechanism uses 3 constants. The numeric 3 is punched in col. 5.)

2) The Constants used by the "learning" mechanism.

The standard mechanism uses:

2A) The "half life" of the Interaction selector. Typical value is 3.0E00. This means that three consecutive successes will double the present probability (or conversely, three consecutive failures in halving of the present probability).

2B) The "half life" of the variable selector. Typical value is 3.0E00.

2C) The "half life" of the function selector. Typical value is 1.5E00. Since any function may be used relatively infrequently it is somewhat desirable to take more powerful action on each encounter.

A great deal of work remains to be done in exploring "learning" mechanisms. It should be observed here that as the constants are made larger, the mechanism "learns" more slowly. In fact, for very large values of the constants the mechanism is essentially deactivated. Very small values of the constants, on the other hand, may cause wildly erratic behavior of the mechanism since each encounter so strongly distorts the relative probabilities.

The values given have received much use and appear to give quite stable operation although not necessarily optimum convergence.

The user will ordinarily duplicate this card and the next one from run to run.

Core and Tape Layout Card

In order to allow easy extension of this program in the future, the multiple core program arrangement can be changed and tape layout changed without disrupting the entire program by using this card. Present core arrangement and tape layout is the following: (Users wishing to modify the layout are advised to study the program flow charts carefully.)

1) Starting Program is in two consecutive core loads. The first core of the Starting program is now core 1. Punch 1 in column 5.

2) The Student Program is one core load and is now core 3. Punch 3 in column 10.

3) The Editor Program is one core load and is now core 4. Punch 4 in column 15.

4) The Regression Program and Program Generator Program are three core loads, the first two of which are the Regression program. These must be consecutive core loads. Punch 5 in column 20.

5) The Processed Data erasable tape is now tape 3. Punch 3 in column 22.

5A) The Selector mechanism is now stored as the first record on tape 3. Punch 1 in column 24.

5B) The Raw Data after processing into terms values is now stored beginning as the second record in tape 3. Punch 2 in column 26.

5C) The Terms selected for each pass are now stored as the second record on tape 3. Punch 2 in column 28.

6) The Raw Data erasable tape is now tape 4. Punch 4 in column 34.

6A) The raw data is now stored on tape 4 in binary beginning as the first record on tape. Punch 1 in column 36.

Space is allocated in storage for five tape record assignments for each tape. At this time, the only assignments are the above.

Initial Random Number Card

The random number subroutine used by the Simple Learning Mechanism may be reset arbitrarily at the beginning of each problem. Since the program will produce one of these cards at the end of the problem the sequence may be continued easily. The format is 3I10. The first number is any odd integer modulo 32768_{10}. The second number is any integer modulo 32768_{10}. The third number is any integer modulo 32_{10}. The subroutine combines these integers to form one 35 binary digit odd integer. This integer serves as the first member of the pseudo-random number sequence generated by the library subroutine RAM2.

Ordered Term Insertion Card(s)

If the parameter (8) of the Problem Control Parameter Card is non-zero, the user must supply a set of cards to define the order in which terms are to be inserted. Thus allows an arbitrary equation may be generated without regard to the statistical analysis after which the statistical analysis may be used to discard those terms that are not sufficiently important to meet the deletion error eriterion. If a theoretical relation is available for which a study is being made to determine how the relation may be improved, this feature may be useful. Otherwise, one must assume the risk that some of the theoretical terms will be displaced in the search. The user must be aware that the use of these term order cards is a severe constraint on the analysis and treat the results accordingly.

The format is 14I5. Each five columns contains a integer whose value is the number of a term to be inserted. The first number is the first term to be inserted, the second number is the second term and so on. For example, if parameter (8) on the Problem Control Card were three and column five on the Ordered Term Insertion Card were six, column ten were three and column fifteen were one, the effect would be to insert term six, then term three and then term one after which the Stepwise Regression Program would examine the equation to be sure that these terms meet the deletion error criterion. If any of these terms fail the test they will be discarded. When all of the terms in the equation meet the deletion error test, the remaining terms not yet in the equation will be tested for insertion. From this point on, the standard analysis is followed.

Data Deck Preparation

1. Format Specification Card

Since the data may come from various sources, the data deck allows the data format to be specified at execution time. This is done by using a standard FORTRAN format statement beginning with the word FORMAT (beginning in column seven and ending in column thirteen, followed by any allowable format specification that can be placed on one card and terminating with a ")" right parenthesis in or before column 72. For example, the following format statements would be acceptable:

```
Column
   7
   FORMAT (5F10.2,E16.8)
         or
   FORMAT (4E16.7, F10.1/4E15.8)
```

This card must immediately preceed the data deck, and is known as the Format Specification Card.

2. Data Cards

Following this card are the data cards. Arbitrary formats are allowed as described above. The data must be listed for each observation in the following order however. (All values are floating point numbers)

1) Observation Number. There must be a positive observation number for every observation. Run number one must appear once and only once.

2) Independent Variables. These values are listed in order following the observation number. The values must correspond to one observation group.

3) Dependent Variable. The variable whose value is to be predicted must follow the predictor or independent variables.

4) Weight of this Observation group. The weight of the group may be specified or can be assumed to be unity depending on the parameter (11) of Problem Control Card.

3. End of Data Card

The actual data is then followed by a complete blank data set which acts as a termination for the data. If any Observation Number is blank or less than or equal to zero, the data input is terminated at that point. Therefore, the user must take care in preparing the data deck so that the entire set of data will be read into machine storage.

The program automatically counts the weighted data sets to establish the degrees of freedom for the analysis. In this way, new data can be added to the data deck and/or old data can be deleted very easily.

The Accumulated Learning Deck

Whenever a multiple independent variable problem occurs in which a large number of functions are allowed for each independent variable and interactions of all orders are admitted, the result is the generation of a very large set of possible terms that may appear in a predicting equation. Since the most desirable equation consists of a "minimal" set of these terms comprising those terms most significant in explaining the dependent variable behavior, it becomes apparent that the analysis must usually perform a selection process while dealing with a segment of the entire set of terms at each encounter.

If it is possible to verify the validity of terms independently from their method of initial selection then it becomes feasible to allow the machine to select the terms using some heuristic method. The terms so selected will not always be the correct ones or even the "best" ones although the method of selection should certainly tend to operate in this way. The important point to observe is that the validity of the term is tested by the regression analysis independently of the selection and the regression analysis is, therefore, not affected in any way by the heuristic method of selection. Because of this, the heuristic term selection method is free to select terms using any convenient scale for choosing the terms. If the terms so selected are shown to have validity by the regression analysis then the heuristic method that selected the valid term is modified so as to be more likely to select similar terms. A converse action occurs whenever the term is shown to be invalid.

At the completion of each solution pass the current status of the selector mechanism is represented by a set of values which give:

1) The relative probability of each Interaction Order
2) The relative probability of each Independent Variable
3) The relative probability of each function of each variable.

Whenever no accumulated learning deck is available, parameter (12) of the problem control parameter card is set equal to zero. The result is that the program will then assign equal unit relative probability to all interactions, variables and functions of variables.

If previous encounters with similar problems have occurred, however, the program has already had "experience" with a similar problem and can be allowed to take advantage of these encounters by providing the accumulated learning deck that was automatically produced at the conclusion of the former problem together with the new problem data. If the user desires to transmit this information to the program, parameter (12) of the problem control parameter card is set equal to one and the accumulated learning deck is placed after the data deck.

The user can also suggest his own experience to the program by preparing an accumulated learning deck. The format is 5E14.7. The relative probabilities are inserted in the following order:

1) Relative Probabilities for Interactions from first order to the maximum order for the problem.

2) The sum of the preceeding probabilities (1).

3) Relative Probabilities for Independent Variables, from the first to the last in the same order as they appear in the data deck.

4) The sum of the preceeding probabilities (3).

5) Relative probabilities for each function of the first variable followed by the sum of these probabilities.

6) Relative probabilities as in (5) for the second, third, etc. variables.

The preceeding items are punched successively in the available fields as specified by the format. No blank fields are permitted between groups since every field is interpreted consecutively.

The accumulated learning produced by the machine program has each relative probability normalized so that the mean relative probability is unity. In this way, a problem may be easily expanded to more variables, functions, etc. and still retain previous "experience" by making all new entries of unit value.

Because of the independent regression analysis of the terms chosen from the accumulated learning it must be emphasized that this mechanism cannot force the adoption of incorrect terms. Rather, such an incorrect set of "experience" would be modified progressively by the program. If the "experience" supplied is valid for the current problem the result is to speed the generation of the desired equation but invalid "experience" can only temporarily delay this generation.

In general, if good experience is available from previous similar problems or from the user's background the user is strongly advised to make use of it.

Initial Pass Terms Deck

The user may suggest any initial terms that may be desired for the first pass. If the suggested terms stem from theoretical consideration and the theory is in agreement with the data, such a suggestion will speed the generation of the predicting equation by insuring an early treatment of likely terms. Any number of terms may be given initially up to the total number of terms allowed for each pass. The number of terms to be so defined by the user is given by the parameter (9) on the problem control parameter card. If fewer than the total number of terms to be tried at

each pass (parameter (7)) are given initially by the user, the machine program will generate the remainder of the set by using the accumulated learning and the selector mechanism.

At the end of each problem and immediately following the production of the accumulated learning deck, the program produces a set of terms to begin the next encounter with the problem. If the problem is continued later these terms may be supplied by simply including these cards following the accumulated learning deck.

If the user wishes to suggest terms the procedure is the following: (The card format is 14F5.0)

1) Produce a term card (or cards if sufficient variables are present) for each term desired.

2) Treat each consecutive field in the above format specification as in one to one correspondence with the independent variables in the problem.

2A) Insert the appropriate function number in the field corresponding to the desired variable.

2B) Leave blank (or zero) every variable field not associated with the term.

3) Insert the interaction order in the field immediately following the last variable field.

For example, suppose the user is dealing with two independent variables and the "standard" PFNCT subroutine and desires to form the term: X(1).P.3*X(2).P.-1/2.

The term is specified by punching seven in column five, six in column ten and two in column fifteen. See "standard" functions in PFNCT as defined by number. The seven selects the function integer power three and the appearance in column five assigns this function to variable one in this term. The two in column fifteen declares the term to be a second order interaction and thus produces the desired multiplication of the previous functions of the variables. In this way any desired term allowed by the subroutine PFNCT (and hence allowed by the user) can be specified as an initial term. This is true even when the function numbers in the initial term specification exceed the value of parameter (6) on the problem control card. Of course these terms are, in this instance, excluded from automatic generation but will, nevertheless, be used correctly and preserved from pass to pass correctly so long as they are in agreement with the date. Once discarded from the set of terms only those terms allowed by parameter (6) can be regenerated.

Output Function Name Card

At the conclusion of each problem the program produces, in printed form and on punched cards, the external function subroutine for the last regression equation found by the analysis. This function is ready for immediate translation by the MAD translator and may be used in any program as the user may desire. The Output Function Name Card assigns a name to this subroutine. If a blank card is supplied at this place in the input deck the function name will be left blank. Otherwise the desired name is entered somewhere in the columns 1 thru 72 on the Output Function Name Card.

The program will use the last six non-blank alphanumeric characters as the function name. If a total of fewer than six non-blank alphanumeric characters appear in columns 1 thru 72, these characters are taken to be the function name. The rules for allowable function names are those of the MAD translator.

If the desired function name consists of exactly six alphanumeric characters, the user is then free to insert any desired comment before the desired name. The comment will, in this case only, be ignored.

Examples of allowable function names:

ETA17; PRATIO; TORQUE; EFF23

The Structure of the Program

The Stepwise Regression Program, like the Simulator, is structured in several sections. Each section performs certain tasks which cause, as a result of the performance, a selection of a new section of the program to be performed. There are seven basic sections:

1. Input Section
2. Initial Term Section
3. Student Section
4. Editor Section
5. Regression Statistics Section
6. Stepwise Regression Analysis Section
7. Program Generation Section

The input section brings into the program all of the data associated with a given problem. The control parameters and data are all entered at one time so that if any data set encounters trouble the input tape will be properly positioned for the next problem. As discussed in the section on communicating the problem to the program, the initial terms may be supplied by the user or chosen by the machine as desired. If supplied, the input section will then bypass the Initial Term Section and the Student Section.

If the initial terms were not supplied, the program calls the Initial Term Section to choose the terms for the first pass. The selection is based on the Accumulated Learning supplied by the user. If no Accumulated Learning is given, the program assumes initially that all possible terms are equally probable and proceeds with this assumption. The terms are selected

by choosing:

1. Interaction Order
2. Variables for the interaction
3. Functions of the Variables.

After selecting the initial terms, the control passes to the Editor Section. In this section, the terms defined earlier are evaluated for every data point. If an eminent overflow or underflow of the machine register capacity is found, the faulty term is rejected and the problem is passed to the Student Section so that the Accumulated Learning may be adjusted to tend to avoid such a recurrence. If no such machine limitation is found, the control passes to the Regression Statistics Section.

The Regression Statistics Sections computes raw sums of squares and cross-products, means, standard deviations, sums of squares and cross-products adjusted to means, and simple (product-moment) correlation coefficients. Since these statistics are generated in a conventional manner reference is made here to suitable texts for elaboration[11,12,18]. The important item of interest is that all of these statistics are available for the study of the data. Upon completion of these tasks, the results in the form of the regression matrix are passed along with the control to the Stepwise Regression Analysis Section.

In the Stepwise Regression Analysis Section the techniques of Efroymson and Dallemand are employed but modified slightly to allow more flexible manipulation of the analysis by the user. Four basic analyses may be performed:

1. Analysis for fit of data about means.
2. Analysis for fit of data with respect to the coordinate planes with constant term suppressed.
3. Analysis for fit of data with respect to the coordinate plants using constant term
4. Controlled term insertion order analysis.

Of these analyses, the fourth is most risky since the user overrides the statistical analysis. If the user finally removes the imposed constraints on the term insertion process however the Stepwise Regression Analysis Program will automatically discard any terms that are not sufficiently correlated with the data. In this way, occasionally, a special problem may be studied to advantage. The other three analyses are basically similar except as they are related to the coordinate systems and the constant term. The analysis proceeds as follows:

1) Select the term with greatest contribution to the explanation of the as yet unexplained variance of the data.

2) Compare the variance contribution for this term with a random variable to determine whether the contribution could be due to chance.

3) Insert the term if and only if the variance contribution exceeds that of a random variable by whatever amount the user wishes to specify. Commonly the term is inserted if the risk is less than one in twenty to one in one hundred.

4) Review all terms in the equation and reject any that may have been reduced in importance below the user's standard (by combinations of other terms, etc.).

5) Continue this process until none of the terms not in the equation can meet the standard set in (3).

Upon completion of the Stepwise Regression Analysis, the coefficients for each term found to be valid by the analysis are produced together with the corresponding standard errors for the coefficients. Other statistics

produced at this point are the multiple correlation coefficient, the coefficient of determination, the standard error of the dependent variable with respect to the regression equation and the regression constant, if any. The diagonal elements of the inverse regression matrix is also printed for study. If desired, the user may request the calculation and printing of a point by point comparison of the data and the predicting equation results. This calculation displays, in addition to the actual value produced by the regression equation, the predicted values plus and minus one standard deviation. This band of values may be expected to include the true value of the dependent variable 68 percent of the time. Finally, the deviations and the percentage deviations of the predicted points and data points are produced. During the process the largest absolute deviation and the largest absolute percentage deviation are sorted out and printed.

Until the regression equation produced by the Stepwise Regression analysis satisfies the criteria set by the user, the Analysis next returns to the Student Section to reevaluate the Accumulated Learning stored in the selecting mechanism. The criteria set by the user are:

1. Standard error of the dependent variable
2. Coefficient of determination
3. Maximum number of solution passes allowed.

The analysis continues until the generated equation properties equal or better the first two criteria, or until the allowed number of passes are consumed. This action takes place by recognizing the separation of terms by the regression analysis into those terms sufficiently correlated with the data to be included in the regression equation and those terms not this well substantiated.

All of the terms inserted in the regression equation are retained for the next program unless this would not allow the selection of any new terms. In addition the values of all portions of the selecting mechansim involved in choosing the successful terms are increased so as to make the selection of similar terms more likely. Finally, the values of all portions of the selecting mechanism involved in selecting the unsuccessful terms are reduced to make the selection of similar terms less likely.

The modification of the selecting mechanism occurs on an exponential decay basis. This technique allows the user to specify the number of failures to reduce a particular element of the term selection mechanism by one half. Conversely, this value is the number of successes to double the relative likelihood of the element of the selecting mechanism. The elements of the selecting mechanism are:

1. The interaction selector
2. The variable selector
3. The variable function selector.

After the selecting mechanism is modified by this process the terms for a new attempt by the regression analysis are selected using the modified mechanism. In this way extremely large sets of possible terms may be searched in a very effective manner. An interesting property of technique is the ability of the method to work out of "local optima" in the production of the regression equation. That is, the location of very highly correlated terms may result in an equation which may fit less well than an earlier more complicated relation. This discovery may often be used on later trials to modify the selecting mechanism and thus find an equation better than any previous relation.

When the criteria set by the user are satisfied by either producing an equation that meets the specified statistical criteria or by using up the allowed number of trials, the control is passed to the Program Generation Section. This section produces the final equation as a subroutine both in print and on punched cards ready to be included in any program as may be desired. Versions are available to produce the subroutine in either the M.A.D. language or in the FORTRAN language.

Upon completion of the output of the subroutine form of the regression equation, the program takes one additional step. The control is passed to the Student section and the Accumulated Learning is once again modified and the set of terms chosen for a re-entry of the problem at any later time. This information is preserved on punched cards in the exact format expected by the Input section. It is important to recognize that this information is pertinent not only to the problem at hand but also may be used to expedite the solution of any similar problem. This carrying forward of artifical "experience" is an important and unusual feature of the Stepwise Regression Program with Simple Learning. By this means, the program is enabled to accelerate the generation of predicting equations by recognizing the information latent in previous encounters with similar problems. It is also important to observe that if such artificial "experience" is incorrect for the attempt being made no effect upon the generated equation will be observed except for a use of one or more passes to correct the Accumulated Learning and proceed to the generation of the predicting equation. This action occurs because the data itself is the only finally determining source of information upon which the predicting equation can be based. The Accumulated Learning, if correct, can accelerate the determination, but if incorrect cannot prevent the determination.

CONCLUSIONS

The programs which have been discussed in this paper constitute two new and advanced tools for the study and analysis of the behavior of physical systems. The Simulator Program provides a tool for undertaking the simulation of complicated systems. The flexibility inherent in this technique of analysis is made possible by providing for extension and modification of the library within the structure of the Simulator. It is important to understand that the Simulator Program provides a means for bringing to analysis of systems the very best methods and most applicable techniques. In this way, the Simulator Program proceeds from the information supplied by the user (the System Definition and the Constraints on Input Parameters to be supplied to and the Results desired from the generated simulation program) to produce a procedure, or algorithm, to simulate the operation of the defined system when translated and executed on the digital computing machine.

The Stepwise Regression Program with Simple Learning provides the technique which can supply the programs produced by the Simulator with subroutines to implement the methods extracted from the Library of Element Descriptions. This technique is more generally applicable and has already been utilized in the process of its verification to supply useful predicting equations for data taken from many diverse sources. These sources have included calibration data for thermocouples, fatigue life data for plastic gears, electron tube characteristics, electric transmission line loss characteristics, thermodynamic properties of steam, tool life characteristics, psychological test data and data on the effects of various drugs on human

subjects. In addition to these diverse areas listed to illustrate the versatility of the technique, the technique has been applied successfully to the determination of the characteristics of the steam power plants simulated. These characteristics include the expansion line characteristics of the turbine, the extraction pressure characteristics, the exhaust loss characteristics, the generator loss characteristics and special flow leakage characteristics.

In all of these areas, the work thus far has been most encouraging. There is, however, a great deal of work remaining in perfecting, extending and increasing the generality of the techniques. The methods developed thus far hold considerable promise in other areas. Perhaps, some of the most significant extensions will come from the study of the results produced by these techniques by allowing a powerful analysis of the data sets.

SYSTEM SIMULATOR

FLOW DIAGRAMS AND CORE LAYOUTS

(Comments to assist the interpretation of the flow diagrams in this section may be found beginning on page 227.)

SYSTEM SIMULATOR

FLOW DIAGRAMS & CORE LAYOUTS

SAP	SUBROUTINES	NOT	FLOW	DIAGRAMMED
ONLINE				
INPUT				
OUTPUT				
OCTDEC				
INSRTC				
EXTRC				
INBIT				
IFBIT				
NOT				
OR				
SKPFIL				
TAPSEL				
OCT				
IFABIT				
ANDBIT				
OUTPT 1				
PNCH				
PNCH 12				
RNDM 1 B				
AM1BLD				
AM1CNT				
AM1EIM				
AM1SET				
AM1SUB				
SIGN				
AND				

CORE LAYOUT

(MAIN)	EXTERNAL FUNCTION	INTERNAL FUNCTION	SAP
CORE # 1			
INPUT	NUCOP	CONCK	INPUT
	TAPEIN		OUTPUT
			OCTDEC
			ONLINE
			SAVTPH
			INSRTC
			EXTRC
CORE # 2			
ELEMENT DESCRIPTION	TAPMV		INSRTC
	TAPEIN		EXTRC
	ELTPPR		INBIT
			IFBIT
			TAPSEL
			OCTDEC
			INPUT
			OUTPUT
			ONLINE
			OR
			NOT
			SKPFIL
CORE # 3	TSTDMP		IFABIT
MATRICES SET UP	SYNELM		TAPSEL
	IPRELM		OUTPUT
	QCORE		INSRTC
	TAPEIN		EXTRC
	SSORDP		OCT
	NUCOR		OCTDEC
			TAPSEL
			OR
			NOT
			ONLINE
			SKPFIL
			INBIT

MAIN	EXTERNAL FUNCTION	INTERNAL FUNCTION	SAP
CORE # 4			
DESIRED		EXTCHK	SKPFIL
RESULT		ELCHK	IFABIT
REDUCTION		INSERT	ANDBIT
		REMOVE	NOT
		LISTSC	AND
		RDRUM 1	AM1SET
		RDRUM 2	IFBIT
		CHECK	AM1CNT
		WRDRUM	RNDM1B
		WRDRUM 1	AM1SUB
			AM1RLM
			OR
CORE # 5			
PROLOGUE		PCODE	SKPFIL
		PRLOG	EXTRC
		PSCAN	RNDM1B
		SSCAN	INSRTC
		OUTPT 1	OUTPUT
		STORER	PNCH
		PSUB 1	PNCH 12
		PSUB 2	
		PNAME	
		SKIP	
		PLIST	
		FSUB	
		FSUB 1	
		FSUB 2	
		LABELS	
		INSRT	
		SSCAN 1	

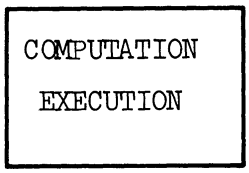
CALCS IS A DUMMY INTERNAL FUNCTION

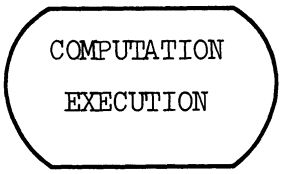
MAIN	EXTERNAL FUNCTION	INTERNAL FUNCTION	SAP
CORE # 6			
PROGRAM GENERATOR		PSCAN	SKPFIL
		SSCAN	NOT
		SSCAN 1	TAPSEL
		PSUB 1	EXTRC
		PSUB 2	AND
		OUTPT 1	INSRTC
		SKIP	SIGN
		QFQ	IFBIT
		LABELS	OR
		FSUB 2	OCTDAC
		FSUB 1	OUTPUT
		INSRT	PNCH12
		QFFQ	
		QPQ	
CORE # 7			
EPILOGUE		EPILOG	SKPFIL
		PSCAN	EXTRC
		SSCAN	OUTPUT
		SSCAN 1	OUTPT 1
		SKIP	PNCH12
		FSUB	INSRTC
		FSUB 1	OCTDEC
		FSUB 2	OR
		LABELS	
		INSRT	
		QPQ	

QPQ AND QFQ ARE DUMMY INTERNAL FUNCTIONS

CORE # 8		
	SELPGM	
DIAGNOSTIC		

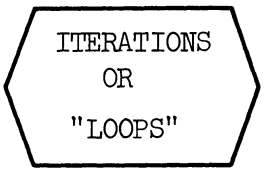
INTERPRETATION OF BOXES





EXECUTION REFERS TO A SUB-ROUTINE OR FUNCTION

WHENEVER STATEMENTS

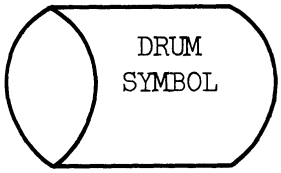


FORM: THROUGH (STATEMENT NAME), FOR (VARIABLE) = (INITIAL VALUE), BY STEPS OF (NUMBER), UNTIL (CONDITION IS)).

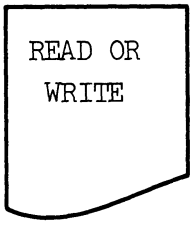
FOR REMOTE CONNECTIONS

FOR MARKING ENDS, OR SCOPES, OF ITERATIONS.

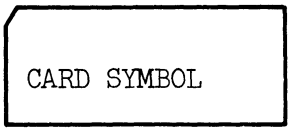
FOR OPERATIONS INVOLVING TAPES.



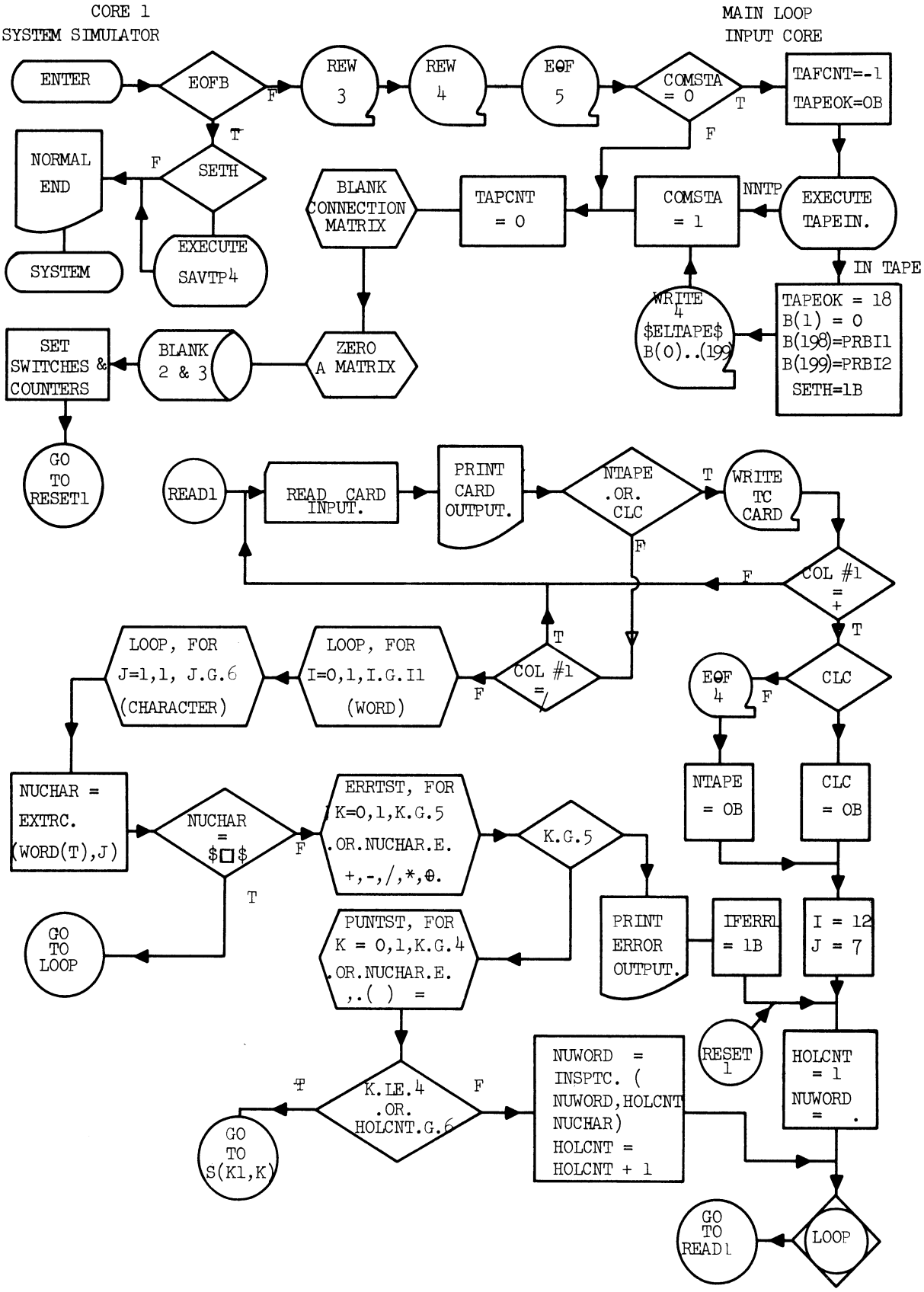
FOR OPERATIONS INVOLVING DRUMS.



USUALLY USED TO DENOTE PRINTED OUTPUT

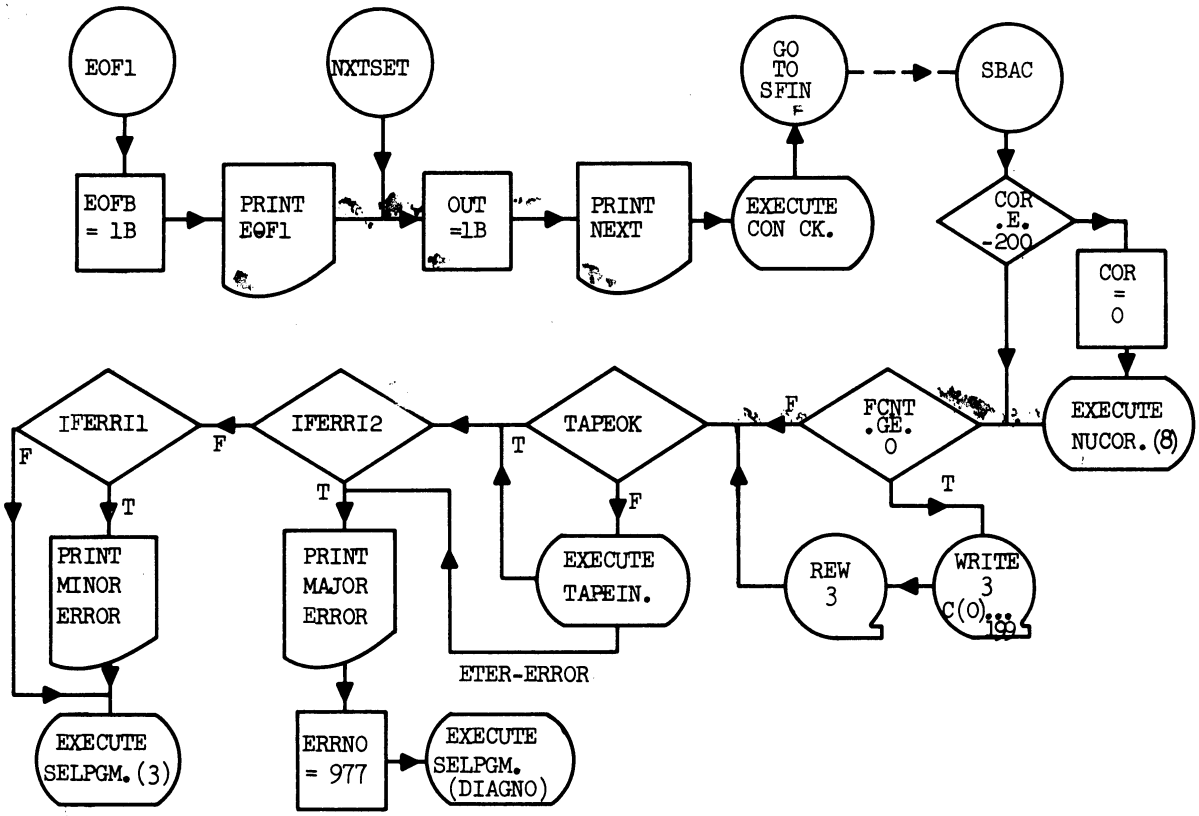


FOR OPERATIONS INVOLVING HOLLERITH CARDS.

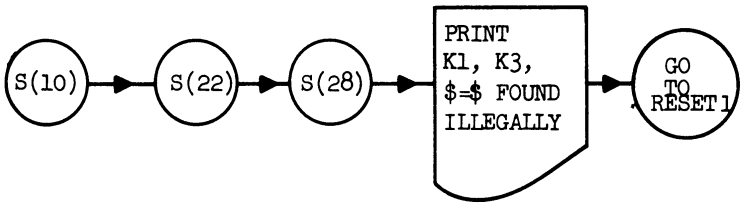

CORE 1
SYSTEM SIMULATOR
MAIN LOOP
INPUT CORE
ENTER
EOFB
F
T
REW 3
REW 4
EOF 5
COMSTA = 0
T
F
TAFCNT=-1
TAPEOK=OB
NORMAL END
F
SETH
T
SYSTEM
EXECUTE SAVTP4
BLANK CONNECTION MATRIX
TAPCNT = 0
COMSTA = 1
NNTP
EXECUTE TAPEIN.
IN TAPE
WRITE 4 $ELTAPE$ B(0)..(199)
TAPEOK = 18
B(1) = 0
B(198)=PRBI1
B(199)=PRBI2
SETH=1B
SET SWITCHES & COUNTERS
BLANK 2 & 3
ZERO A MATRIX
GO TO RESET1
READ1
READ CARD INPUT.
PRINT CARD OUTPUT.
NTAPE .OR. CLC
T
WRITE TC CARD
F
COL #1 = +
T
F
T
LOOP, FOR J=1,1, J.G.6 (CHARACTER)
LOOP, FOR I=0,1,I.G.11 (WORD)
COL #1 = /
F
EOF 4
F
CLC
NUCHAR = EXTRC. (WORD(T),J)
NUCHAR = $□$
F
T
ERRTST, FOR K=0,1,K.G.5 .OR.NUCHAR.E. +,-,/,*,0.
K.G.5
PUNTST, FOR K = 0,1,K.G.4 .OR.NUCHAR.E. ,.( ) =
GO TO LOOP
NTAPE = OB
CLC = OB
PRINT ERROR OUTPUT.
IFERRL = 1B
I = 12
J = 7
RESET 1
HOLCNT = 1
NUWORD = .
K.LE.4 .OR. HOLCNT.G.6
T
F
GO TO S(K1,K)
NUWORD = INSPTC. (NUWORD,HOLCNT NUCHAR)
HOLCNT = HOLCNT + 1
GO TO READ1
LOOP

CORE 1

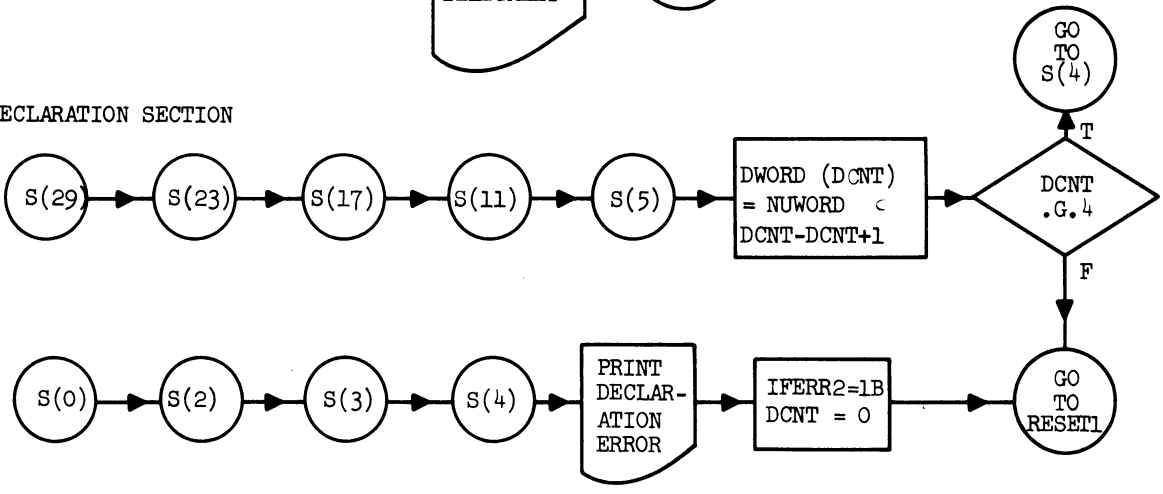
EXITS FROM CORE 1



ILLEGAL STATEMENTS

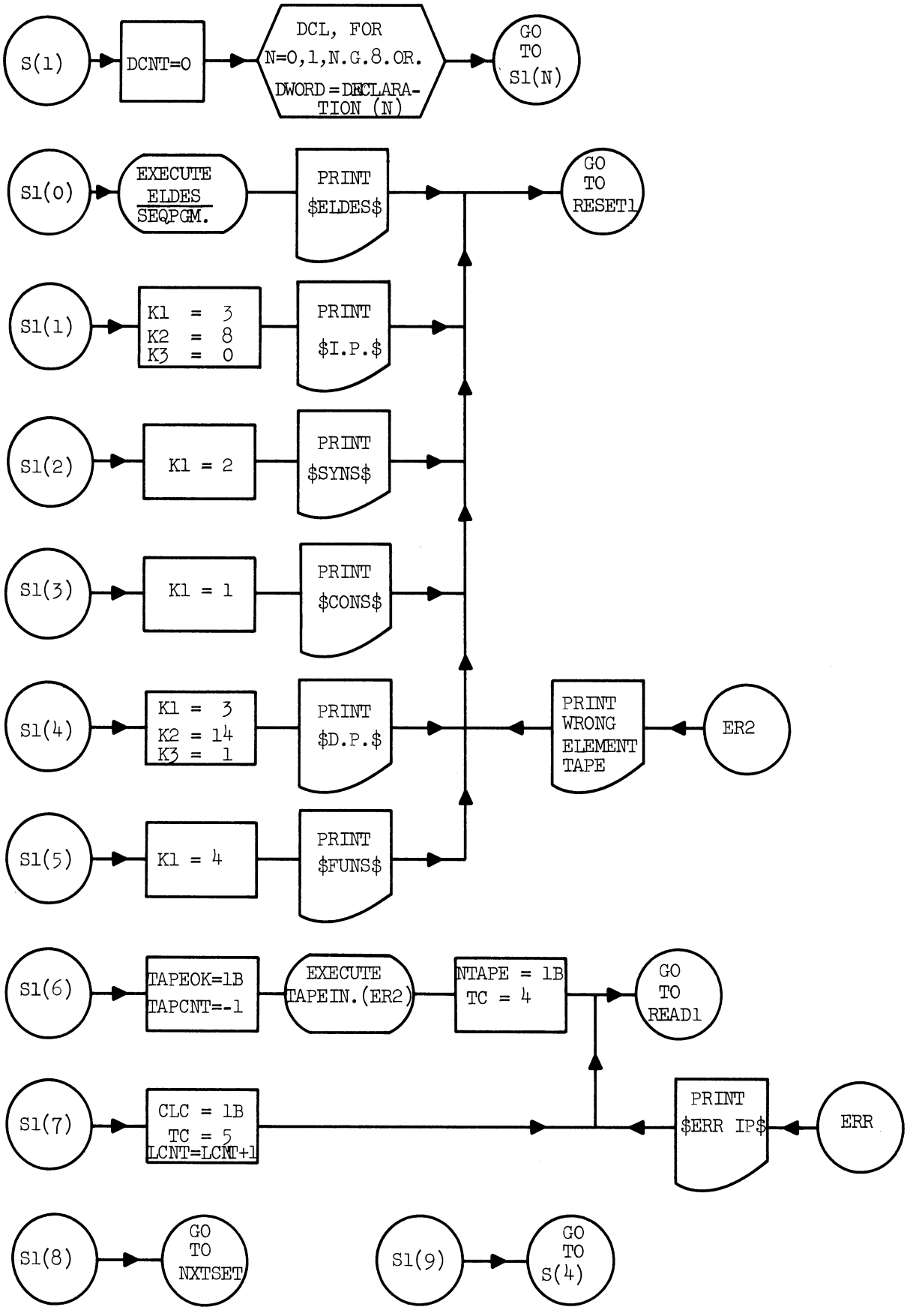


DECLARATION SECTION

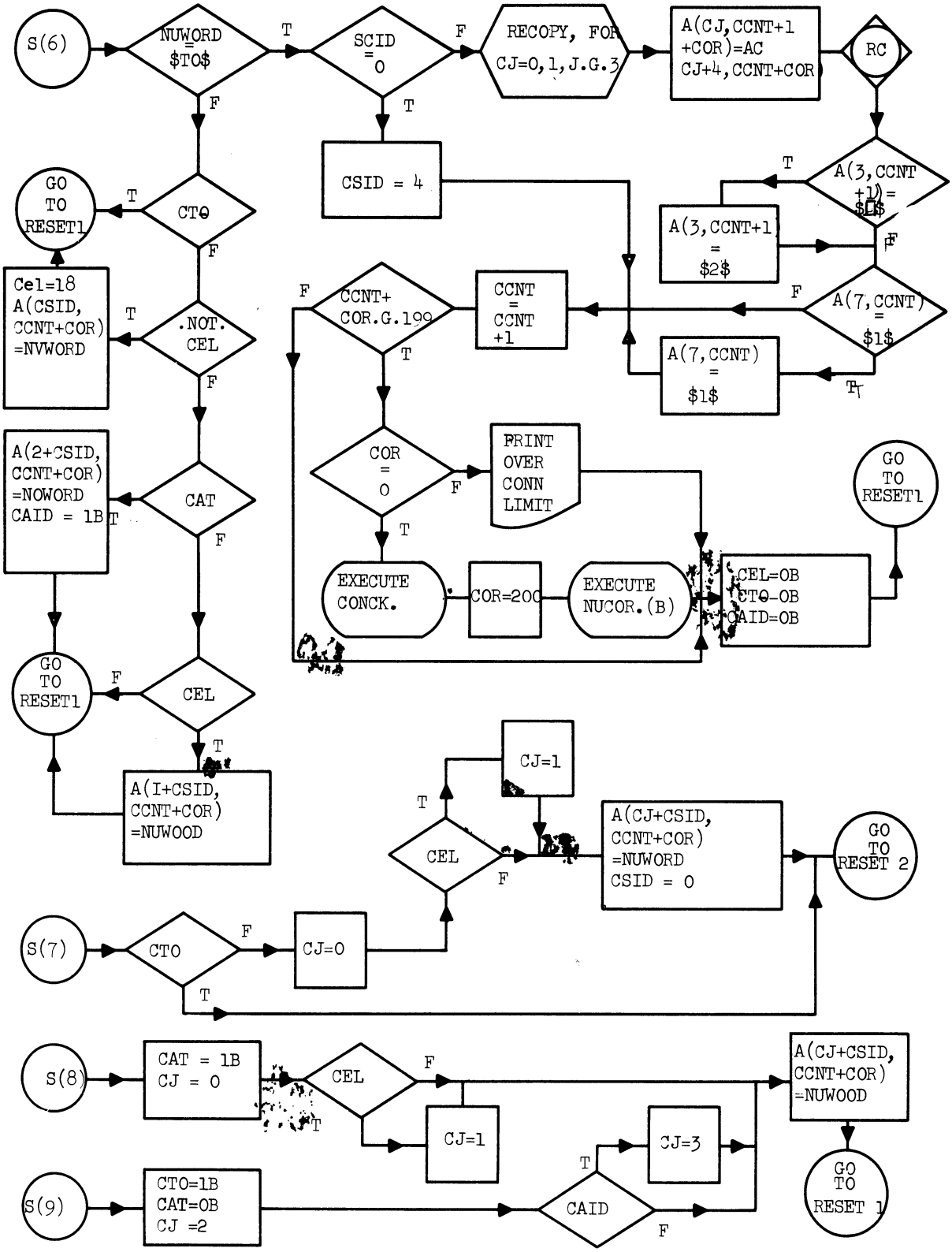


CORE 1

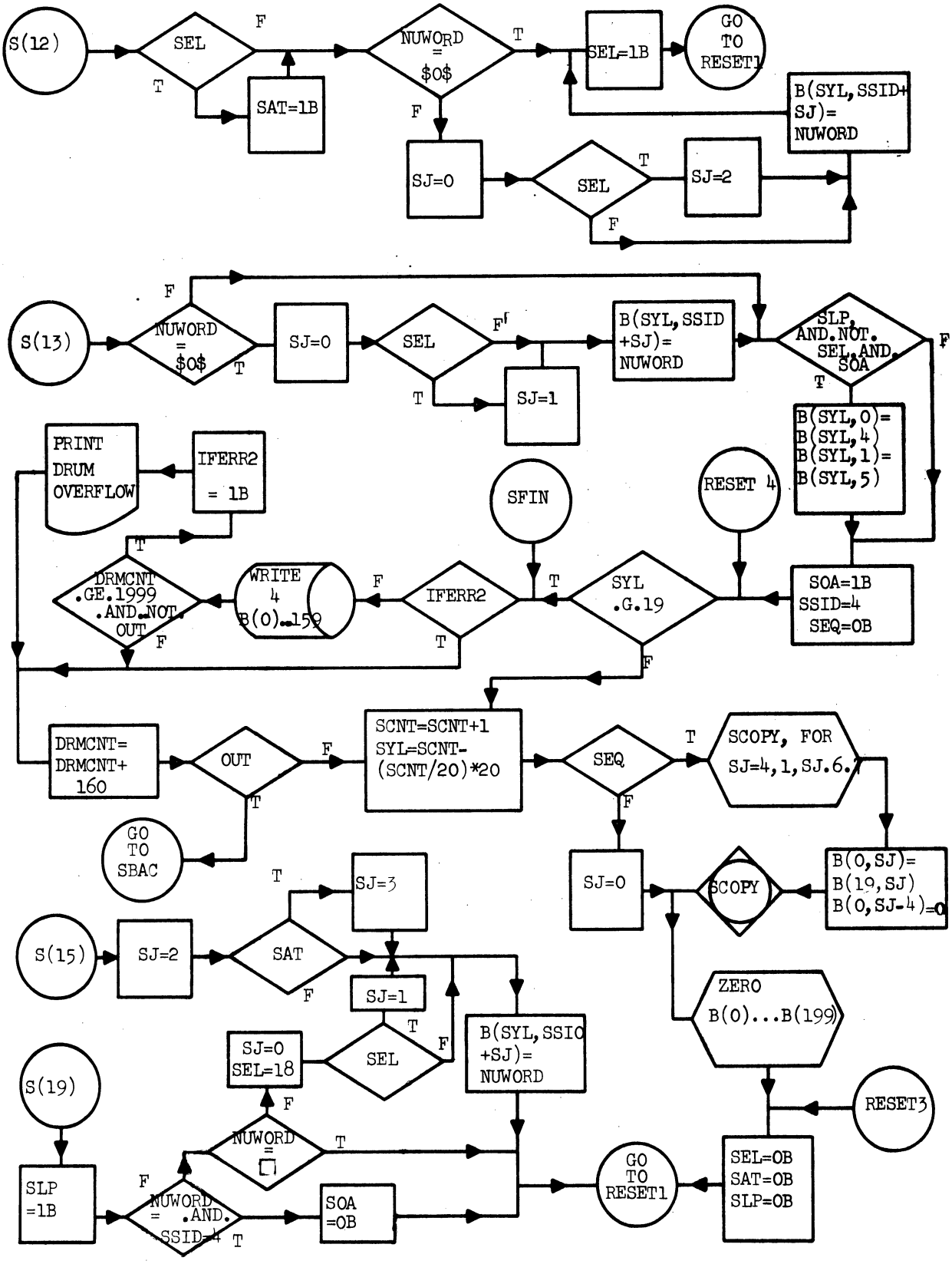
DECLARATION SECTION CONT'D



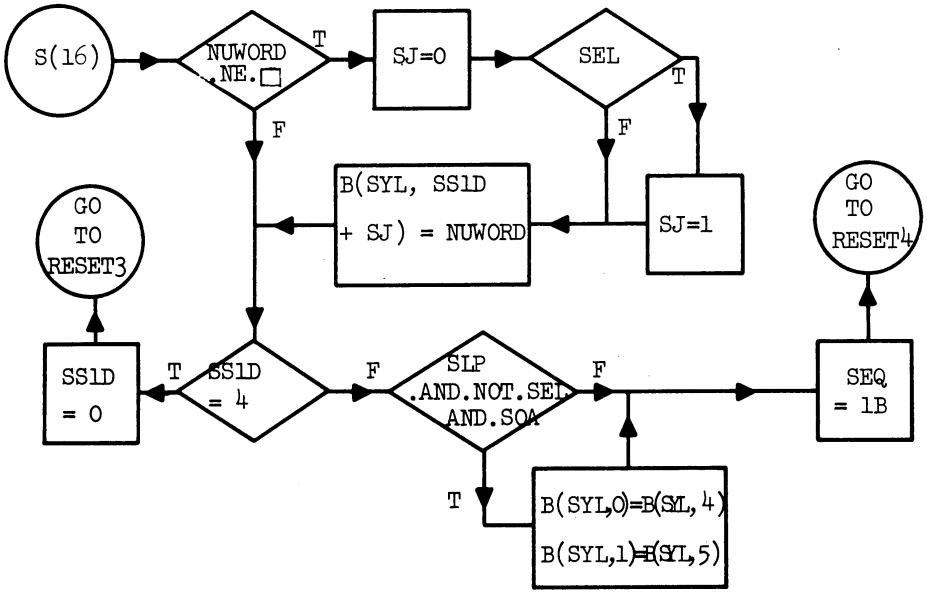
CORE 1
CONNECTION SECTION



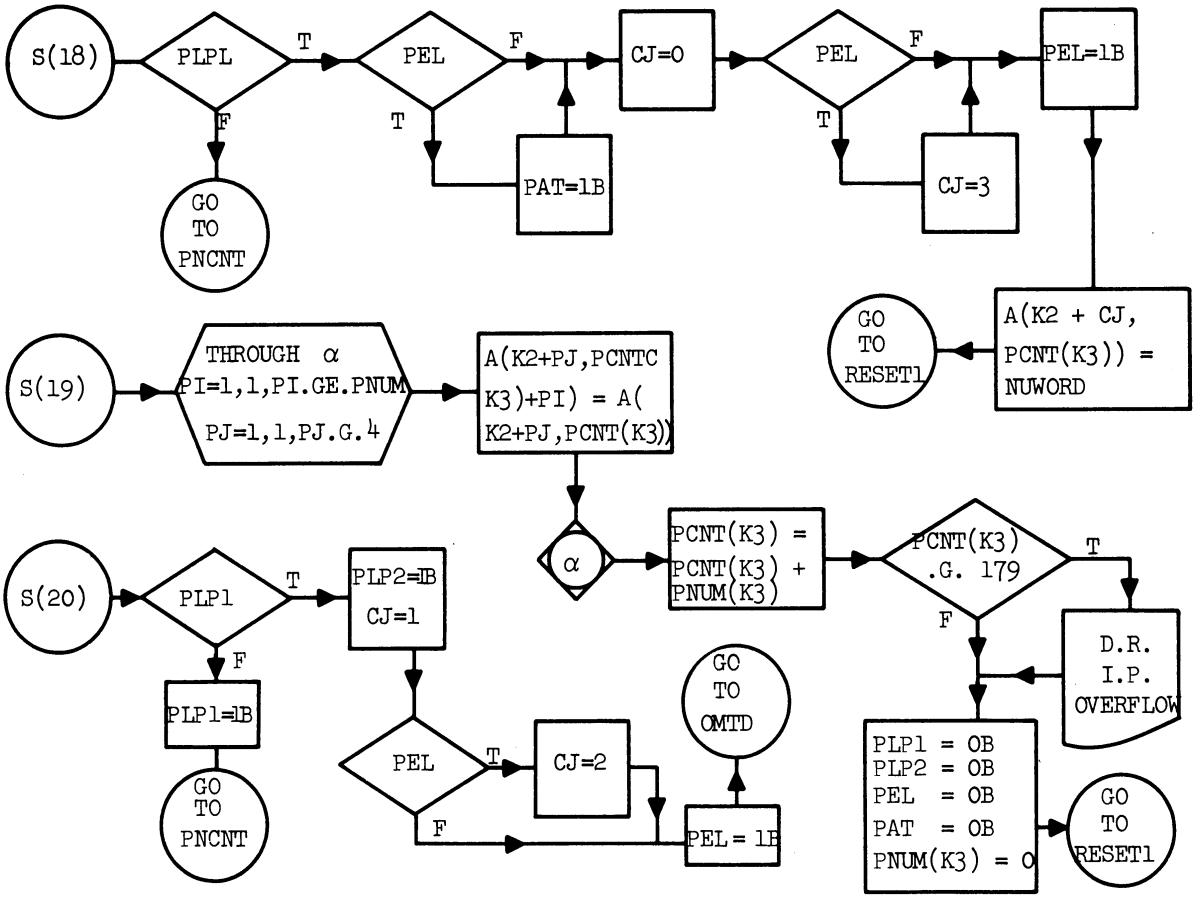
CORE 1
SYNONYM SECTION



SYNONYM SECTION CONT'D

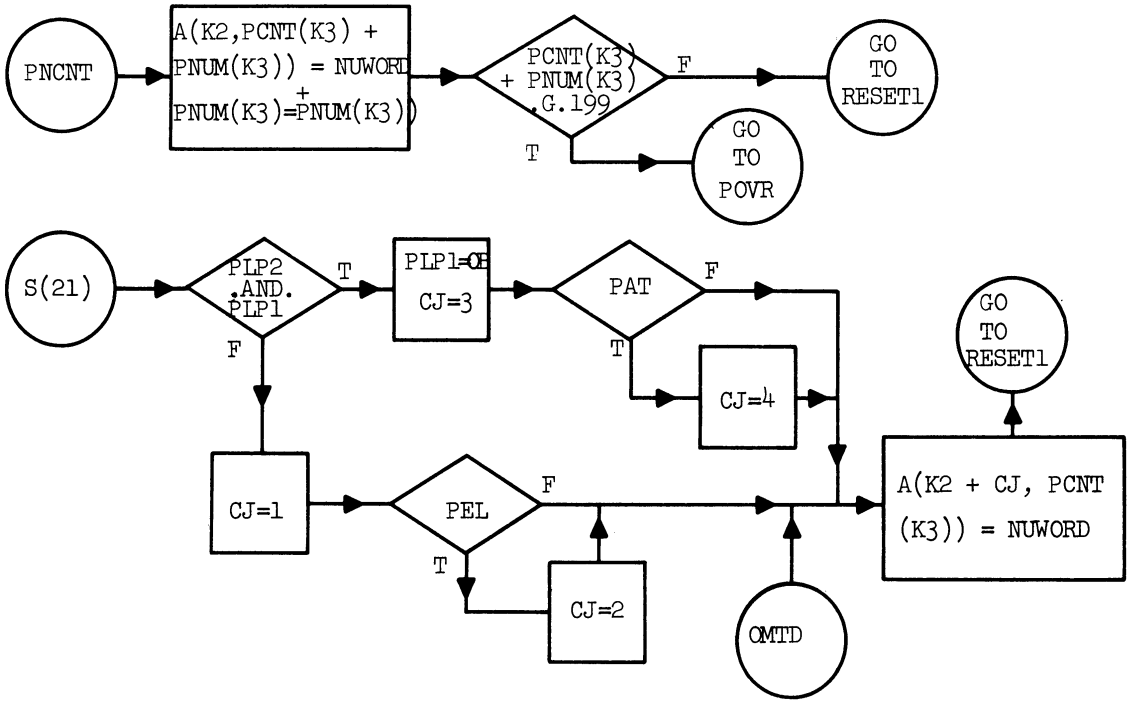


INPUT PARAMETER AND DESIRED RESULTS SECTION

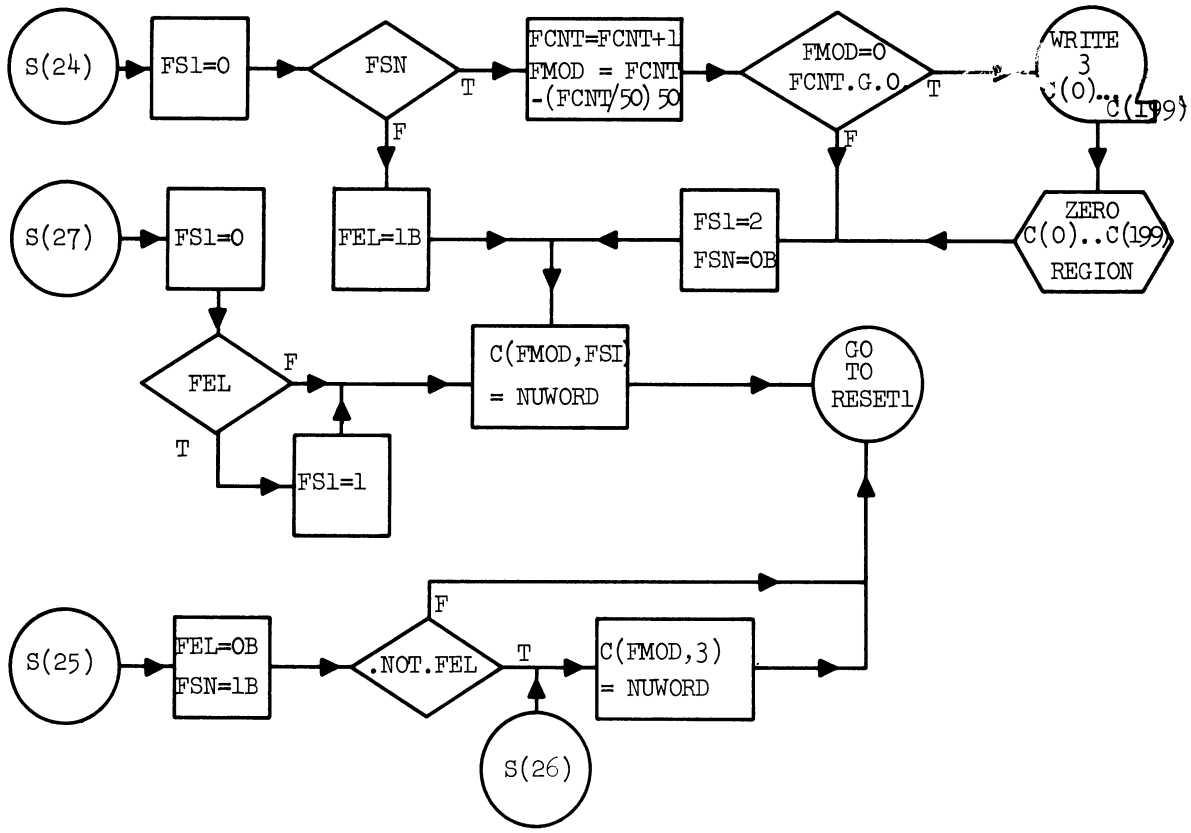


CORE 1

I.P. & D.R. CONT'D

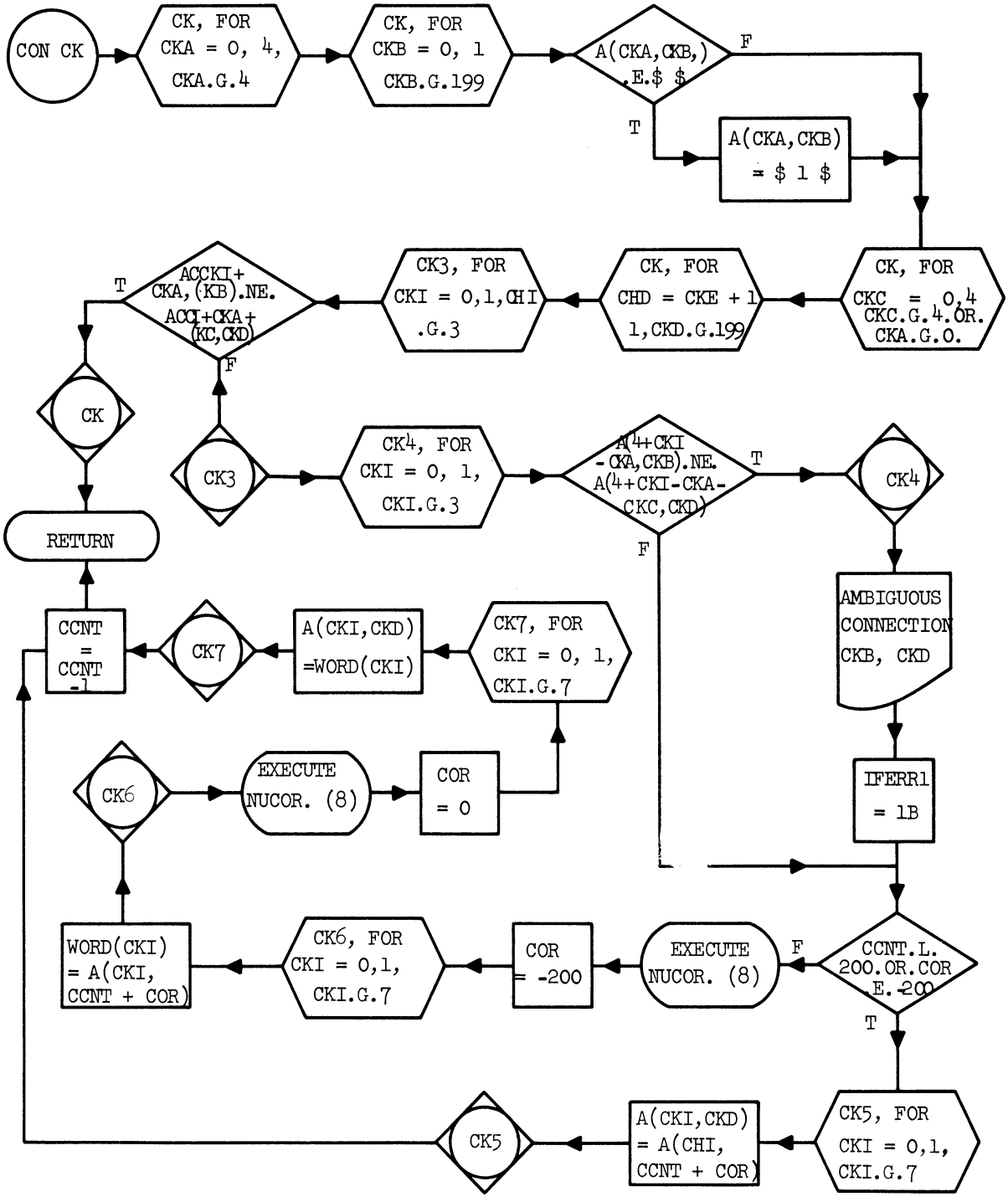


FUNCTION SUBSTITUTIONS

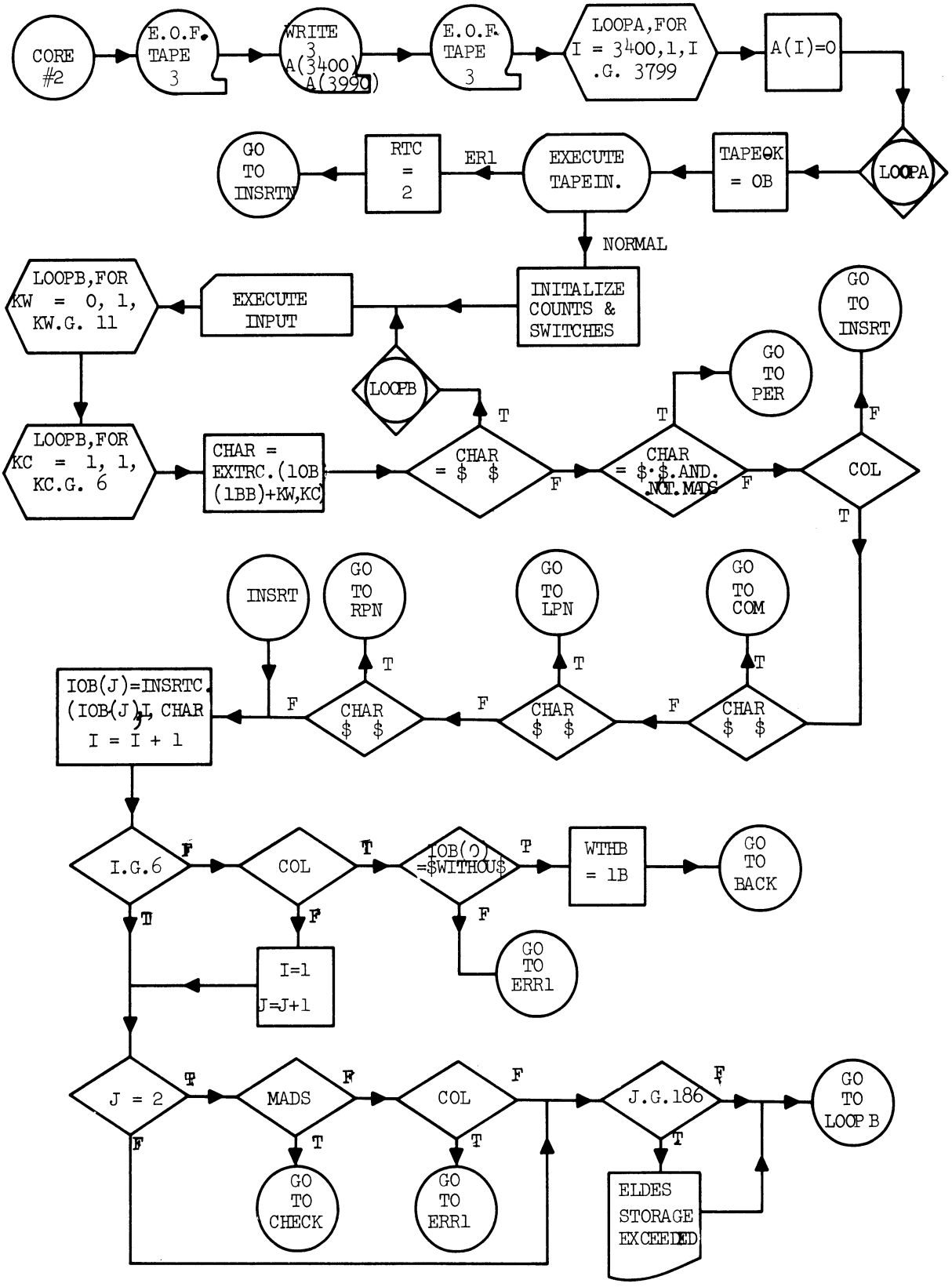


CORE 1

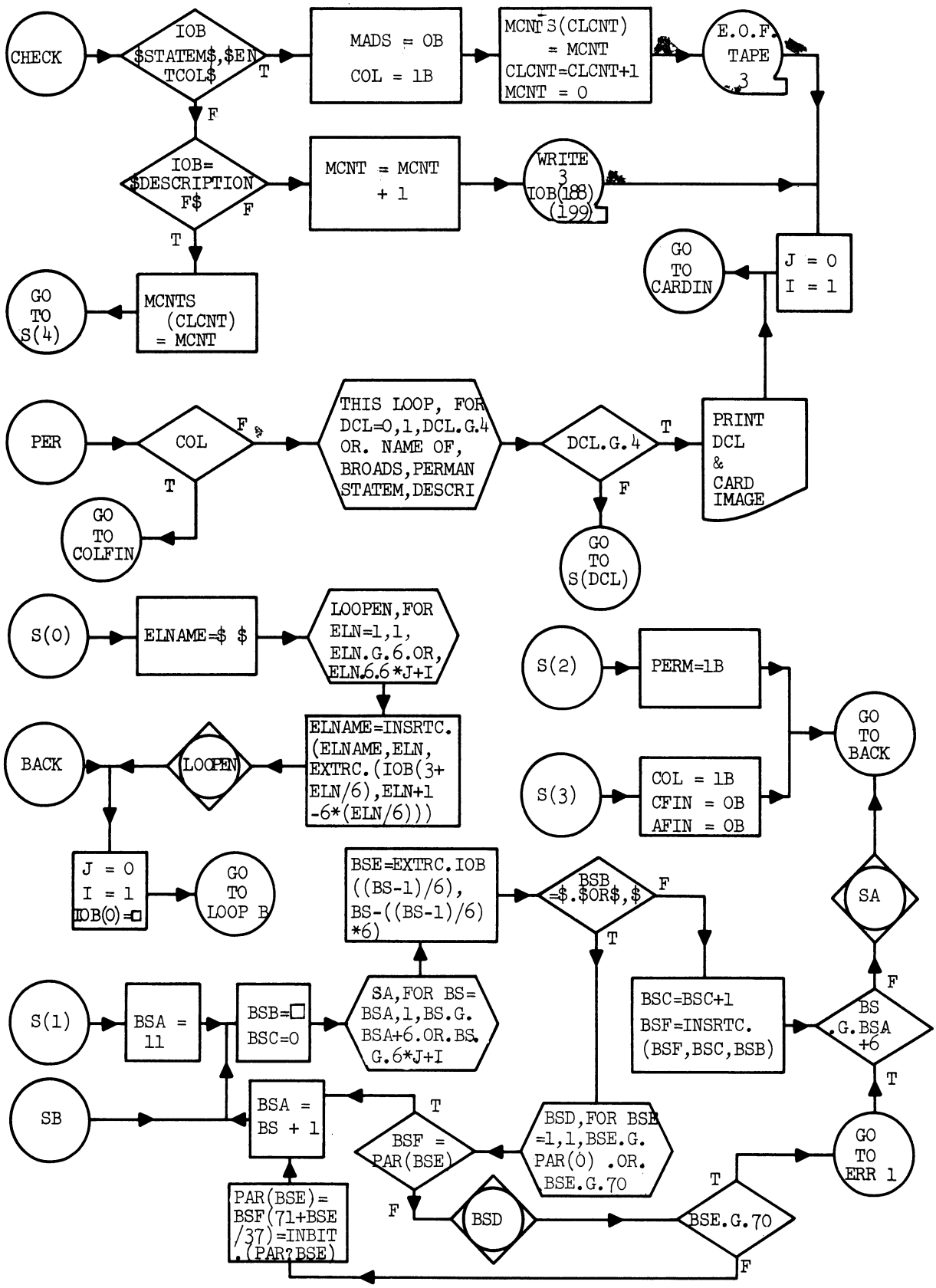
INTERNAL FUNCTION - CONCK.
(CONSISTENCY CHECK)



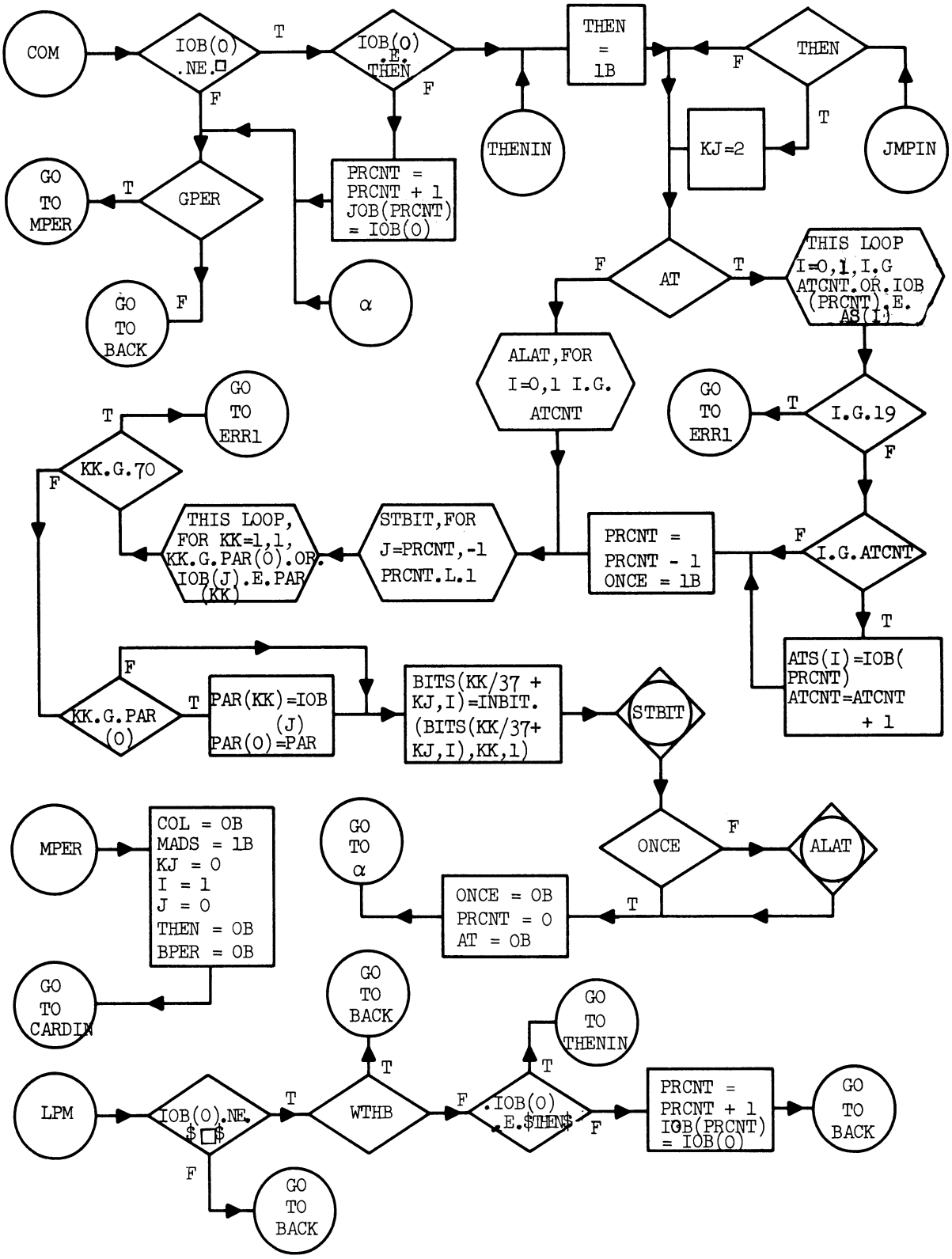
SYSTEM SIMULATOR
CORE #2 ELDES
(ELEMENT DESCRIPTION PROCESSOR)



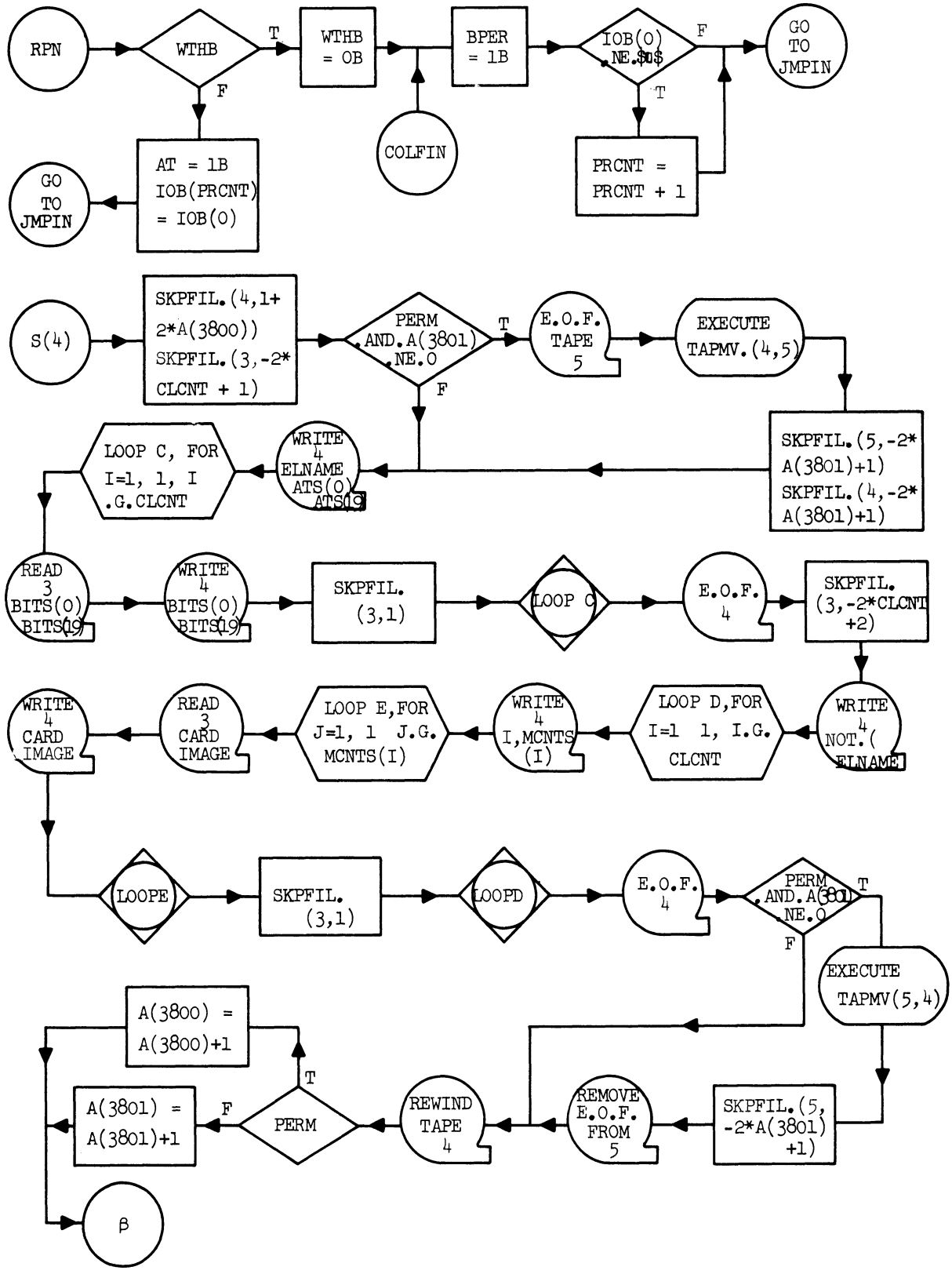
CORE #2 (CONT'D)


CHECK
IOB $STATEM$,$ENTCOL$
T
F
MADS = 0B
COL = 1B
MCNTS(CLCNT) = MCNT
CLCNT=CLCNT+1
MCNT = 0
E.O.F. TAPE 3
IOB= $DESCRIPTION F$
F
T
MCNT = MCNT + 1
WRITE 3 IOB(188) (199)
GO TO S(4)
MCNTS (CLCNT) = MCNT
GO TO CARDIN
J = 0
I = 1
PER
COL
F
T
GO TO COLFIN
THIS LOOP, FOR DCL=0,1,DCL.G.4 OR. NAME OF, BROADS,PERMAN STATEM,DESCRI
DCL.G.4
T
F
PRINT DCL & CARD IMAGE
GO TO S(DCL)
S(0)
ELNAME=$ $
LOOPEN,FOR ELN=1,1, ELN.G.6.OR, ELN.6.6*J+I
S(2)
PERM=1B
GO TO BACK
BACK
LOOPEN
ELNAME=INSRTC. (ELNAME,ELN, EXTRC.(IOB(3+ ELN/6),ELN+1 -6*(ELN/6)))
S(3)
COL = 1B
CFIN = 0B
AFIN = 0B
J = 0
I = 1
IOB(0)=□
GO TO LOOP B
BSE=EXTRC.IOB ((BS-1)/6), BS-((BS-1)/6) *6)
BSB =$.$OR$,$
F
T
SA
S(1)
BSA = 11
BSB=□
BSC=0
SA,FOR BS= BSA,1,BS.G. BSA+6.OR.BS. G.6*J+I
BSC=BSC+1
BSF=INSRTC. (BSF,BSC,BSB)
BS .G.BSA +6
F
T
SB
BSA = BS + 1
T
BSF = PAR(BSE)
BSD,FOR BSE =1,1,BSE.G. PAR(0) .OR. BSE.G.70
GO TO ERR 1
F
BSD
T
BSE.G.70
PAR(BSE)= BSF(71+BSE /37)=INBIT .(PAR?BSE)
F

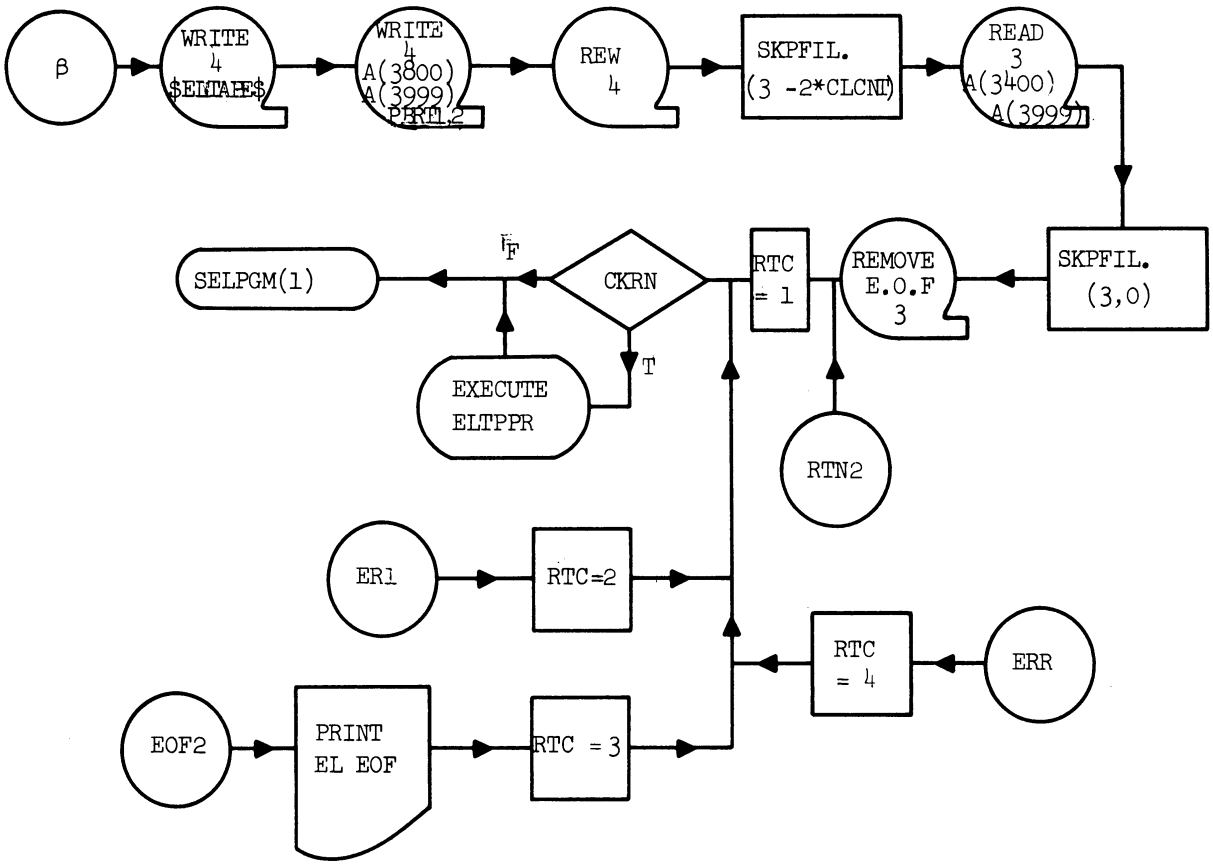
CORE #2 (cont'd)


COM
IOB(0) .NE.□
T
F
IOB(0) .E. THEN
F
THEN = 1B
THENIN
THEN
F
T
KJ=2
JMPIN
GO TO MPER
T
GPER
PRCNT = PRCNT + 1 JOB(PRCNT) = IOB(0)
GO TO BACK
F
α
AT
F
T
THIS LOOP I=0,1,I.G ATCNT.OR.IOB (PRCNT).E. AS(I)
ALAT,FOR I=0,1 I.G. ATCNT
GO TO ERR1
T
I.G.19
F
T
GO TO ERR1
KK.G.70
F
THIS LOOP, FOR KK=1,1, KK.G.PAR(0).OR. IOB(J).E.PAR (KK)
STBIT,FOR J=PRCNT,-1 PRCNT.L.1
PRCNT = PRCNT - 1 ONCE = 1B
F
I.G.ATCNT
T
ATS(I)=IOB(PRCNT) ATCNT=ATCNT + 1
F
KK.G.PAR (0)
T
PAR(KK)=IOB (J) PAR(0)=PAR
BITS(KK/37 + KJ,I)=INBIT. (BITS(KK/37+ KJ,I),KK,1)
STBIT
MPER
COL = OB MADS = 1B KJ = 0 I = 1 J = 0 THEN = OB BPER = OB
GO TO CARDIN
GO TO α
ONCE = OB PRCNT = 0 AT = OB
ONCE
F
ALAT
T
GO TO BACK
GO TO THENIN
LPM
IOB(0).NE. $□$
T
WTHB
T
F
IOB(0) .E.$THEN$
F
PRCNT = PRCNT + 1 IOB(PRCNT) = IOB(0)
GO TO BACK
F
GO TO BACK

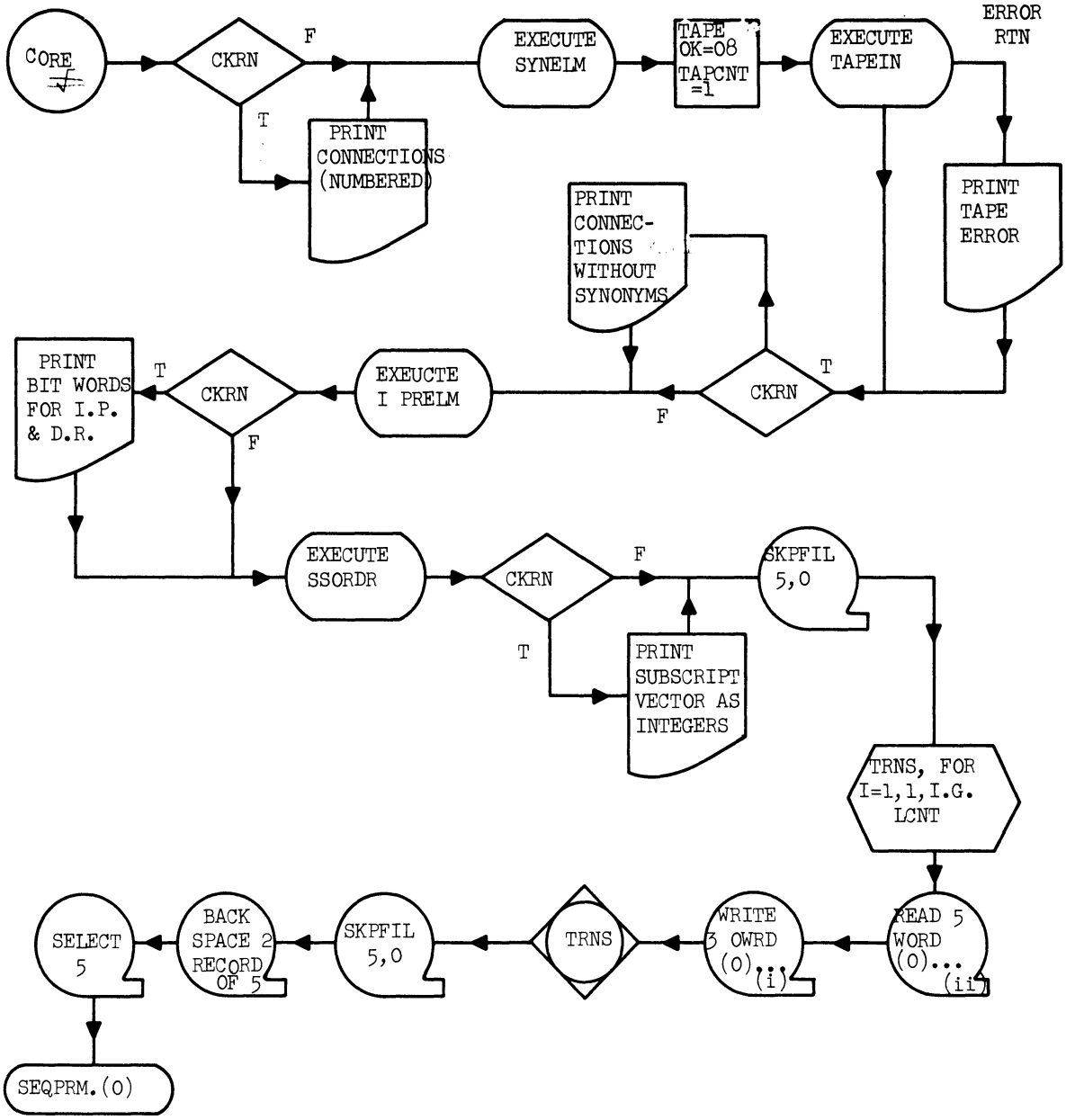
CORE #2 (CONT'D)


RPN
WTHB
T
F
WTHB = 0B
BPER = 1B
IOB(0) .NE.0
F
T
GO TO JMPIN
COLFIN
PRCNT = PRCNT + 1
AT = 1B IOB(PRCNT) = IOB(0)
GO TO JMPIN
S(4)
SKPFIL.(4,1+2*A(3800)) SKPFIL.(3,-2*CLCNT + 1)
PERM .AND.A(3801) .NE.0
T
F
E.O.F. TAPE 5
EXECUTE TAPMV.(4,5)
SKPFIL.(5,-2*A(3801)+1) SKPFIL.(4,-2*A(3801)+1)
WRITE 4 ELNAME ATS(0) ATS(19)
LOOP C, FOR I=1, 1, I .G. CLCNT
READ 3 BITS(0) BITS(19)
WRITE 4 BITS(0) BITS(19)
SKPFIL. (3,1)
LOOP C
E.O.F. 4
SKPFIL. (3,-2*CLCNT +2)
WRITE 4 NOT.(ELNAME)
LOOP D,FOR I=1 1, I.G. CLCNT
WRITE 4 I,MCNTS (I)
LOOP E,FOR J=1, 1 J.G. MCNTS(I)
READ 3 CARD IMAGE
WRITE 4 CARD IMAGE
LOOPE
SKPFIL. (3,1)
LOOPD
E.O.F. 4
PERM .AND.A(3801) .NE.0
T
F
EXECUTE TAPMV(5,4)
SKPFIL.(5, -2*A(3801) +1)
REMOVE E.O.F. FROM 5
REWIND TAPE 4
PERM
T
F
A(3801) = A(3801)+1
A(3800) = A(3800)+1
β

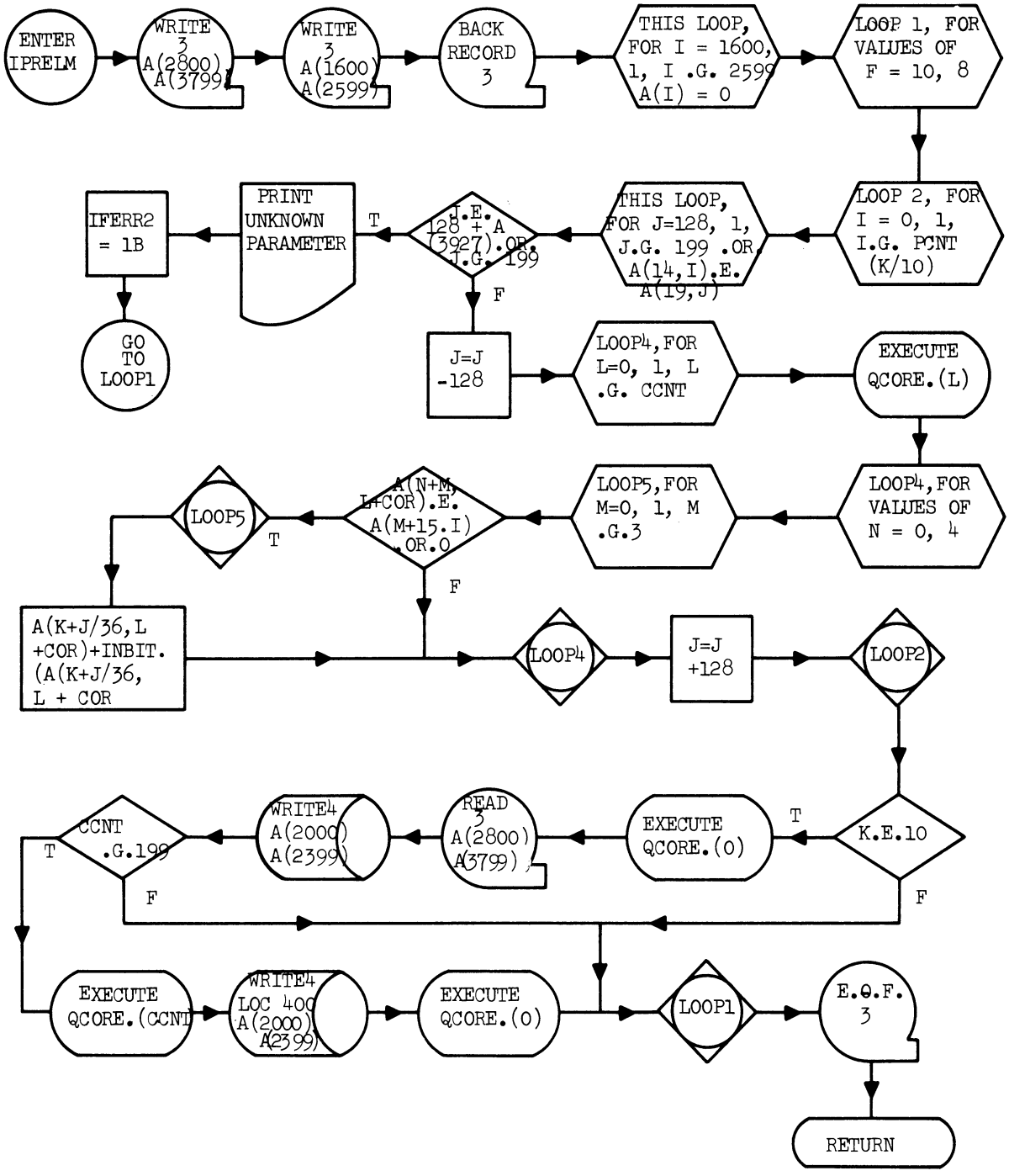
CORE #2 (cont'd)


β
WRITE 4 $ELTABE$
WRITE 4 A(3800) A(3999) PRT1.2
REW 4
SKPFIL. (3 -2*CLCNT)
READ 3 A(3400) A(3999)
SKPFIL. (3,0)
REMOVE E.O.F 3
RTC = 1
CKRN
F
T
SELPGM(1)
EXECUTE ELTPPR
RTN2
ER1
RTC=2
RTC = 4
ERR
EOF2
PRINT EL EOF
RTC = 3

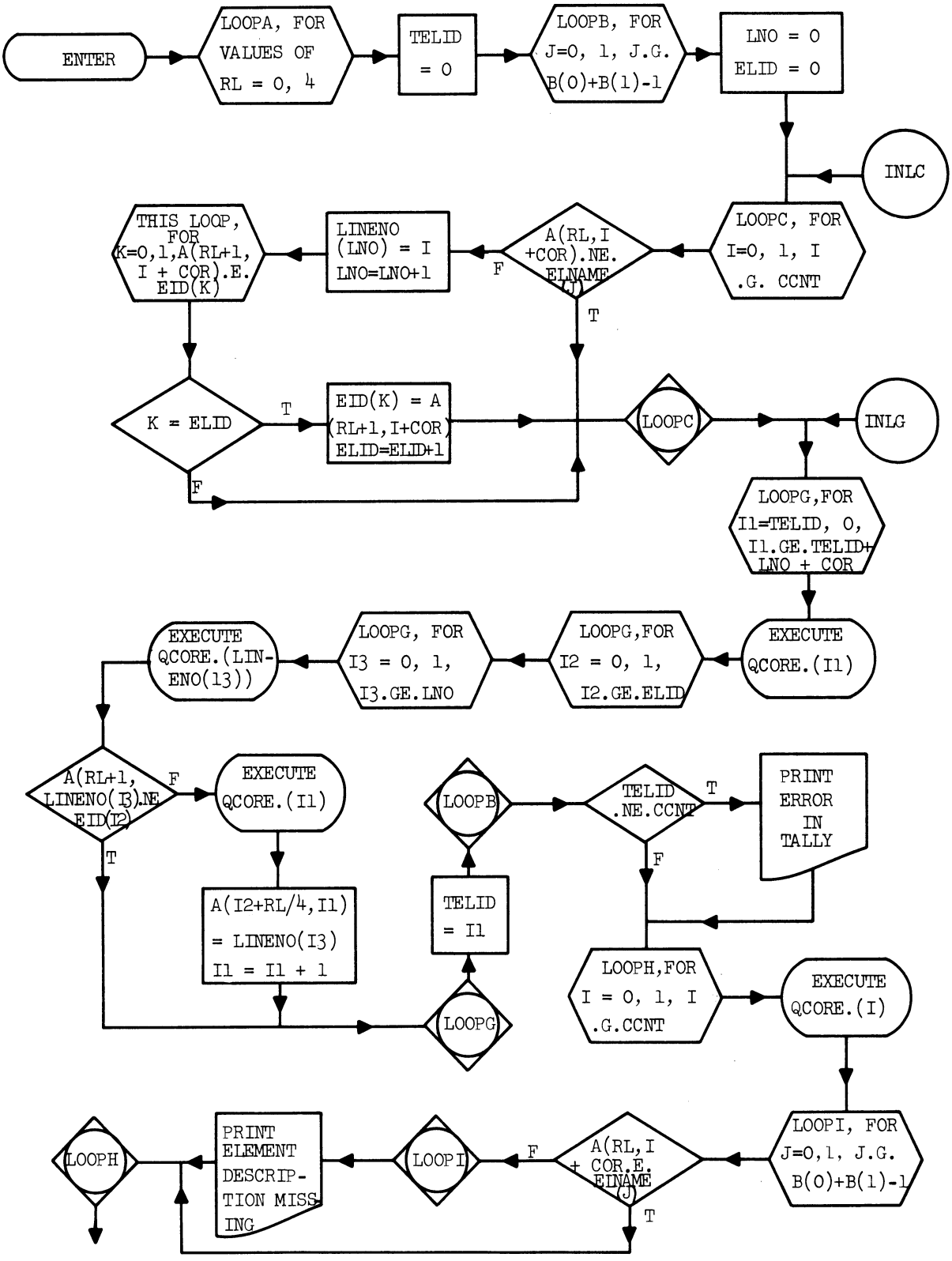
SYSTEM SIMULATOR
CORE NO. 3 SETUP CORE



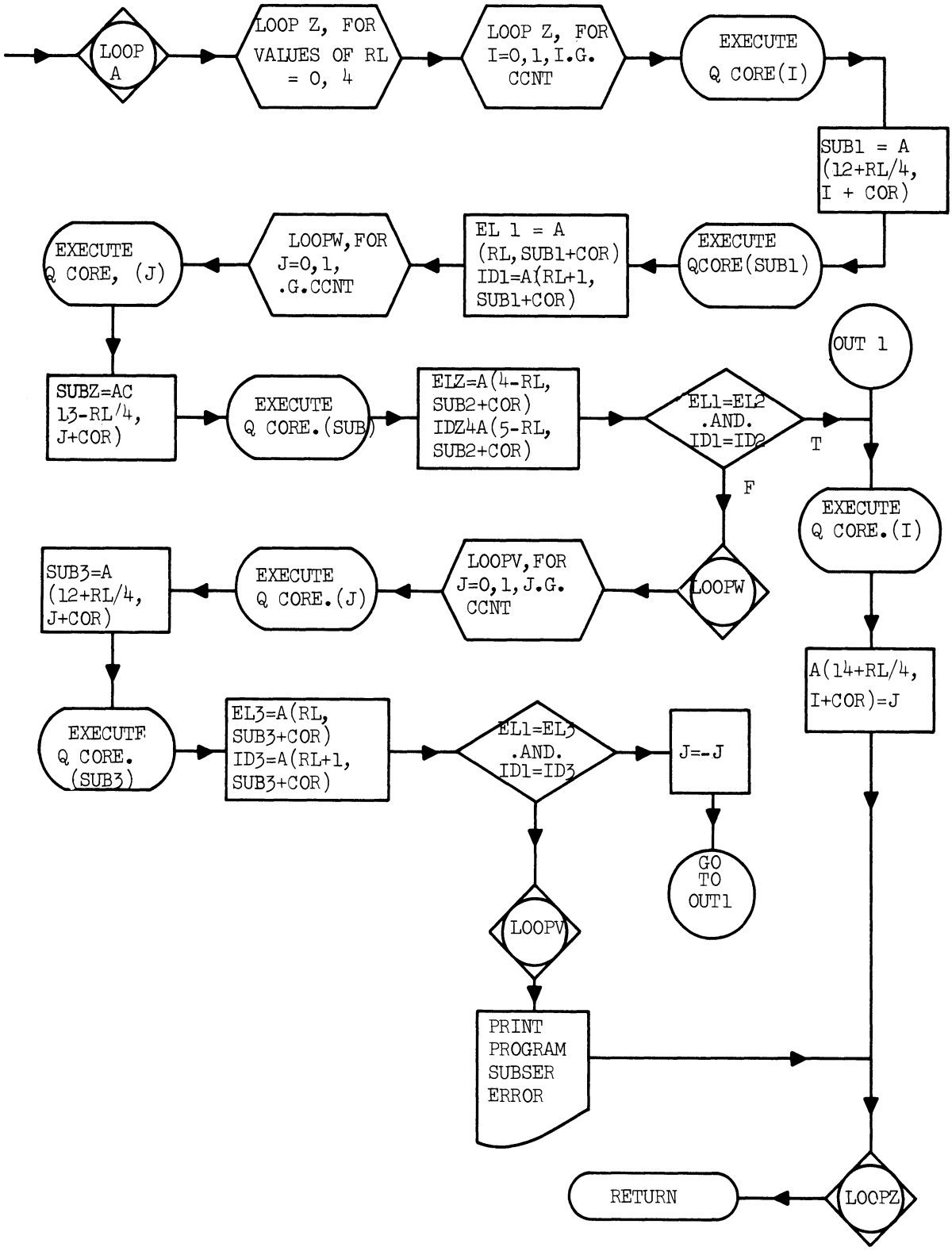
IPRELM CORE NO. 3 INITIAL PROCESSING, INPUT PARAMETERS AND DESIRED RESULTS



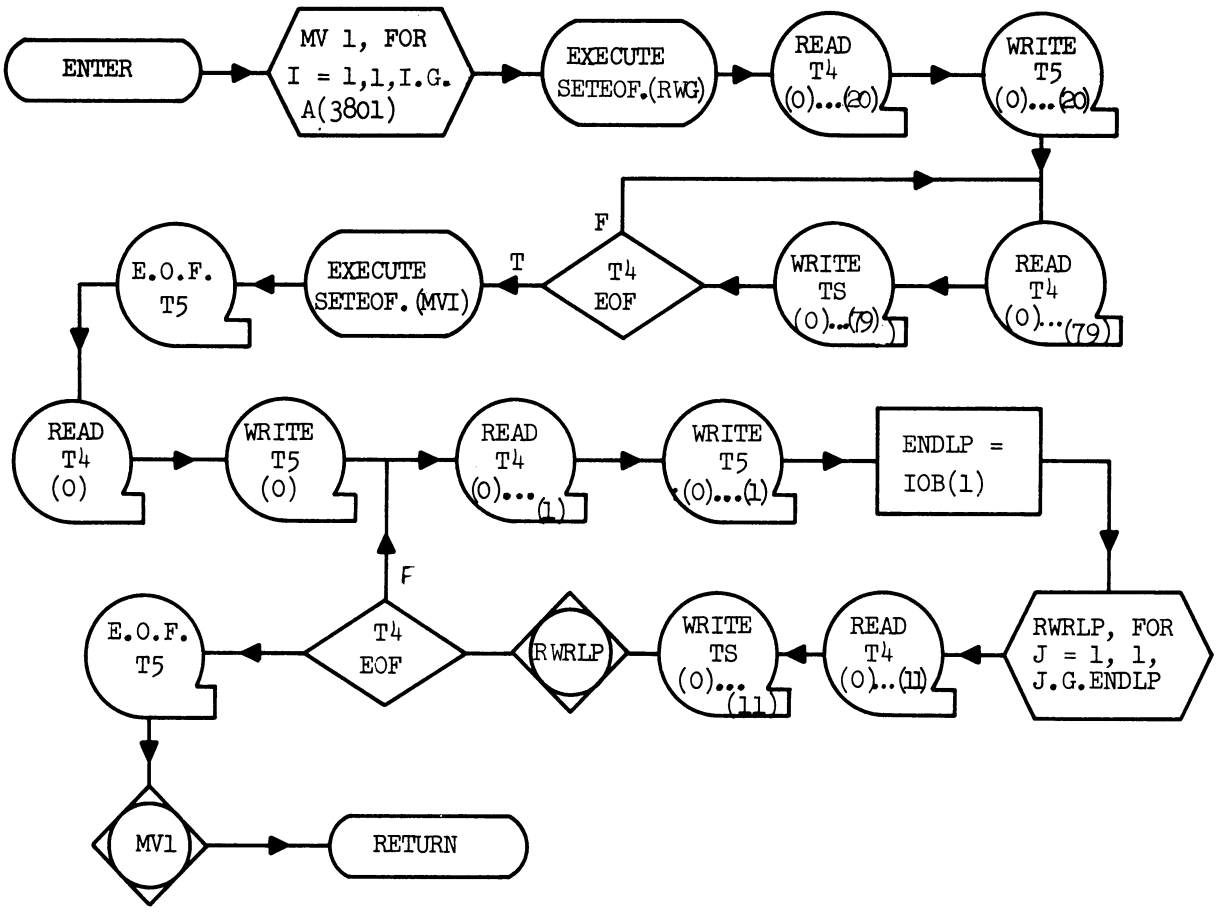
CORE # 3 SSORDR, MATRIX ORDERING ROUTINE



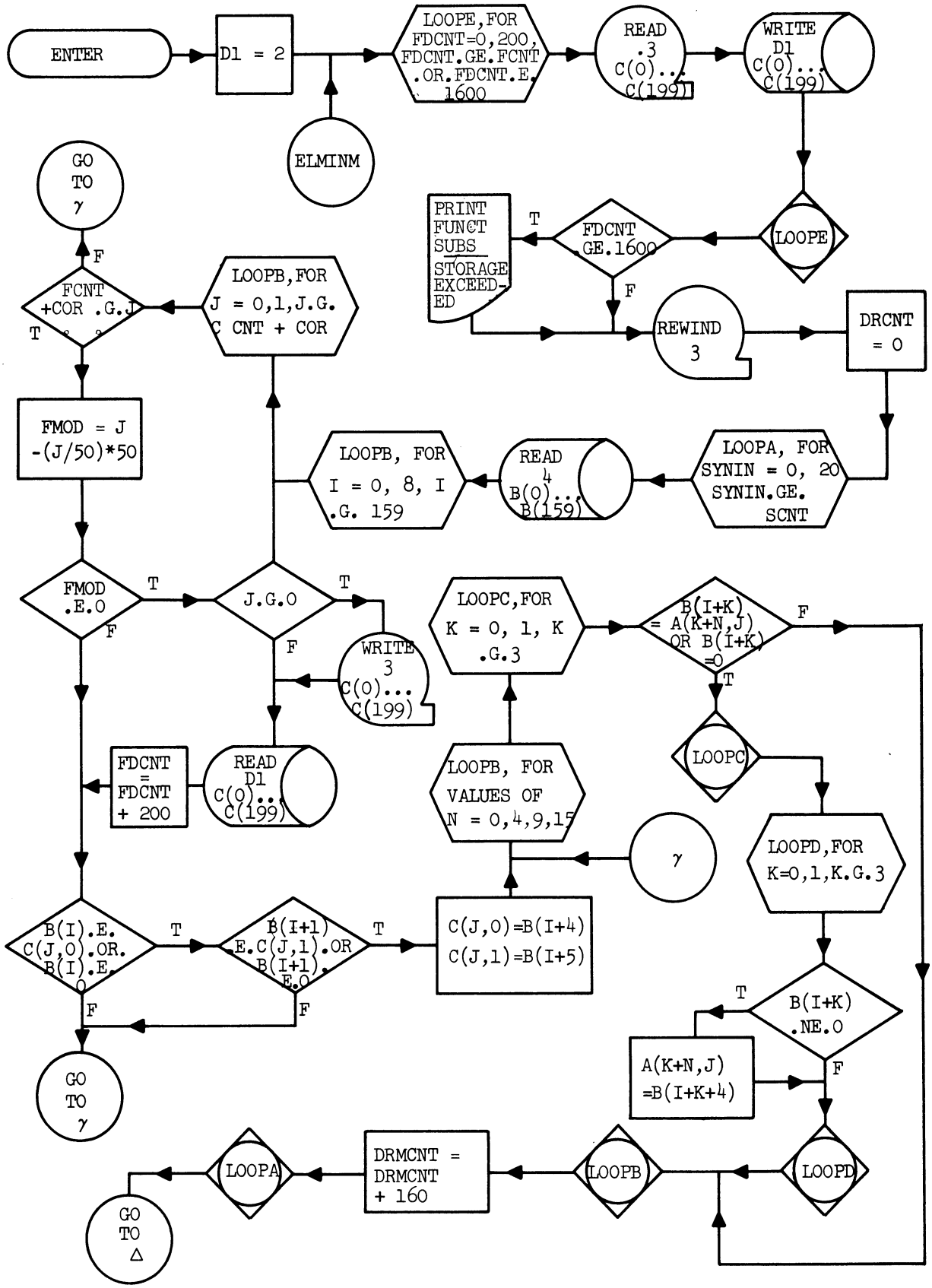
SSORDR (CONT)

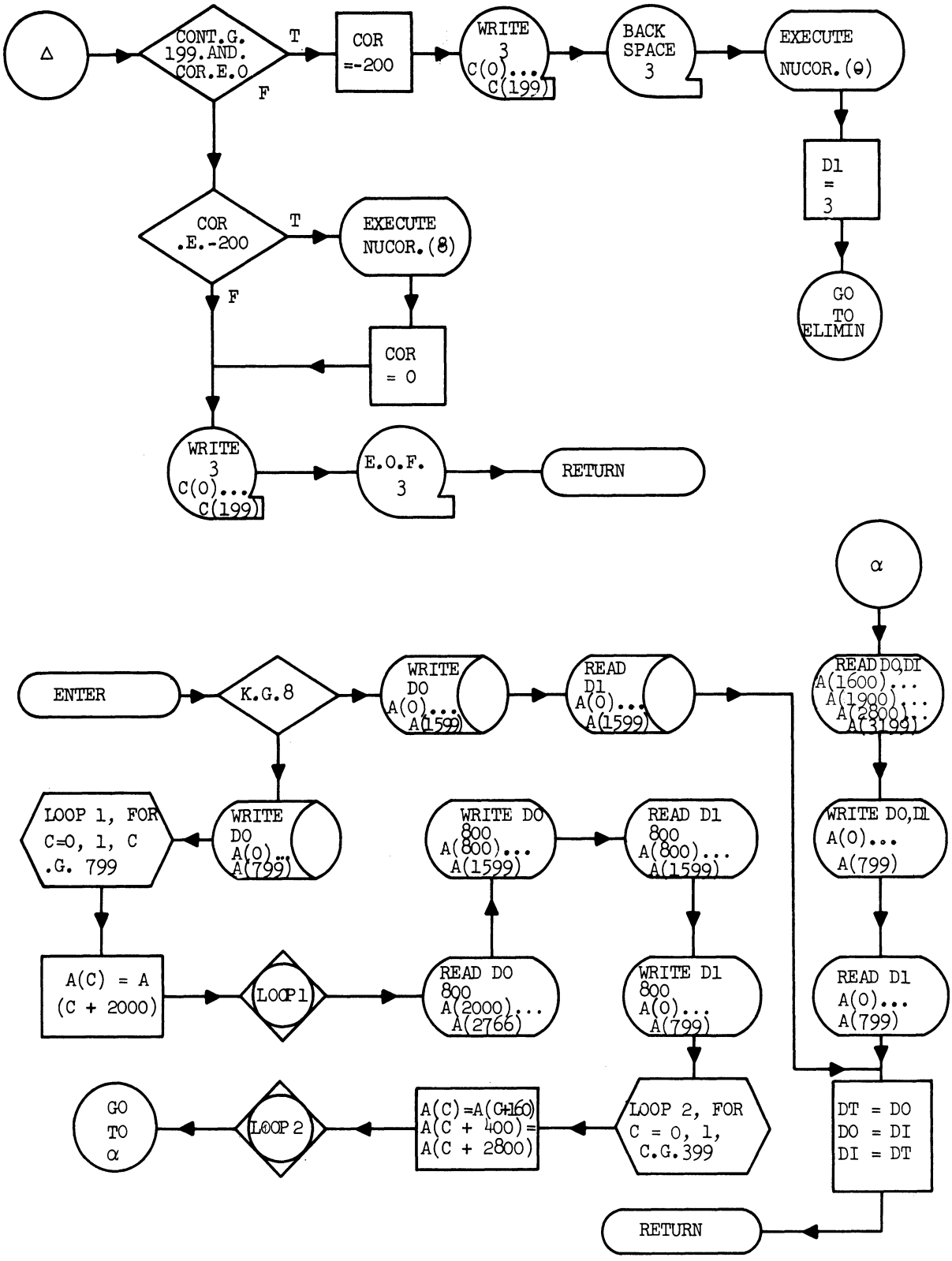

LOOP A
LOOP Z, FOR VALUES OF RL = 0, 4
LOOP Z, FOR I=0,1,I.G. CCNT
EXECUTE Q CORE(I)
SUB1 = A (12+RL/4, I + COR)
EXECUTE QCORE(SUB1)
EL 1 = A (RL, SUB1+COR) ID1=A(RL+1, SUB1+COR)
LOOPW, FOR J=0,1, .G.CCNT
EXECUTE Q CORE, (J)
OUT 1
SUBZ=AC 13-RL/4, J+COR)
EXECUTE Q CORE.(SUB)
ELZ=A(4-RL, SUB2+COR) IDZ4A(5-RL, SUB2+COR)
EL1=EL2 .AND. ID1=ID2
T
F
EXECUTE Q CORE.(I)
LOOPW
LOOPV, FOR J=0,1, J.G. CCNT
EXECUTE Q CORE.(J)
SUB3=A (12+RL/4, J+COR)
A(14+RL/4, I+COR)=J
EXECUTF Q CORE. (SUB3)
EL3=A(RL, SUB3+COR) ID3=A(RL+1, SUB3+COR)
EL1=EL3 .AND. ID1=ID3
J=-J
GO TO OUT1
LOOPV
PRINT PROGRAM SUBSER ERROR
RETURN
LOOPZ

TAPMV.(TH,TE) TAPE MOVER

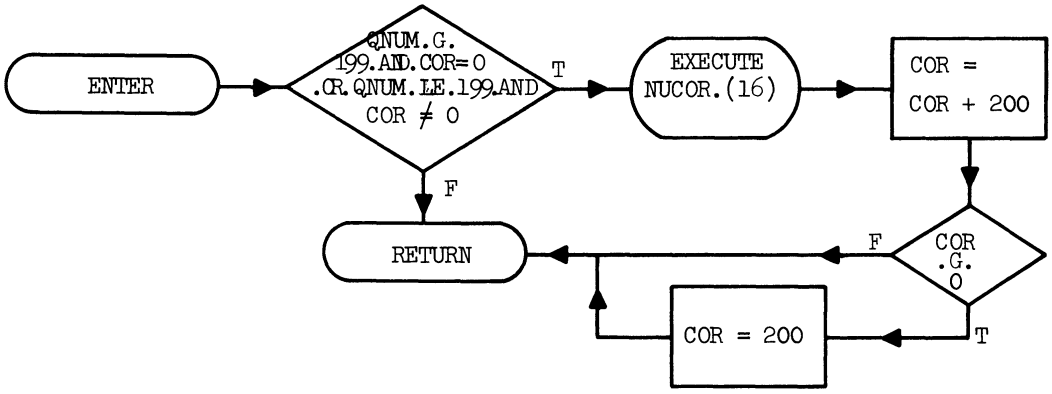


CORE #3 SYNELM SYNONYM ELIMINATION

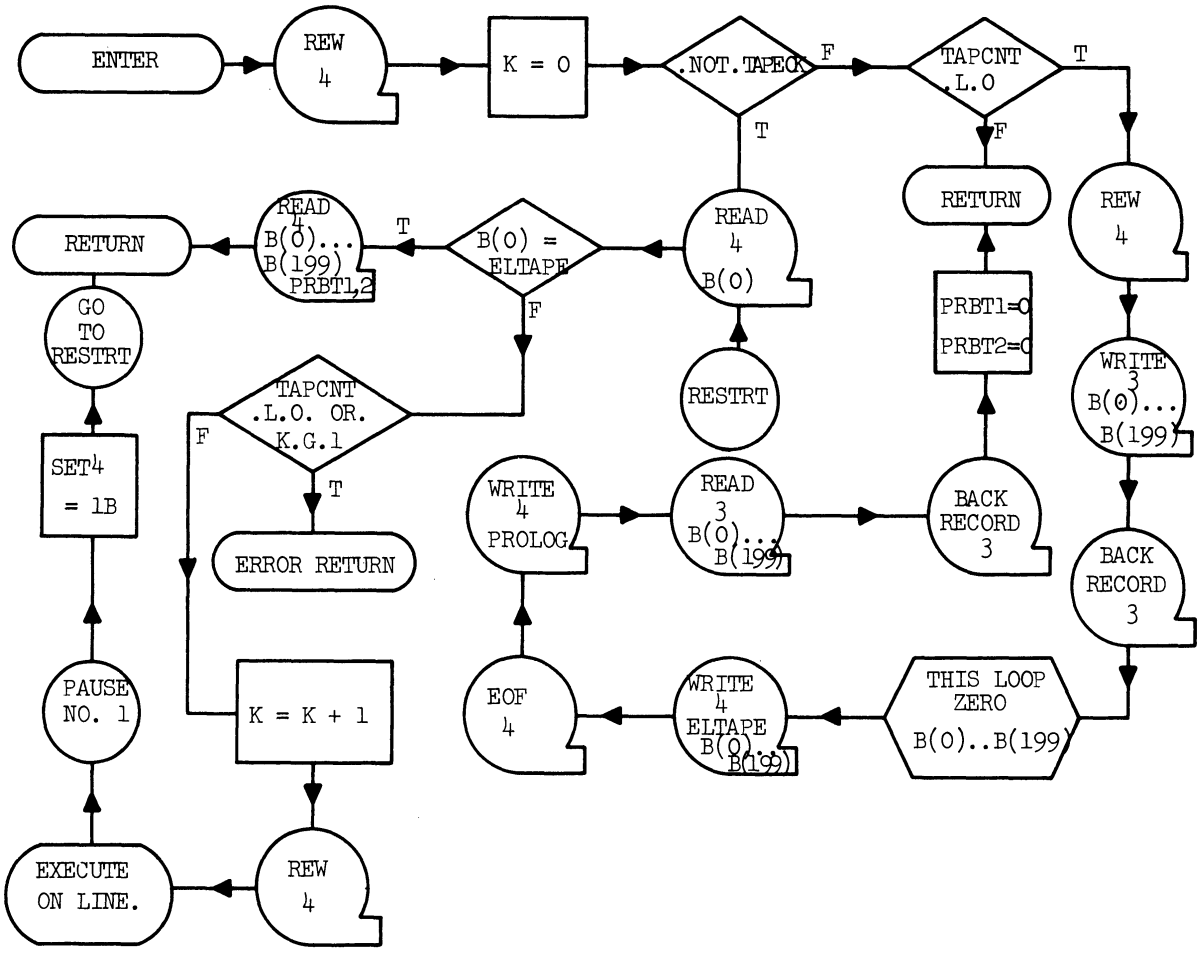



Δ
CONT.G. 199.AND. COR.E.0
T
F
COR =-200
WRITE 3 C(0)... C(199)
BACK SPACE 3
EXECUTE NUCOR.(9)
D1 = 3
GO TO ELIMIN
COR .E.-200
T
F
EXECUTE NUCOR.(8)
COR = 0
WRITE 3 C(0)... C(199)
E.O.F. 3
RETURN
α
ENTER
K.G.8
WRITE DO A(0)... A(1599)
READ D1 A(0)... A(1599)
READ DO,DI A(1600)... A(1900)... A(2800)... A(3199)
LOOP 1, FOR C=0, 1, C .G. 799
WRITE DO A(0)... A(799)
WRITE DO 800 A(800)... A(1599)
READ D1 800 A(800)... A(1599)
WRITE DO,DI A(0)... A(799)
A(C) = A (C + 2000)
LOOP1
READ DO 800 A(2000)... A(2766)
WRITE D1 800 A(0)... A(799)
READ D1 A(0)... A(799)
GO TO α
LOOP 2
A(C)=A(C+160) A(C + 400)= A(C + 2800)
LOOP 2, FOR C = 0, 1, C.G.399
DT = DO DO = DI DI = DT
RETURN

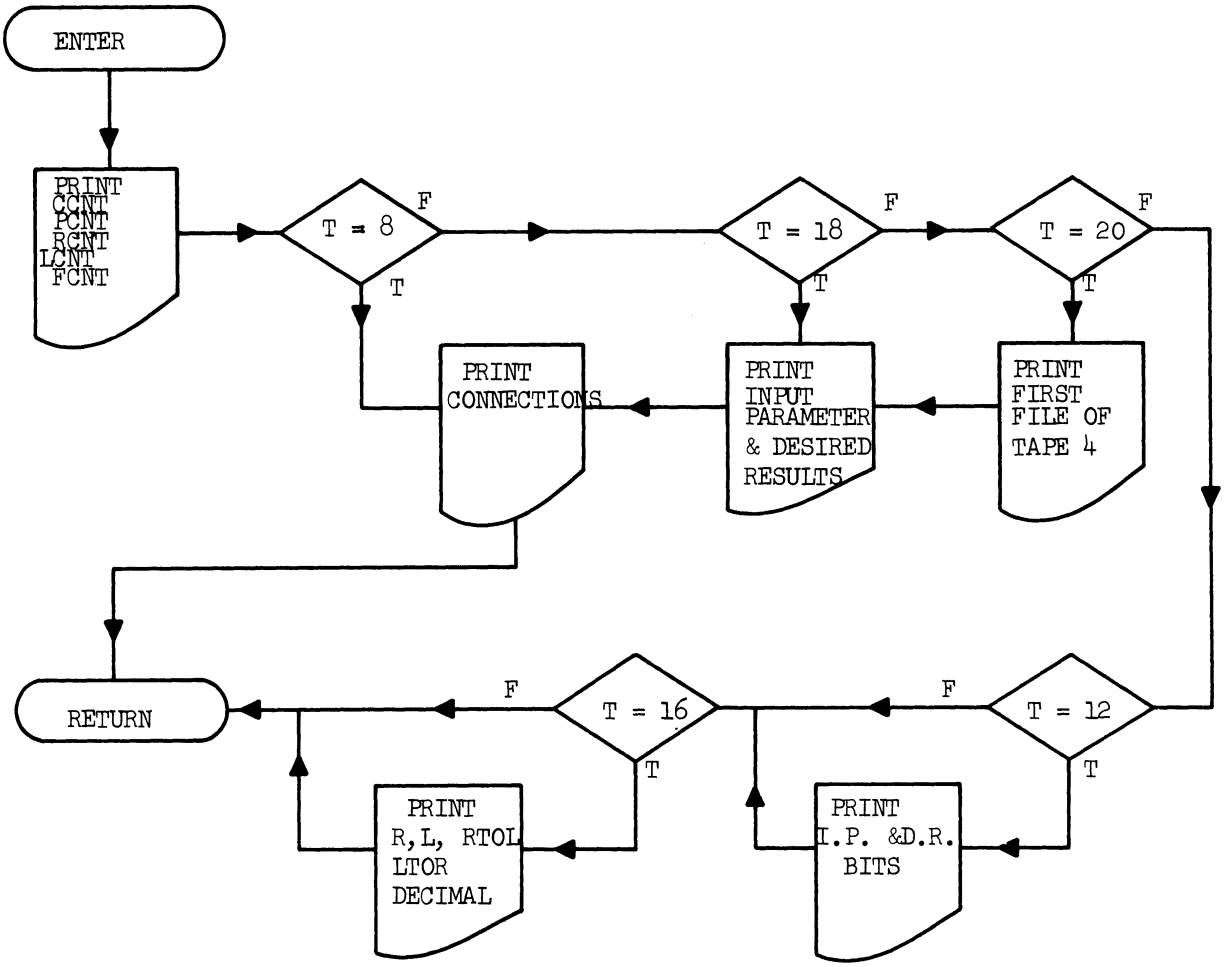
CORE # 2 QCORE



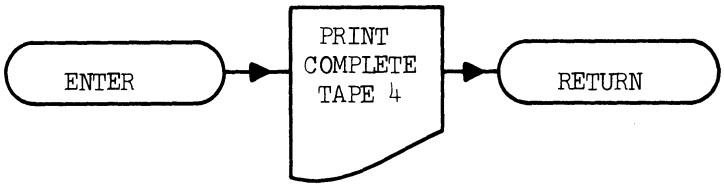
TAPEIN, ELEMENT TAPE CHECKOUT ROUTINE



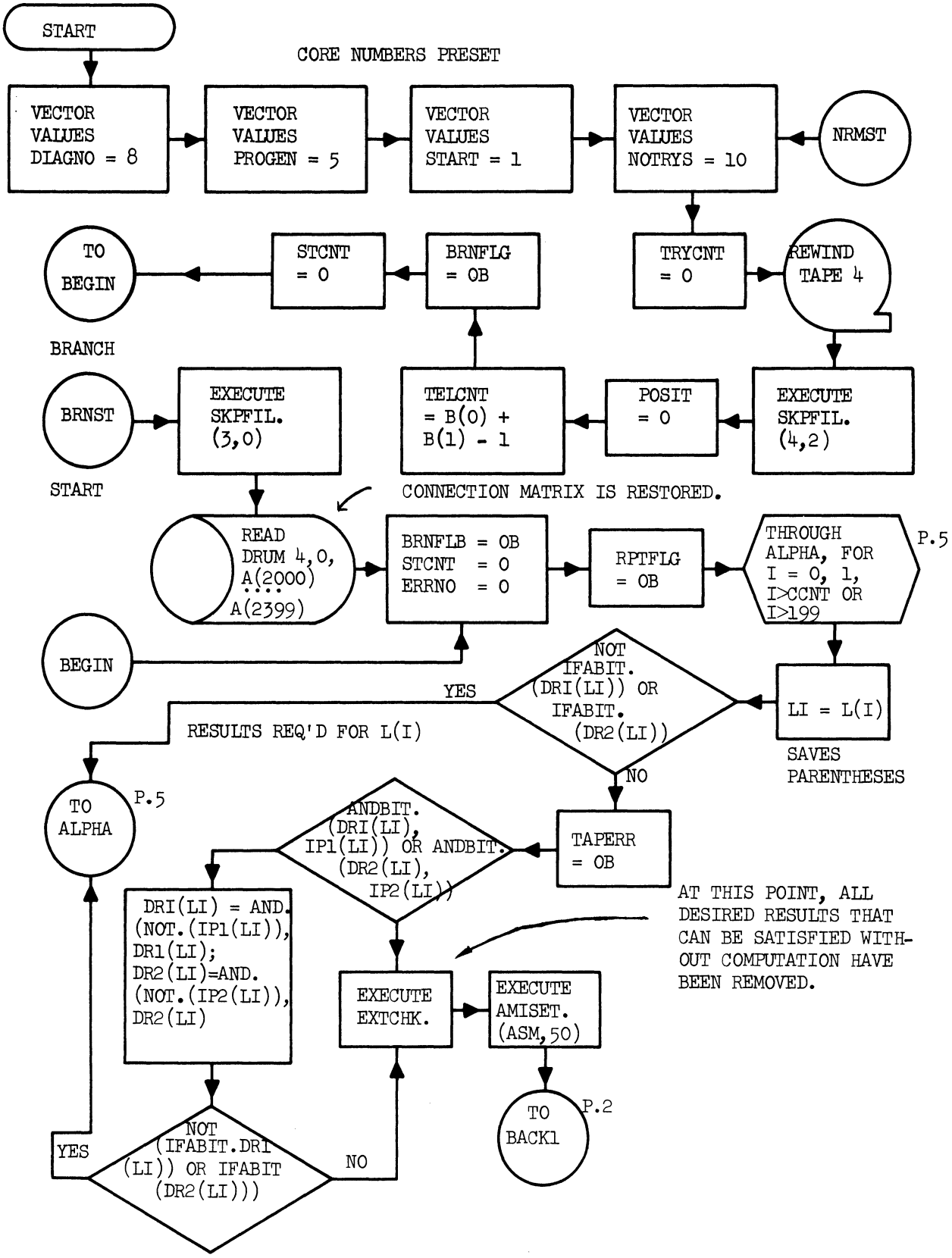
TSTDMP, TEST DUMP ROUTINE, FOR CHECKING OUT PROCEDURES
Core 3

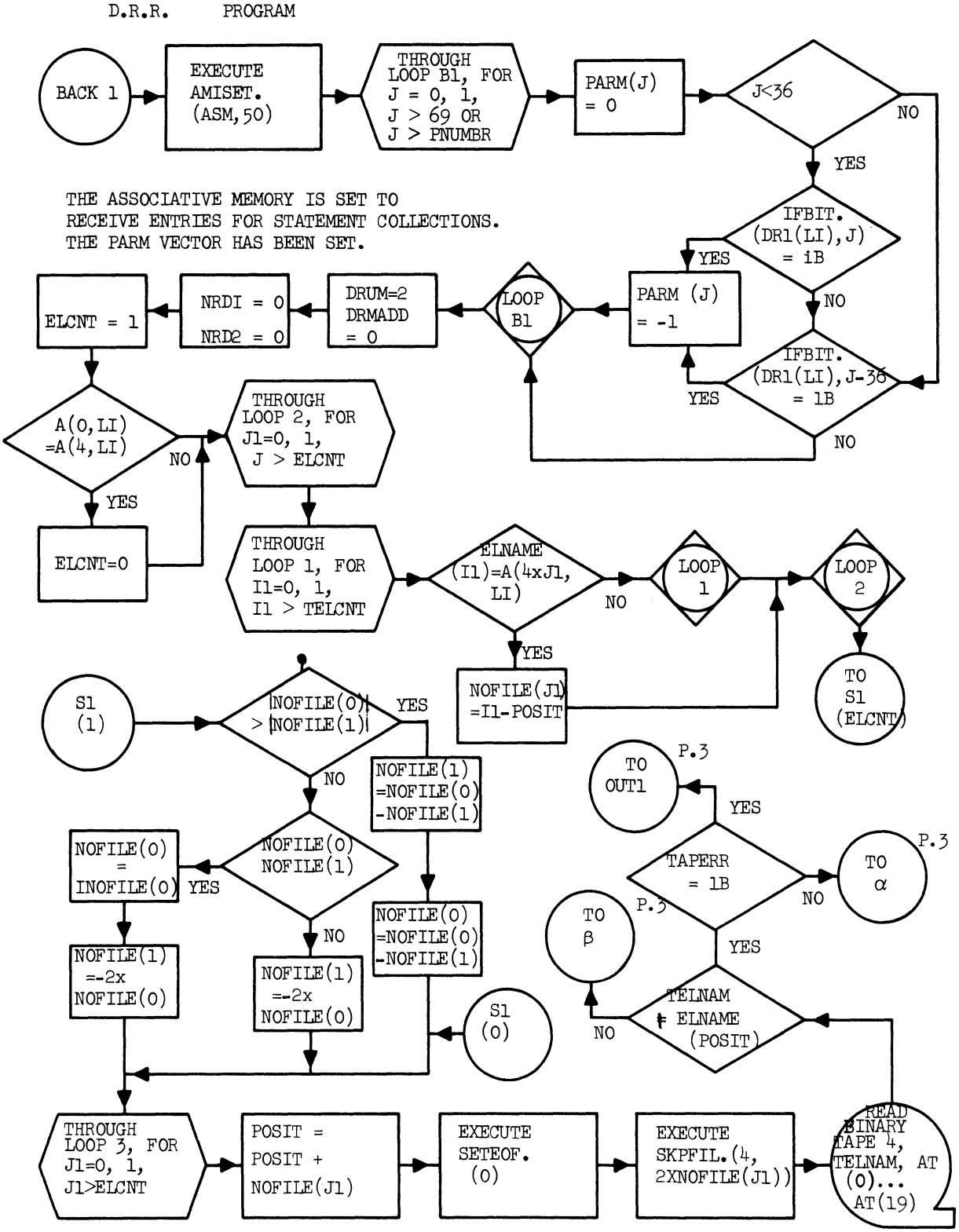


ELTPPR, ELEMENT DESCRIPTION TAPE PRINTER

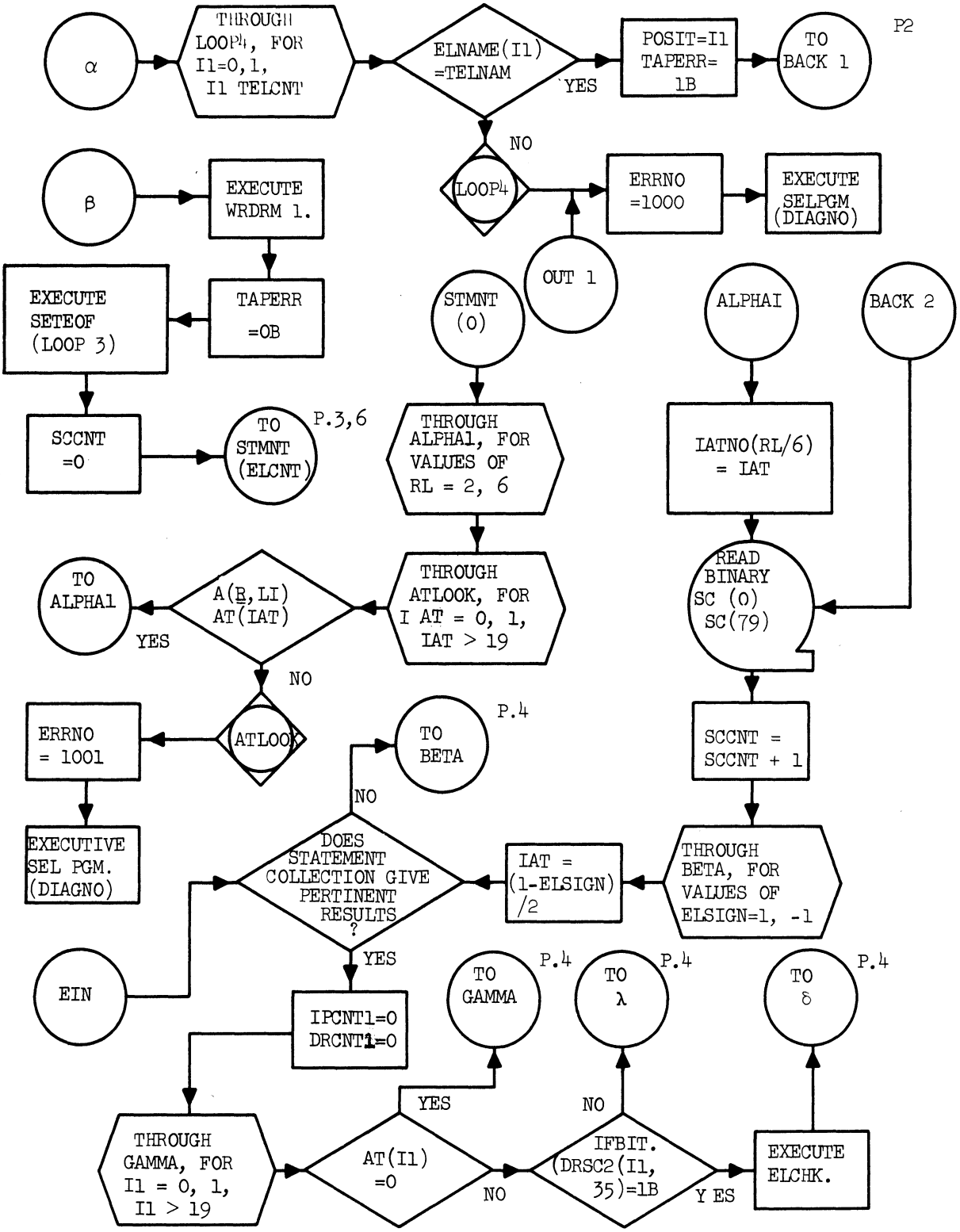


D.R.R. PROGRAM

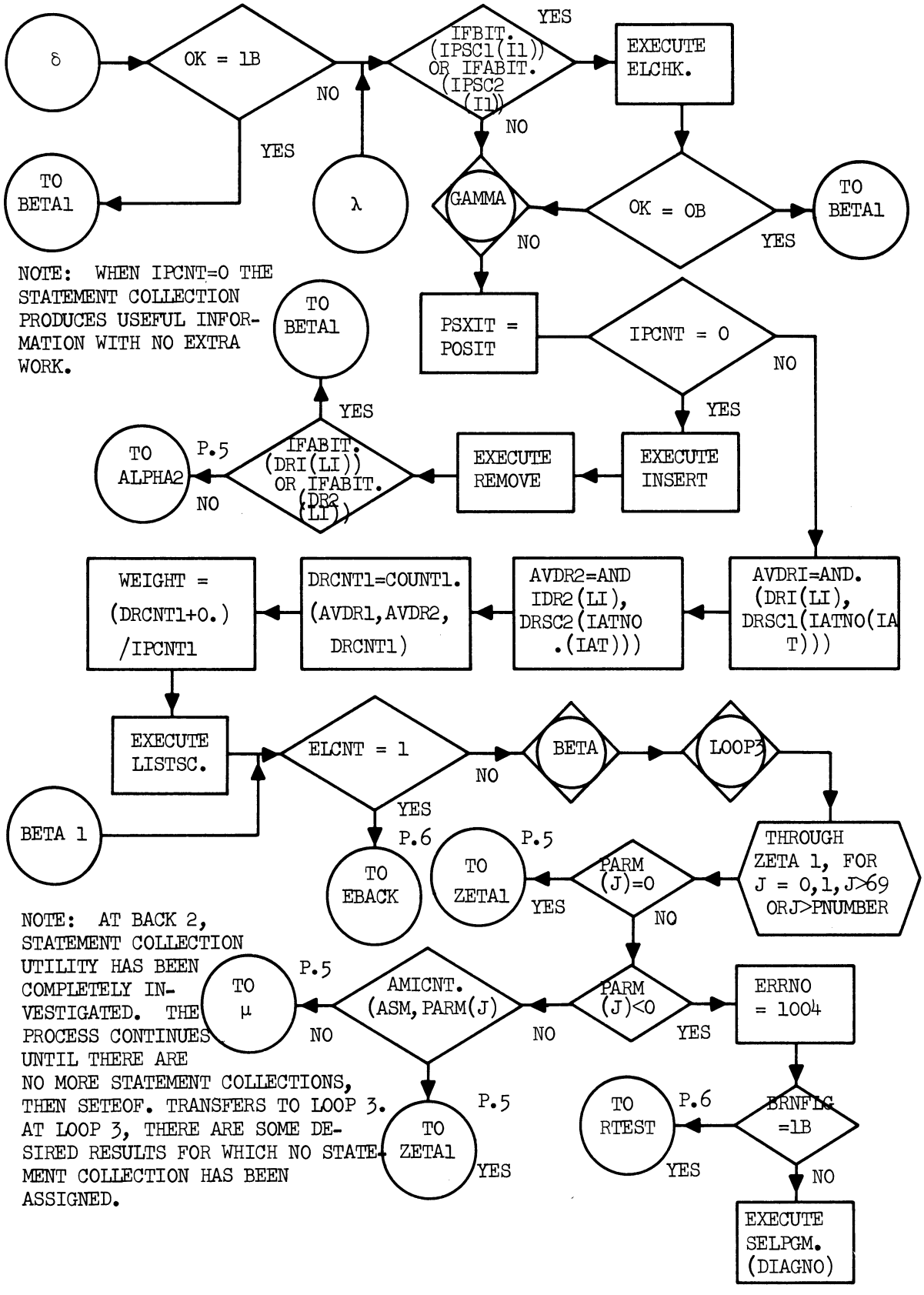

START
CORE NUMBERS PRESET
VECTOR VALUES DIAGNO = 8
VECTOR VALUES PROGEN = 5
VECTOR VALUES START = 1
VECTOR VALUES NOTRYS = 10
NRMST
TRYCNT = 0
REWIND TAPE 4
BRNFLG = OB
STCNT = 0
TO BEGIN
BRANCH
EXECUTE SKPFIL. (4,2)
POSIT = 0
TELCNT = B(0) + B(1) - 1
EXECUTE SKPFIL. (3,0)
BRNST
START
CONNECTION MATRIX IS RESTORED.
READ DRUM 4,0, A(2000) ... A(2399)
BRNFLB = OB
STCNT = 0
ERRNO = 0
RPTFLG = OB
THROUGH ALPHA, FOR I = 0, 1, I>CCNT OR I>199
P.5
BEGIN
NOT IFABIT. (DRI(LI)) OR IFABIT. (DR2(LI))
YES
LI = L(I)
SAVES PARENTHESES
RESULTS REQ'D FOR L(I)
NO
TO ALPHA
P.5
ANDBIT. (DRI(LI), IP1(LI)) OR ANDBIT. (DR2(LI), IP2(LI))
TAPERR = OB
DRI(LI) = AND. (NOT.(IP1(LI)), DR1(LI); DR2(LI)=AND. (NOT.(IP2(LI)), DR2(LI)
AT THIS POINT, ALL DESIRED RESULTS THAT CAN BE SATISFIED WITHOUT COMPUTATION HAVE BEEN REMOVED.
EXECUTE EXTCHK.
EXECUTE AMISET. (ASM,50)
TO BACK1
P.2
NOT (IFABIT.DRI (LI)) OR IFABIT (DR2(LI)))
YES
NO


D.R.R. PROGRAM
BACK 1
EXECUTE AMISET. (ASM,50)
THROUGH LOOP B1, FOR J = 0, 1, J > 69 OR J > PNUMBR
PARM(J) = 0
J<36
NO
YES
THE ASSOCIATIVE MEMORY IS SET TO RECEIVE ENTRIES FOR STATEMENT COLLECTIONS. THE PARM VECTOR HAS BEEN SET.
IFBIT. (DR1(LI),J) = iB
YES
NO
PARM (J) = -1
LOOP B1
DRUM=2 DRMADD = 0
NRDI = 0 NRD2 = 0
ELCNT = 1
IFBIT. (DR1(LI),J-36) = 1B
YES
NO
A(0,LI) =A(4,LI)
NO
YES
THROUGH LOOP 2, FOR J1=0, 1, J > ELCNT
ELCNT=0
THROUGH LOOP 1, FOR I1=0, 1, I1 > TELCNT
ELNAME (I1)=A(4xJ1, LI)
NO
YES
LOOP 1
LOOP 2
NOFILE(J1) =I1-POSIT
TO S1 (ELCNT)
S1 (1)
|NOFILE(0)| > |NOFILE(1)|
YES
NO
NOFILE(1) =NOFILE(0) -NOFILE(1)
NOFILE(0) NOFILE(1)
NOFILE(0) = INOFILE(0)
YES
NO
NOFILE(1) =-2x NOFILE(0)
NOFILE(1) =-2x NOFILE(0)
NOFILE(0) =NOFILE(0) -NOFILE(1)
S1 (0)
TO OUT1
P.3
YES
TAPERR = 1B
NO
TO α
P.3
TO β
P.3
YES
NO
TELNAM ≠ ELNAME (POSIT)
THROUGH LOOP 3, FOR J1=0, 1, J1>ELCNT
POSIT = POSIT + NOFILE(J1)
EXECUTE SETEOF. (0)
EXECUTE SKPFIL.(4, 2XNOFILE(J1))
READ BINARY TAPE 4, TELNAM, AT (0)... AT(19)

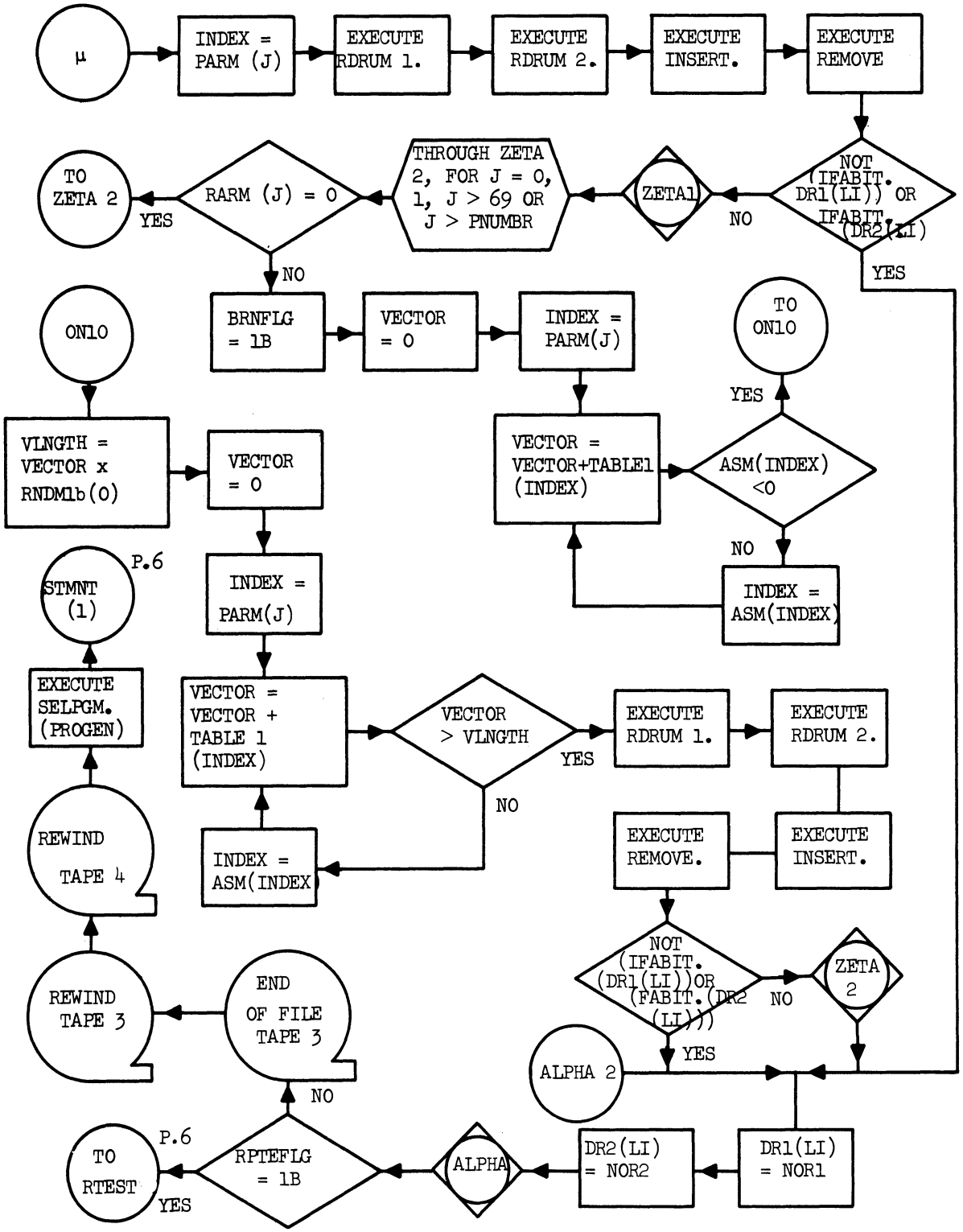
D.R.R. PROGRAM
Page 3



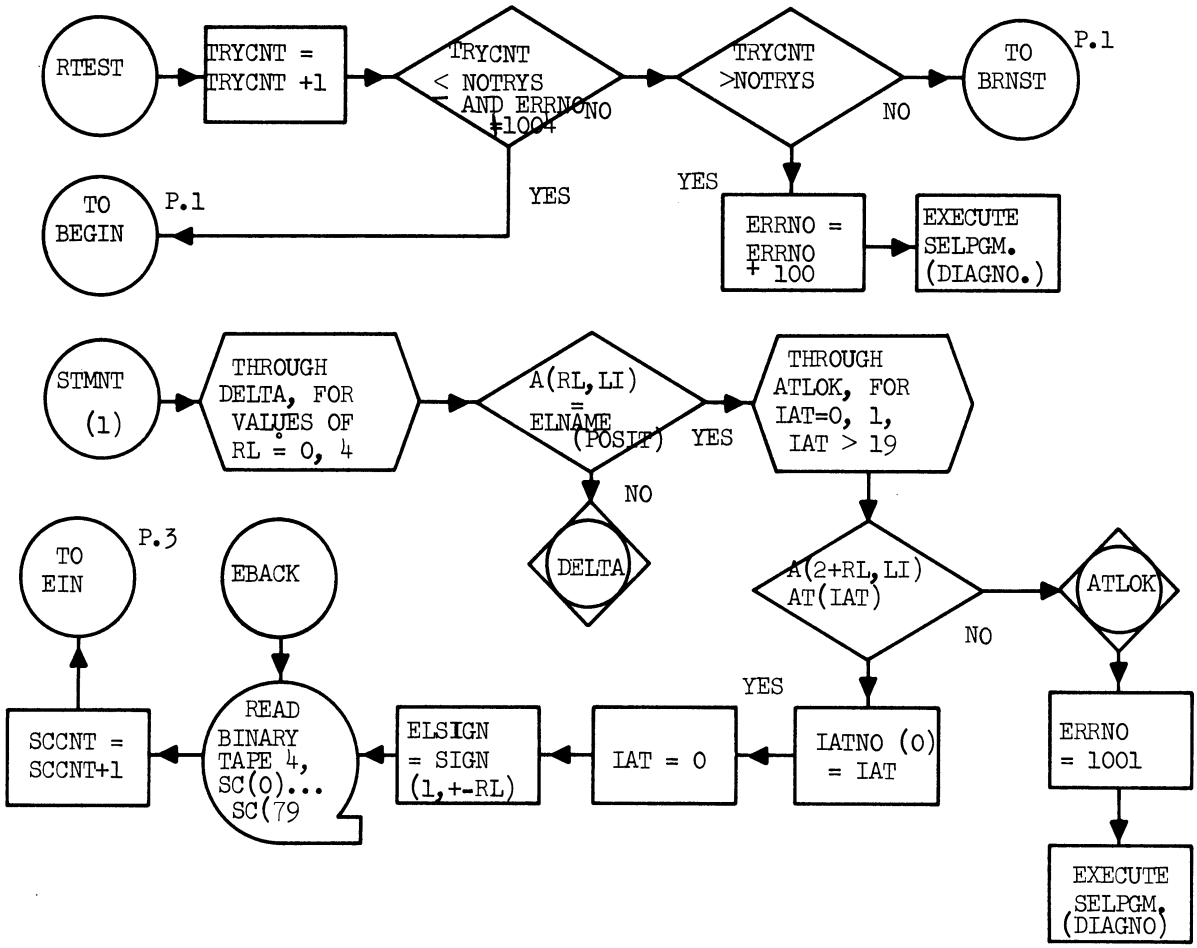
D.R.R. PROGRAM


δ
OK = 1B
NO
YES
TO BETA1
λ
IFBIT. (IPSC1(I1)) OR IFABIT. (IPSC2 (I1))
YES
NO
EXECUTE ELCHK.
GAMMA
NO
OK = 0B
YES
TO BETA1
PSXIT = POSIT
IPCNT = 0
NO
YES
NOTE: WHEN IPCNT=0 THE STATEMENT COLLECTION PRODUCES USEFUL INFORMATION WITH NO EXTRA WORK.
TO BETA1
YES
TO ALPHA2
P.5
NO
IFABIT. (DRI(LI)) OR IFABIT. (DR2 (LI))
EXECUTE REMOVE
EXECUTE INSERT
AVDRI=AND. (DRI(LI), DRSC1(IATNO(IAT)))
AVDR2=AND IDR2(LI), DRSC2(IATNO .(IAT)))
DRCNT1=COUNT1. (AVDR1,AVDR2, DRCNT1)
WEIGHT = (DRCNT1+0.) /IPCNT1
EXECUTE LISTSC.
ELCNT = 1
NO
BETA
LOOP3
YES
P.6
TO EBACK
P.5
TO ZETA1
YES
PARM (J)=0
NO
THROUGH ZETA 1, FOR J = 0,1,J>69 ORJ>PNUMBER
BETA 1
NOTE: AT BACK 2, STATEMENT COLLECTION UTILITY HAS BEEN COMPLETELY INVESTIGATED. THE PROCESS CONTINUES UNTIL THERE ARE NO MORE STATEMENT COLLECTIONS, THEN SETEOF. TRANSFERS TO LOOP 3. AT LOOP 3, THERE ARE SOME DESIRED RESULTS FOR WHICH NO STATEMENT COLLECTION HAS BEEN ASSIGNED.
TO μ
P.5
NO
AMICNT. (ASM, PARM(J)
NO
PARM (J)<0
YES
ERRNO = 1004
P.5
TO ZETA1
YES
TO RTEST
P.6
YES
BRNFLG =1B
NO
EXECUTE SELPGM. (DIAGNO)

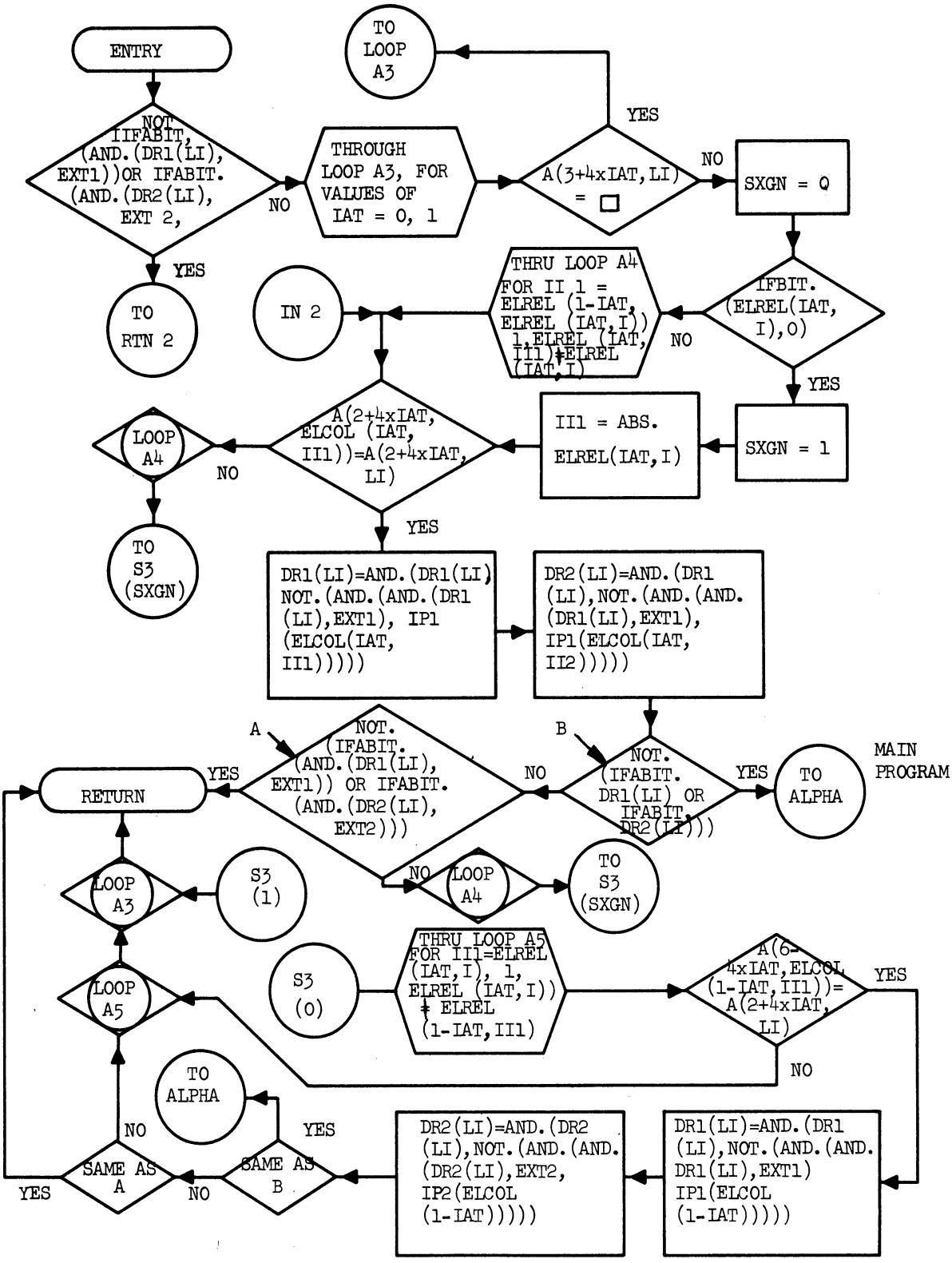
D.R.R. PROGRAM


μ
INDEX = PARM (J)
EXECUTE RDRUM 1.
EXECUTE RDRUM 2.
EXECUTE INSERT.
EXECUTE REMOVE
NOT (IFABIT. DR1(LI)) OR IFABIT. (DR2(LI)
YES
NO
ZETA1
THROUGH ZETA 2, FOR J = 0, 1, J > 69 OR J > PNUMBR
RARM (J) = 0
YES
TO ZETA 2
NO
BRNFLG = 1B
VECTOR = 0
INDEX = PARM(J)
ON10
VLNGTH = VECTOR x RNDM1b(0)
VECTOR = 0
INDEX = PARM(J)
VECTOR = VECTOR + TABLE 1 (INDEX)
VECTOR > VLNGTH
YES
NO
INDEX = ASM(INDEX)
EXECUTE RDRUM 1.
EXECUTE RDRUM 2.
EXECUTE INSERT.
EXECUTE REMOVE.
NOT (IFABIT. (DR1(LI))OR (FABIT.(DR2 (LI)))
NO
ZETA 2
YES
ALPHA 2
DR1(LI) = NOR1
DR2(LI) = NOR2
ALPHA
RPTEFLG = 1B
YES
TO RTEST
P.6
NO
END OF FILE TAPE 3
REWIND TAPE 3
REWIND TAPE 4
EXECUTE SELPGM. (PROGEN)
STMNT (1)
P.6
VECTOR = VECTOR+TABLE1 (INDEX)
ASM(INDEX) <0
YES
NO
INDEX = ASM(INDEX)
TO ON10

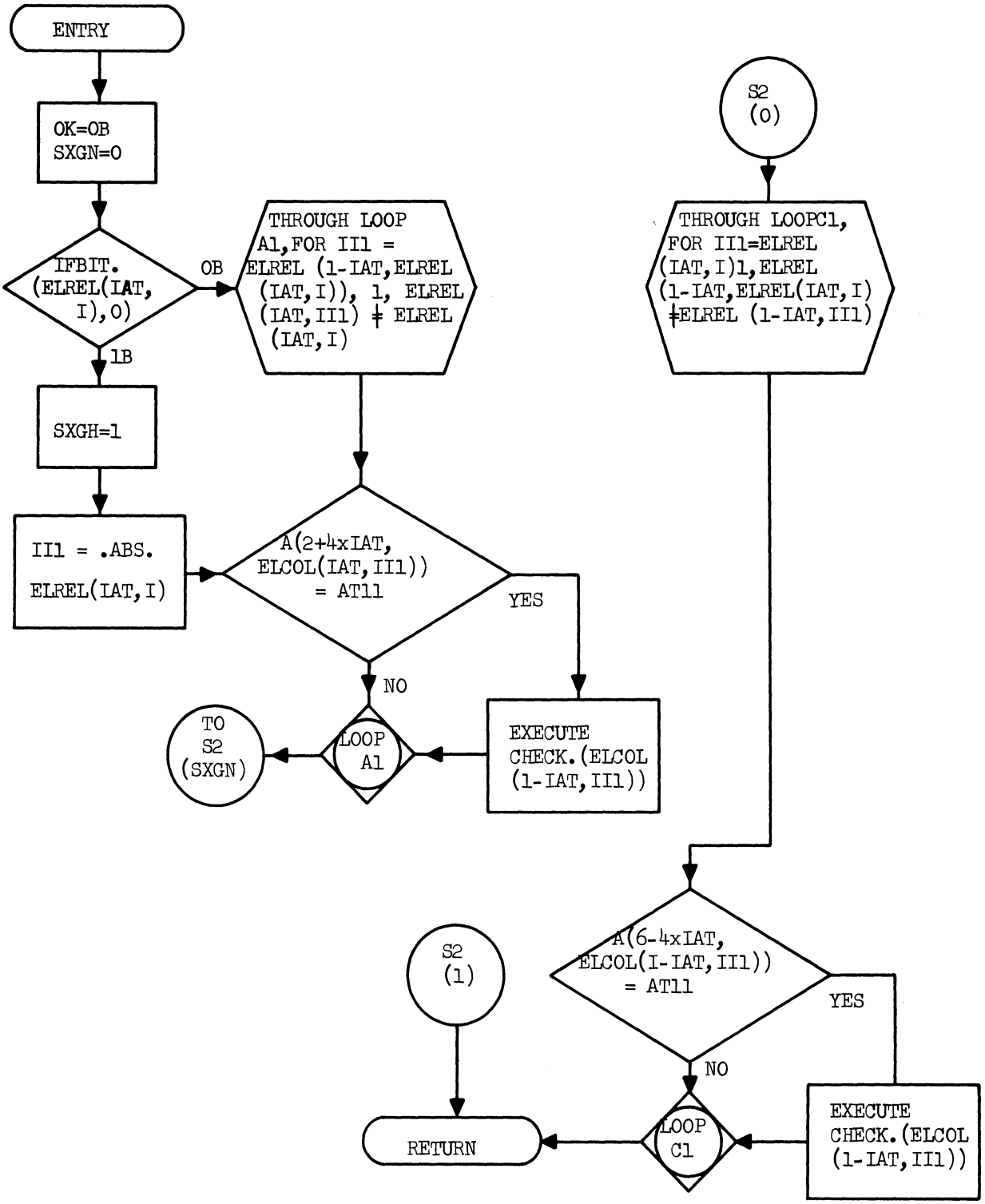
D.R.R. PROGRAM


RTEST
TRYCNT = TRYCNT +1
TRYCNT < NOTRYS AND ERRNO ≠1004
NO
YES
TRYCNT >NOTRYS
NO
YES
TO BRNST
P.1
TO BEGIN
P.1
ERRNO = ERRNO + 100
EXECUTE SELPGM. (DIAGNO.)
STMNT (1)
THROUGH DELTA, FOR VALUES OF RL = 0, 4
A(RL,LI) = ELNAME (POSIT)
YES
NO
DELTA
THROUGH ATLOK, FOR IAT=0, 1, IAT > 19
A(2+RL,LI) AT(IAT)
NO
YES
ATLOK
IATNO (0) = IAT
IAT = 0
ELSIGN = SIGN (1,+-RL)
READ BINARY TAPE 4, SC(0)... SC(79
EBACK
SCCNT = SCCNT+1
TO EIN
P.3
ERRNO = 1001
EXECUTE SELPGM. (DIAGNO)

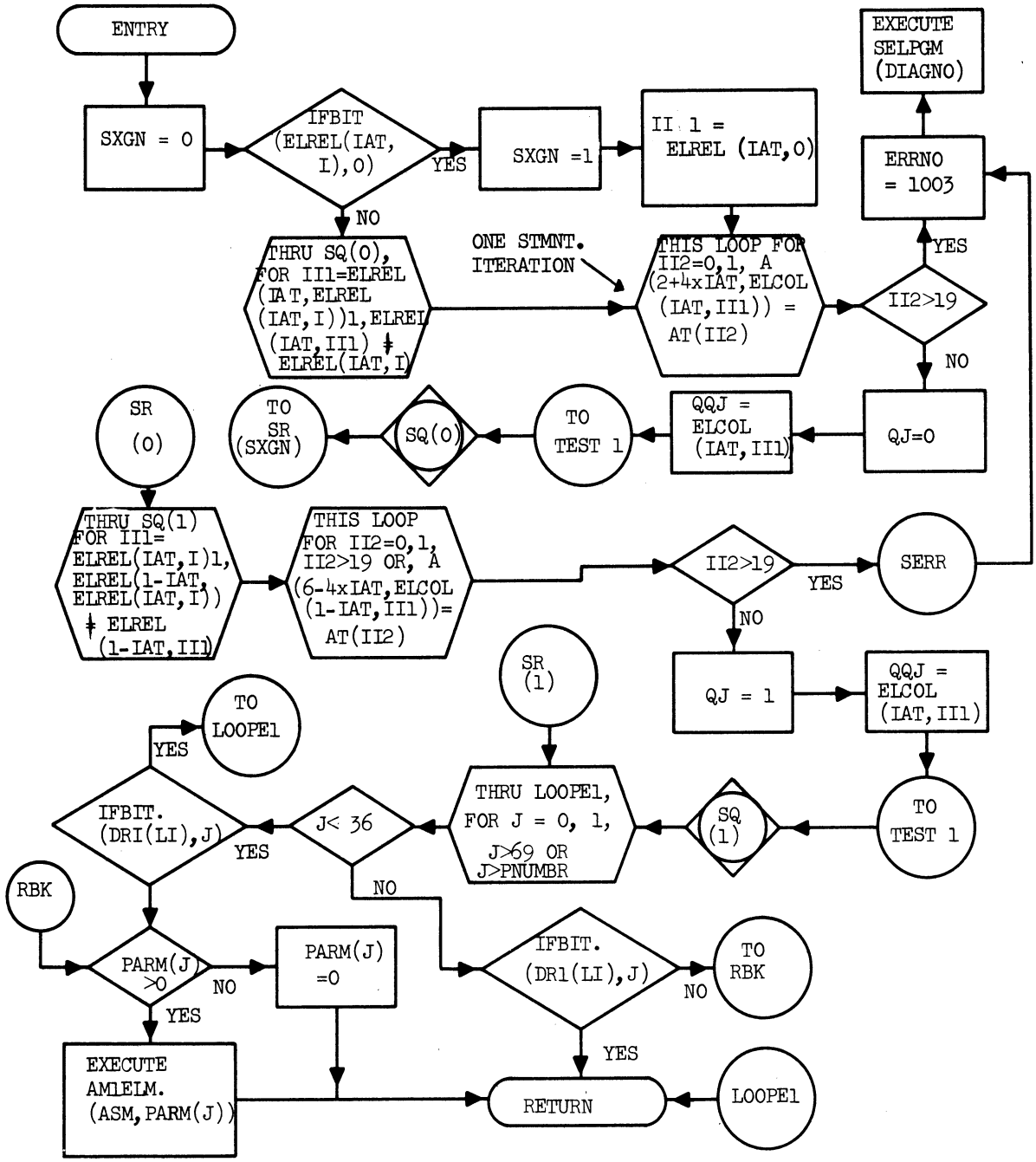
INTERNAL FUNCTION EXTCHK.



INTERNAL FUNCTION ELCHK.

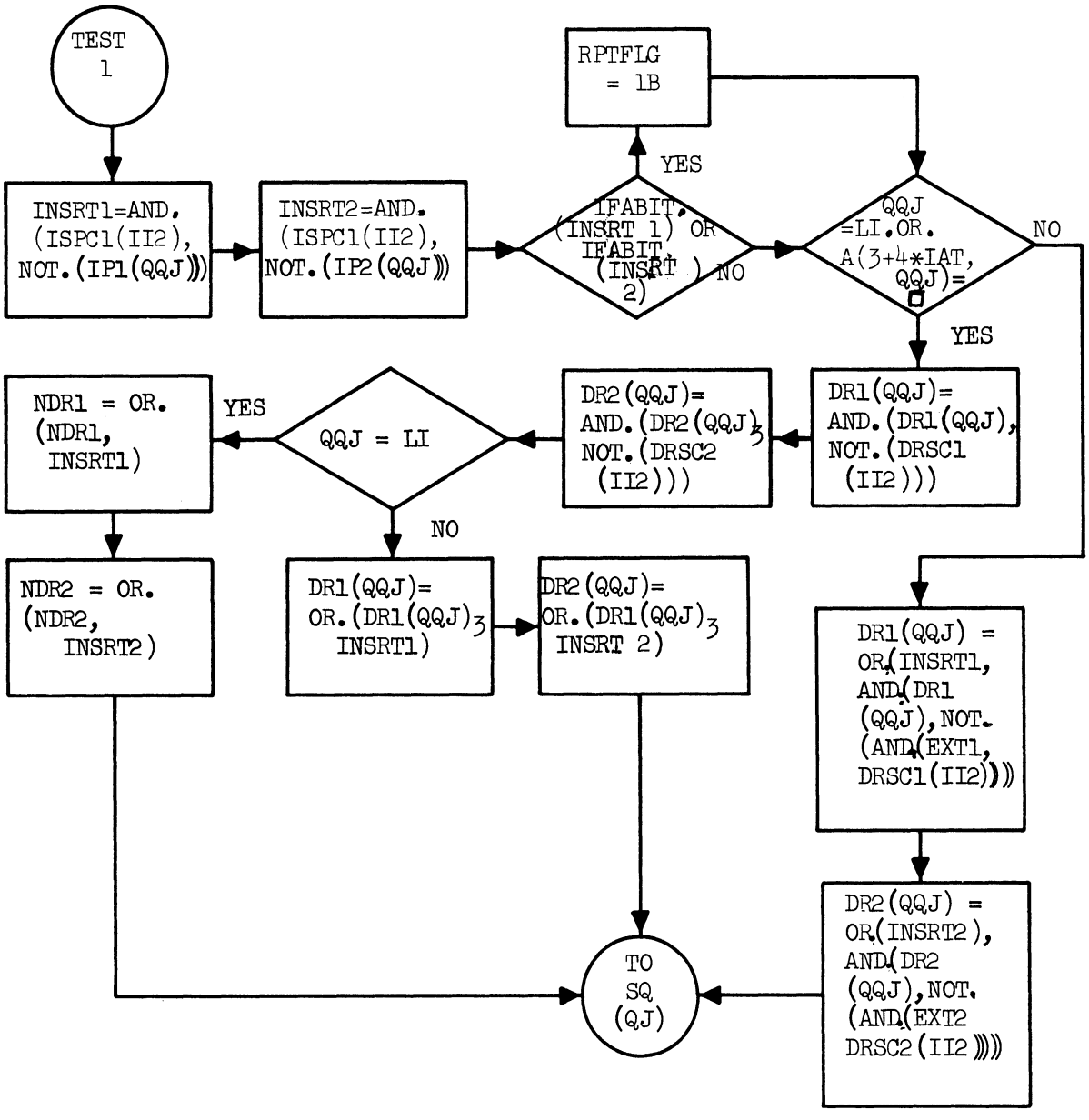


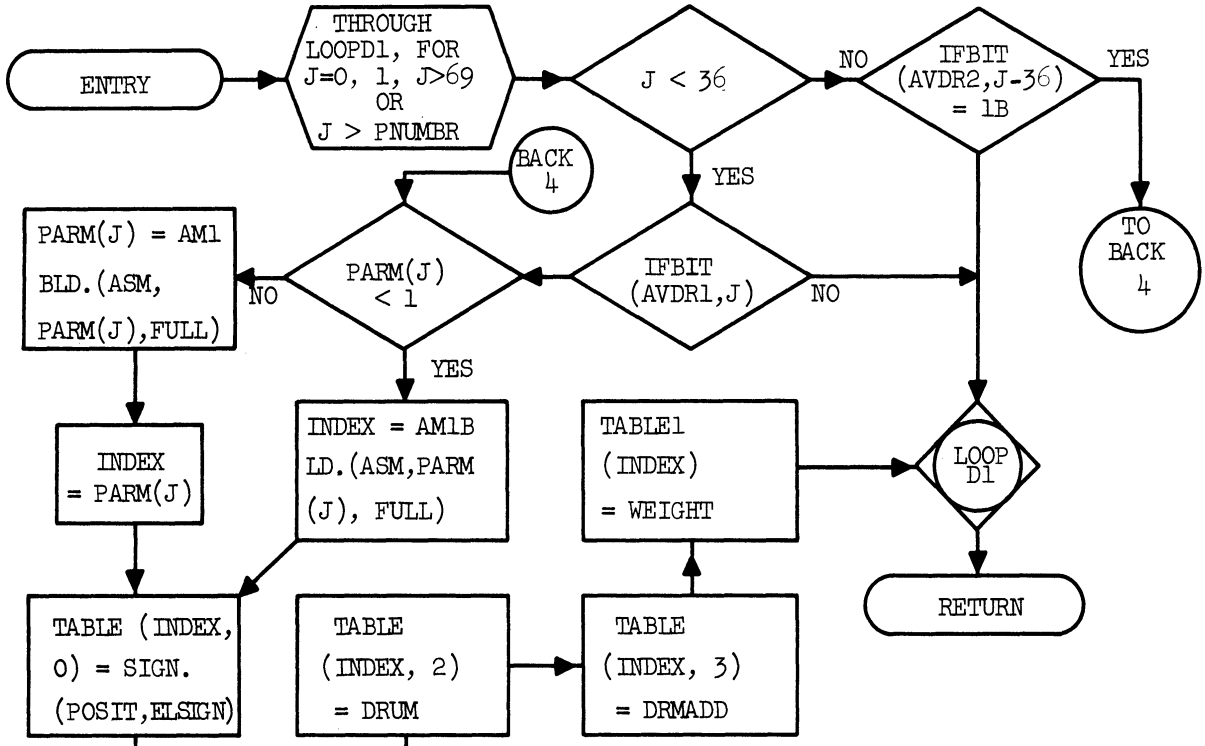
INTERNAL FUNCTION REMOVE
Page 1



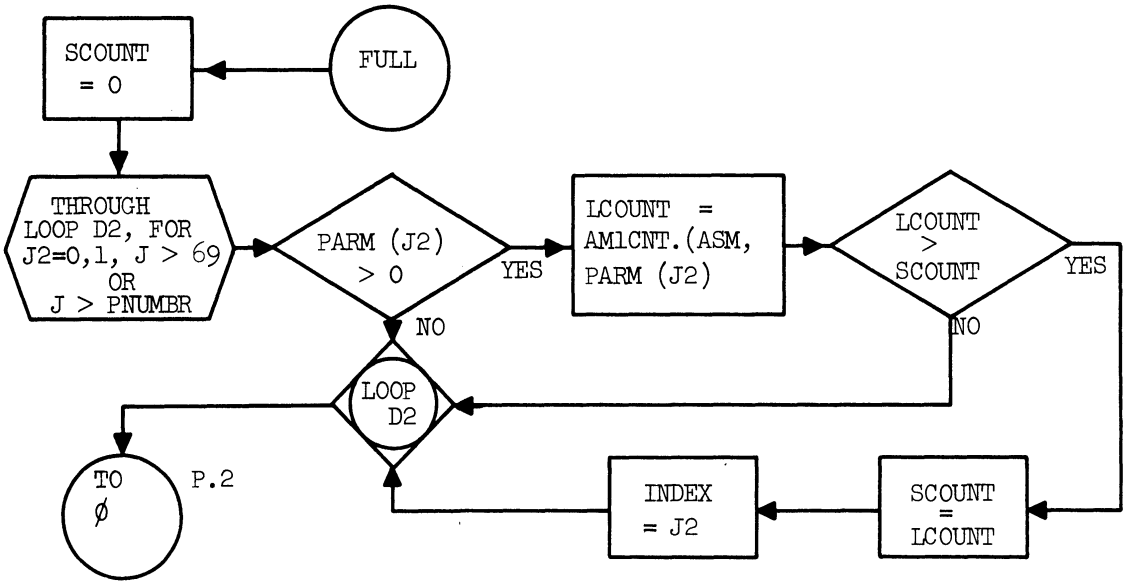
INTERNAL FUNCTION REMOVE

Page 2

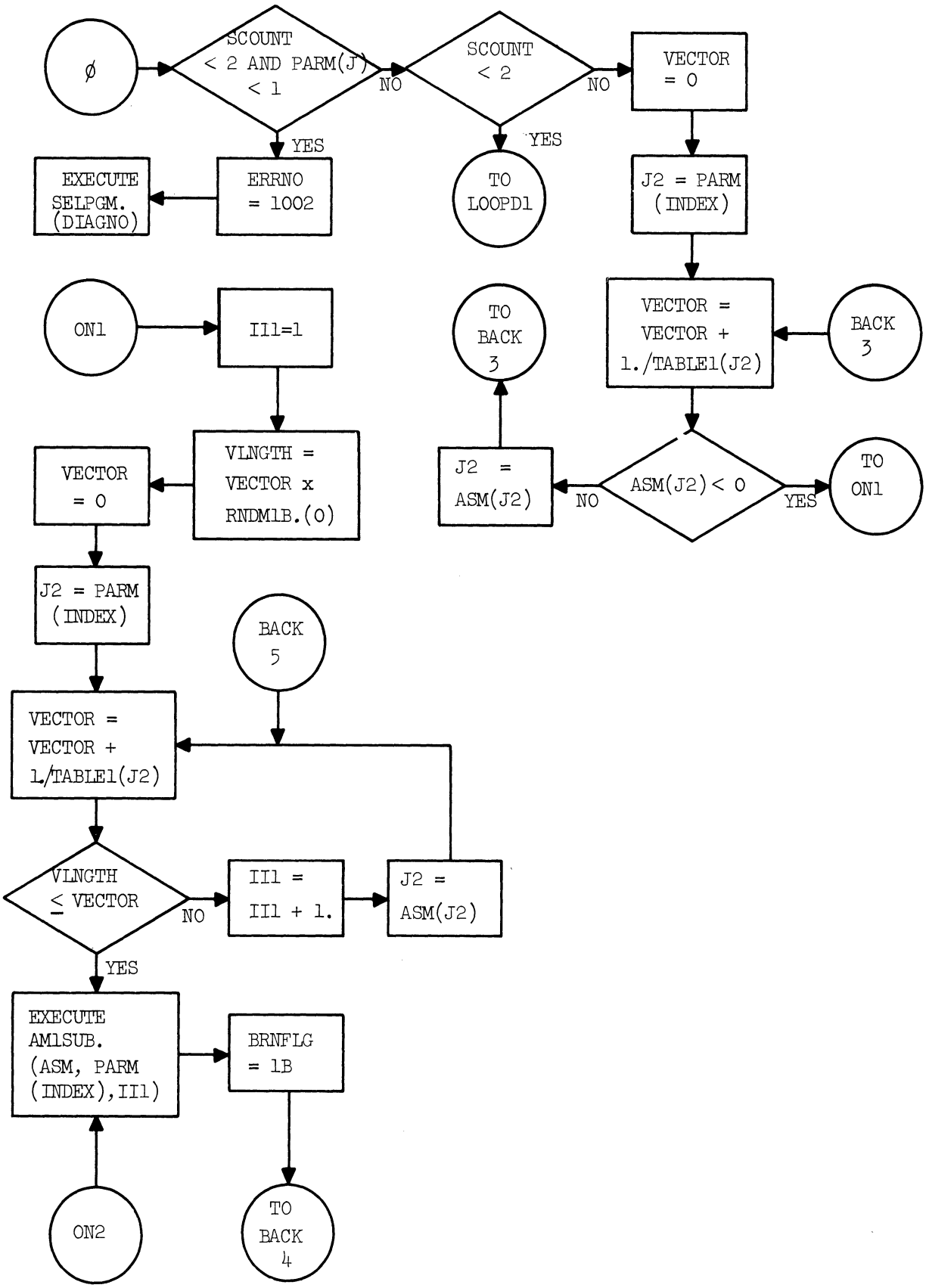




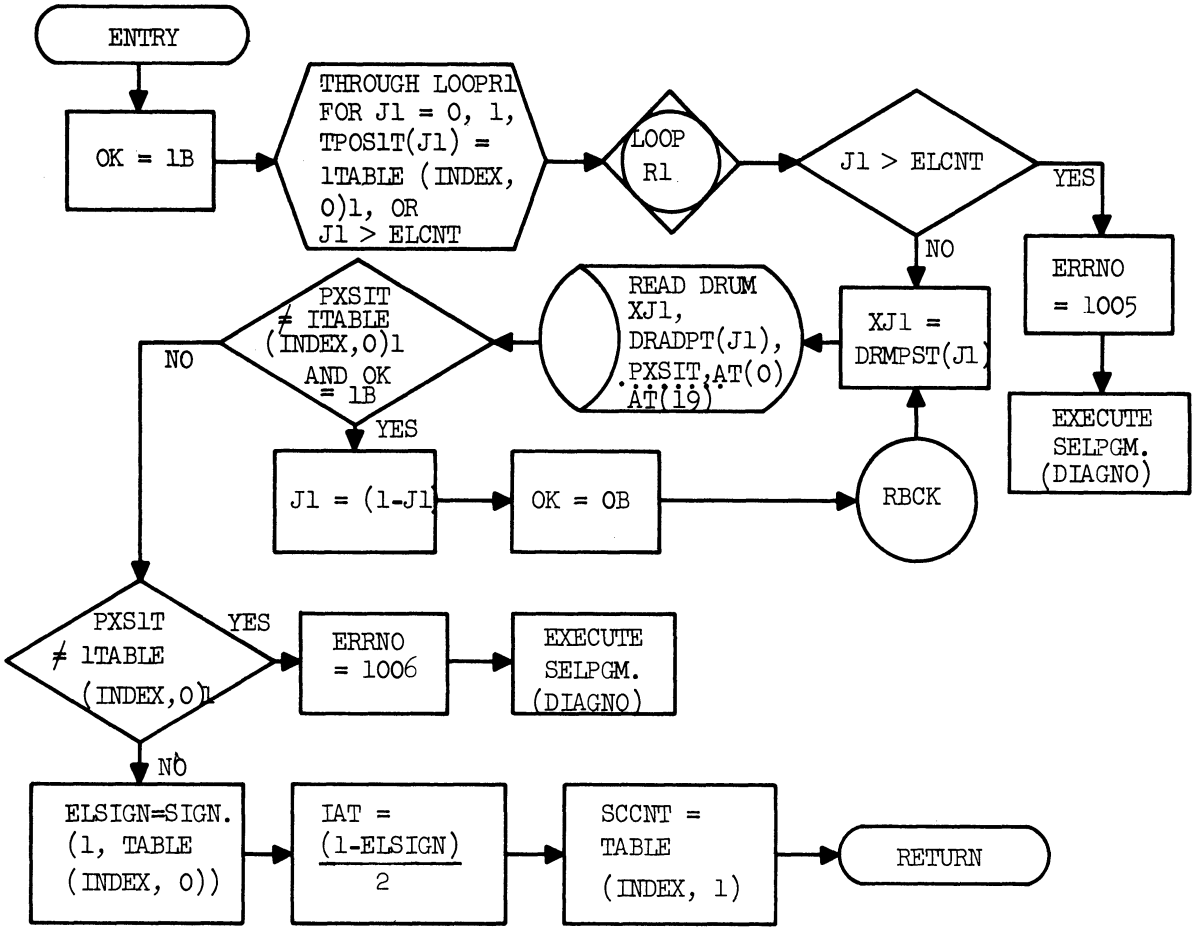
THE NEXT SECTION IS USED WHENEVER THE NUMBER OF STATEMENT COLLECTIONS EXCEEDS AVAILABLE STORAGE. THE PARAMETER WITH THE GREATEST NUMBER OF COLLECTIONS DISCARDS ONE COLLECTION; MOST PROBABLY THE ONE WITH THE LEAST WEIGHT.



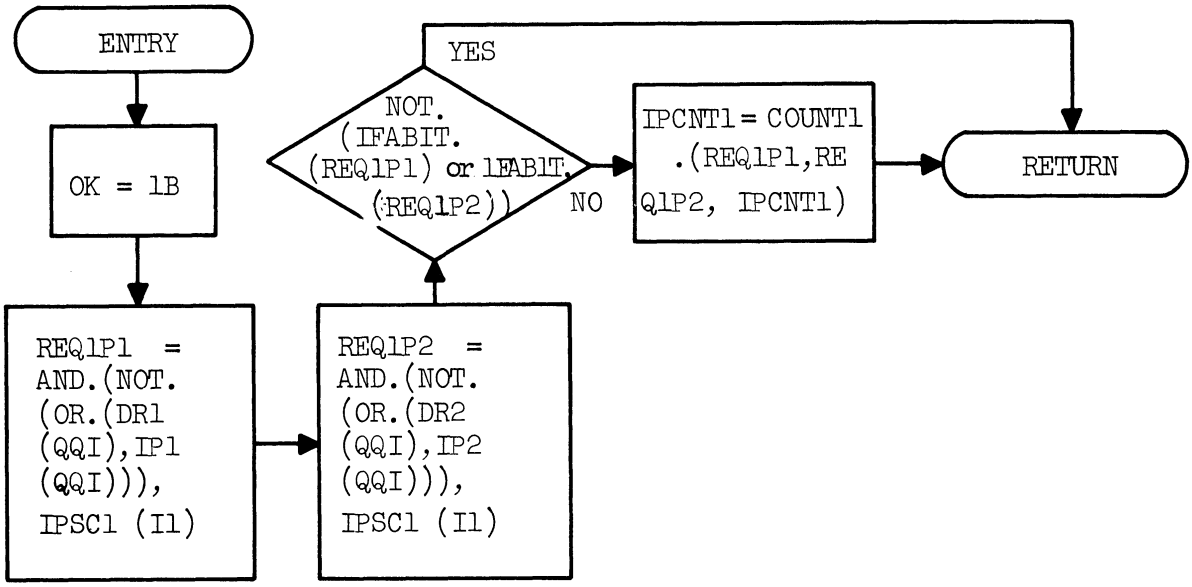
INTERNAL FUNCTION LISTSC.



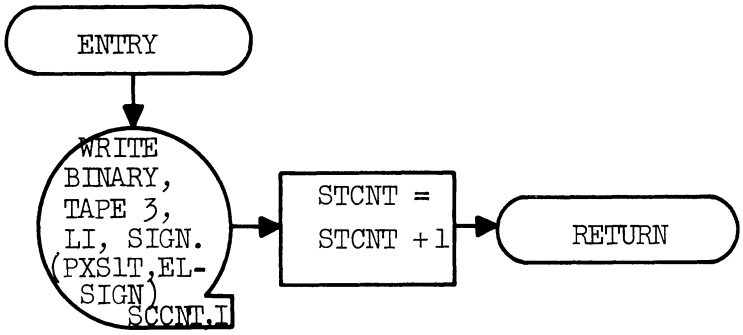
INTERNAL FUNCTION RDRUM1.



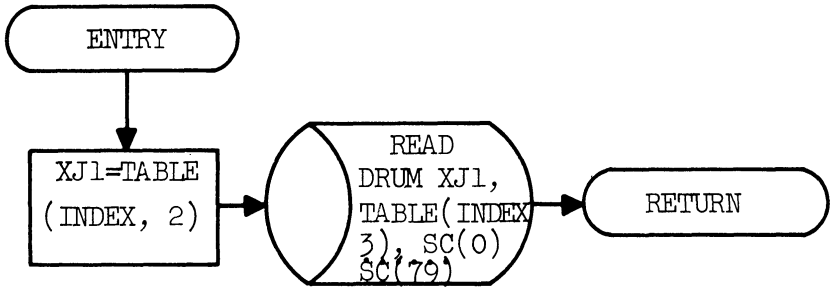
INTERNAL FUNCTION CHECK. (QQI)



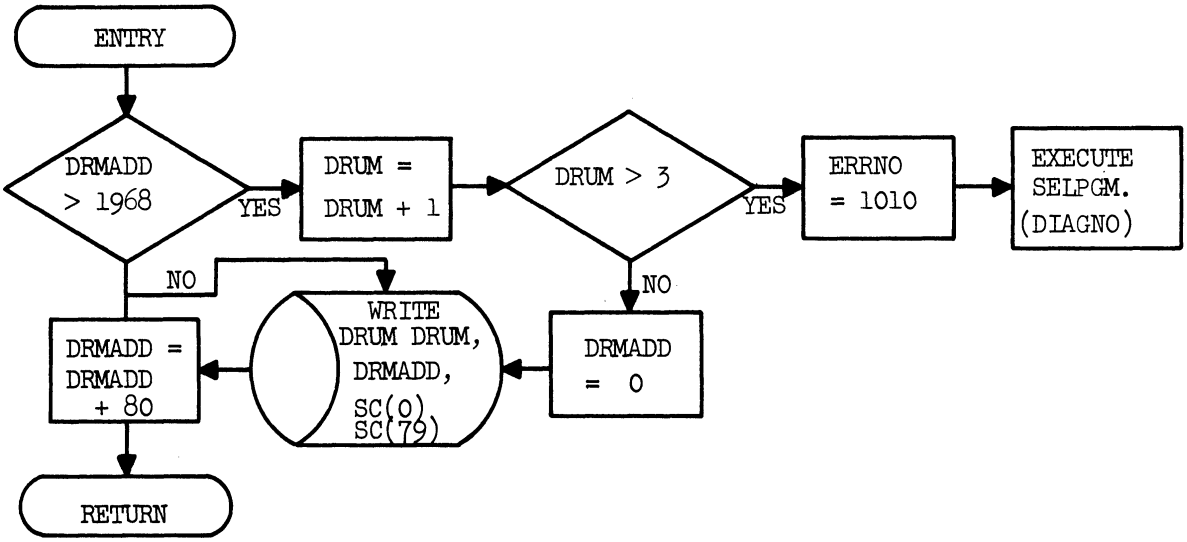
INTERNAL FUNCTION INSERT.



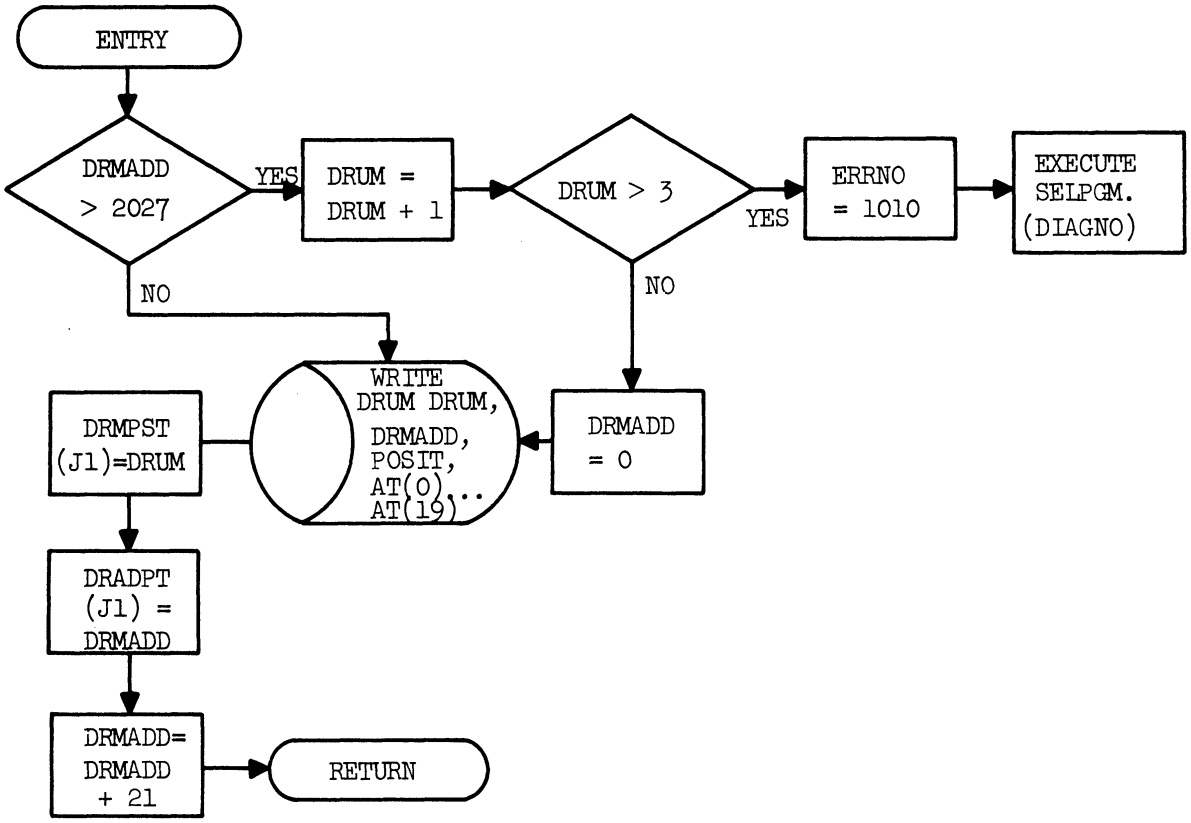
INTERNAL FUNCTION RDRUM2.



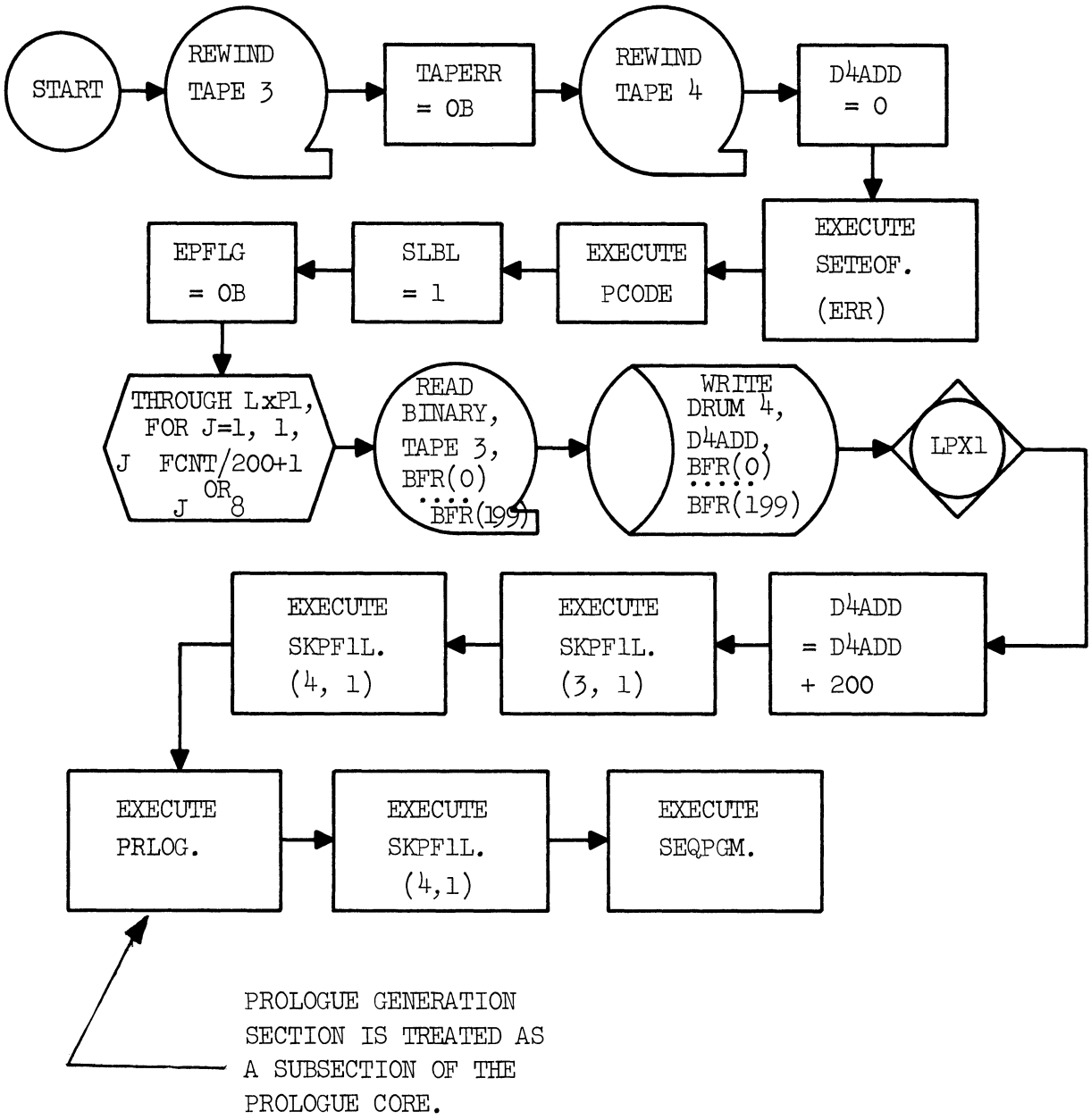
INTERNAL FUNCTION WRDRUM



INTERNAL FUNCTION WRDRUM1

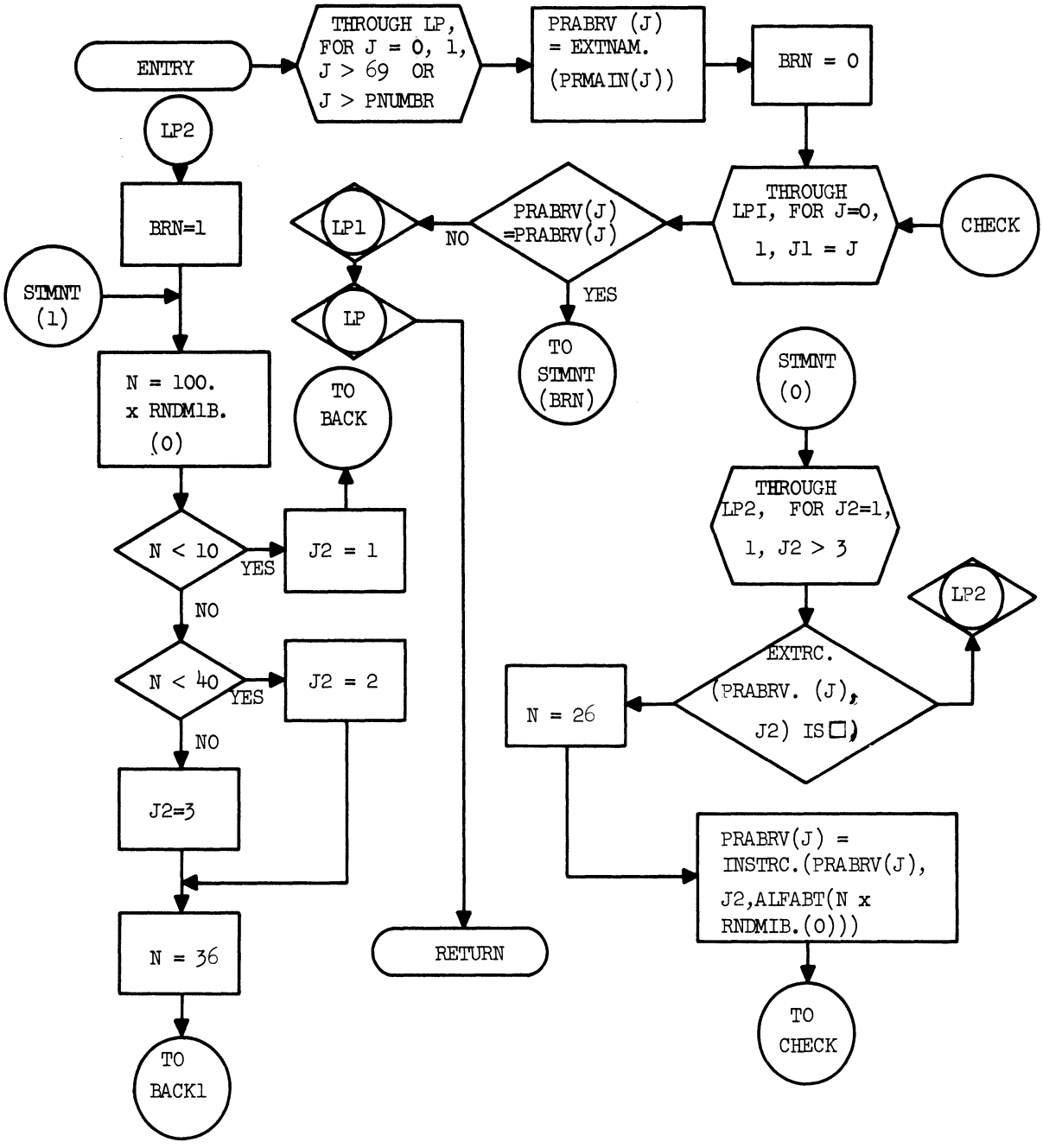


PROLOGUE - MAIN PROGRAM

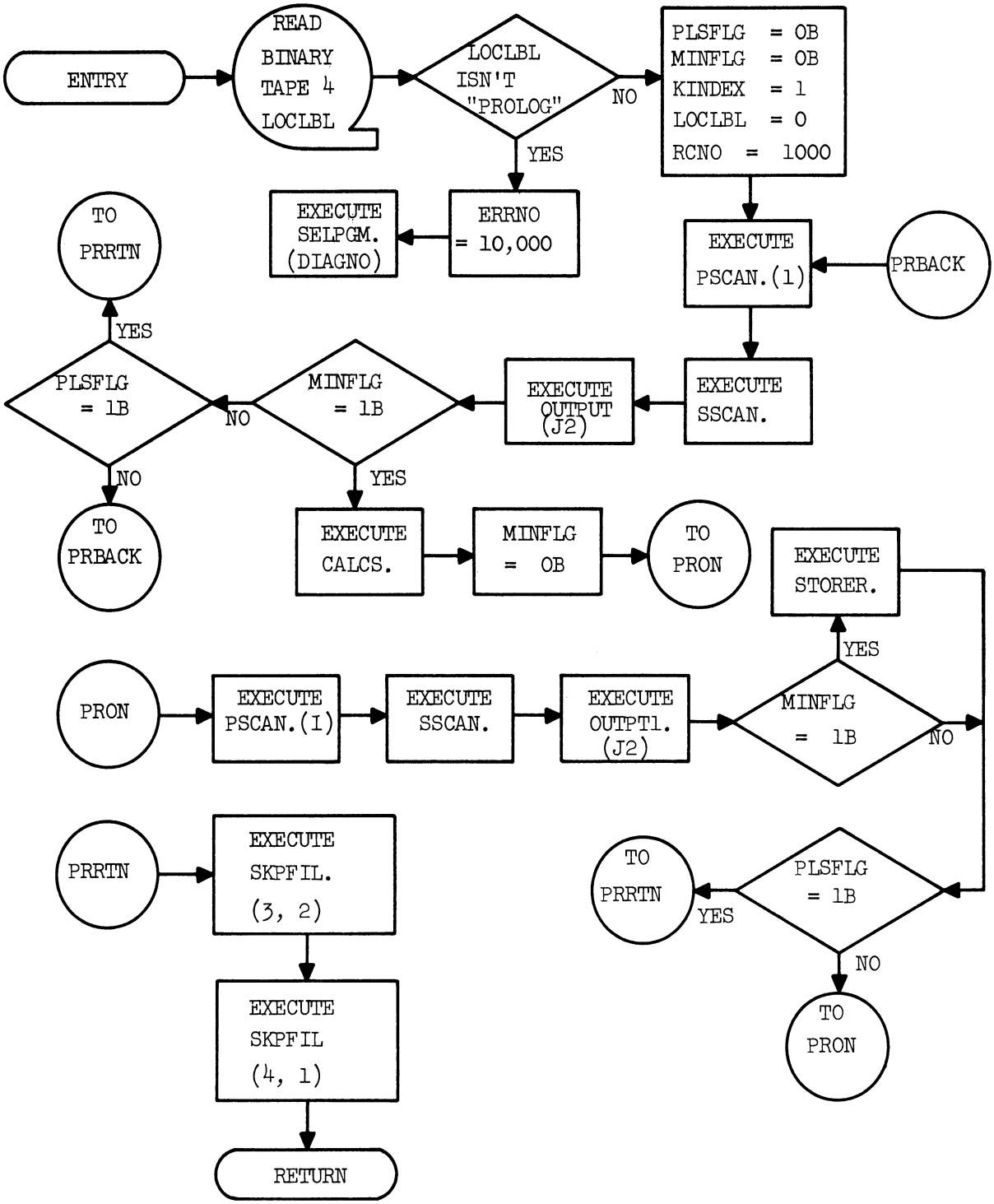


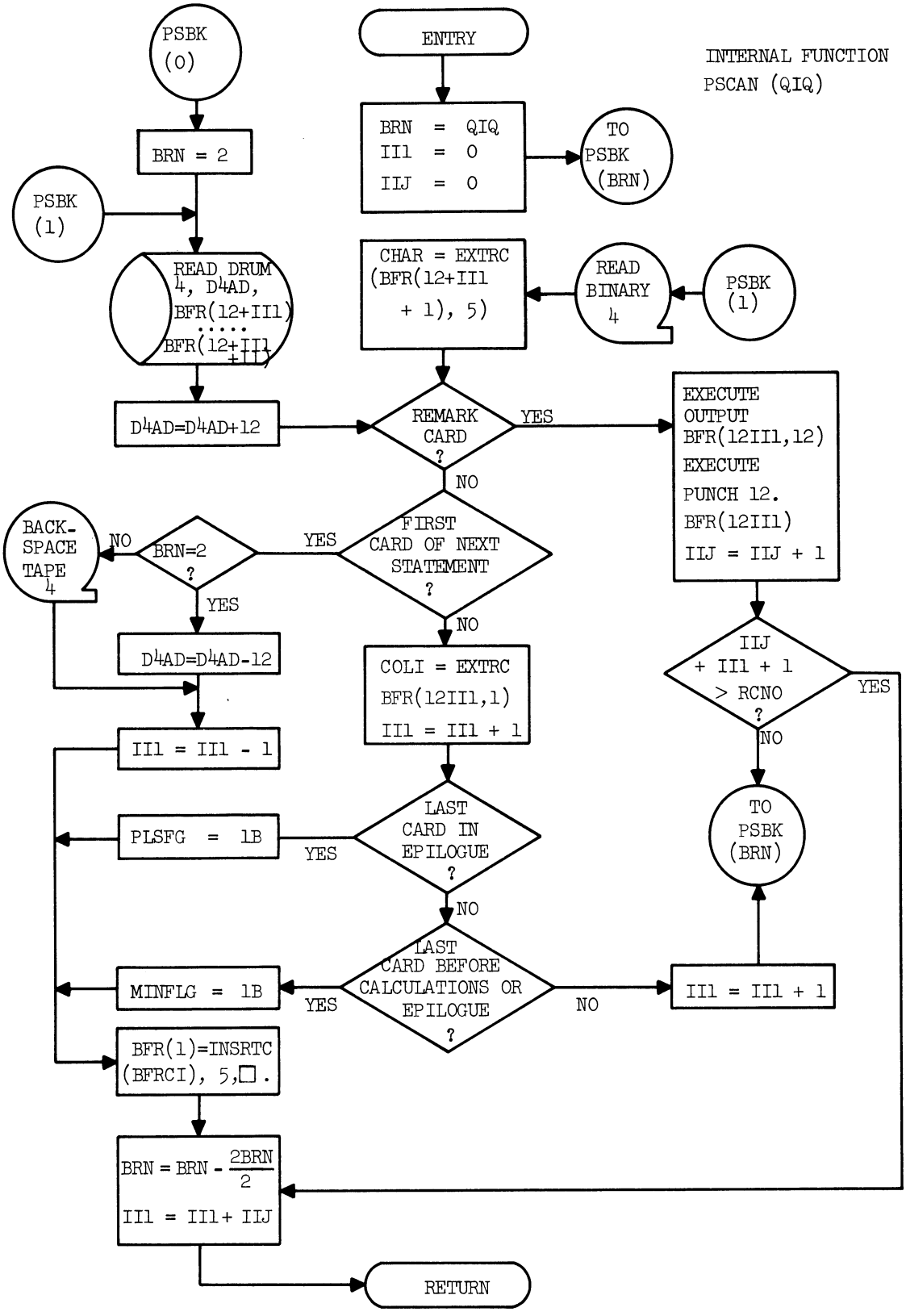
INTERNAL FUNCTION PCODE

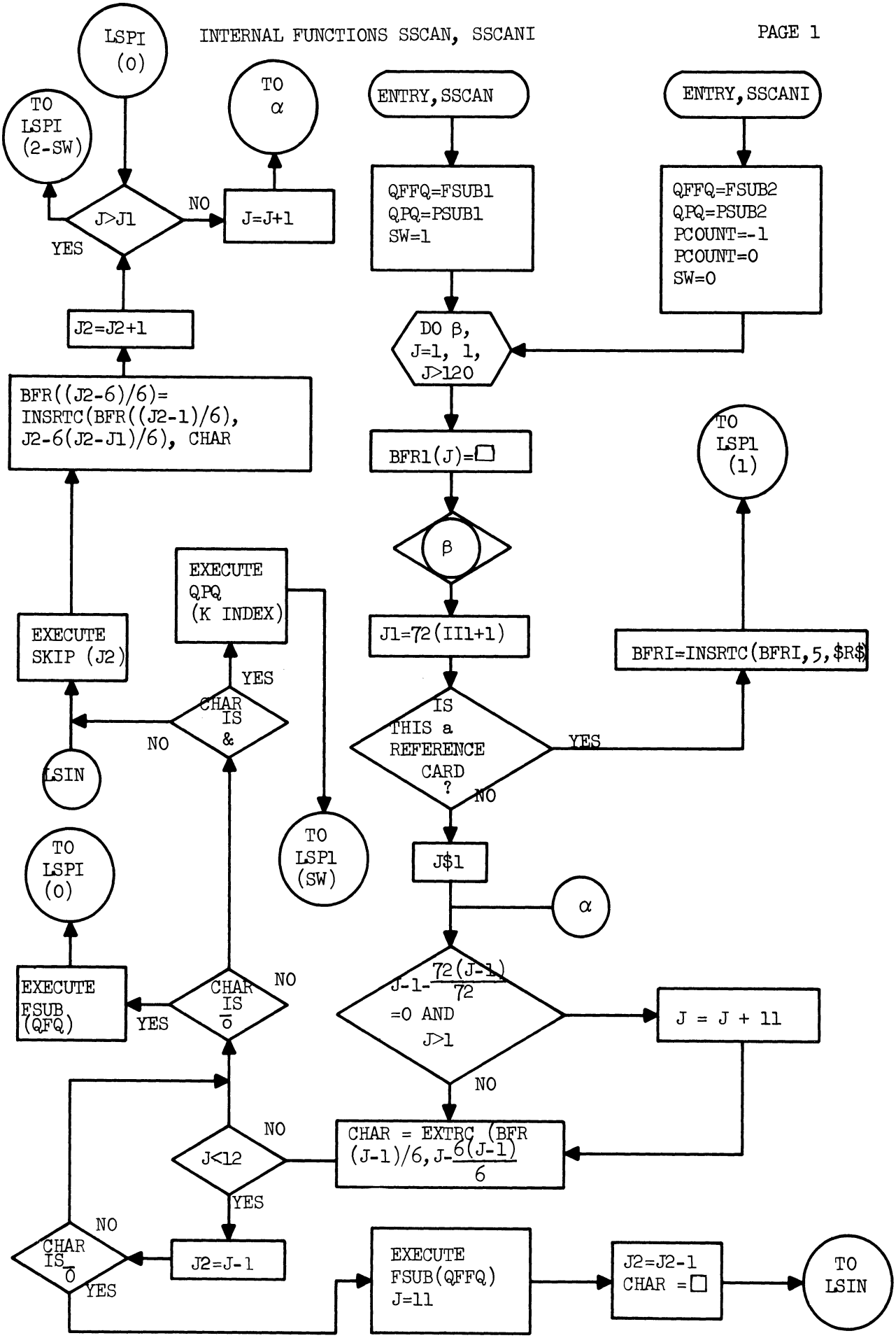
(CONSTRUCTS UNIQUE THREE CHARACTER CODES.)



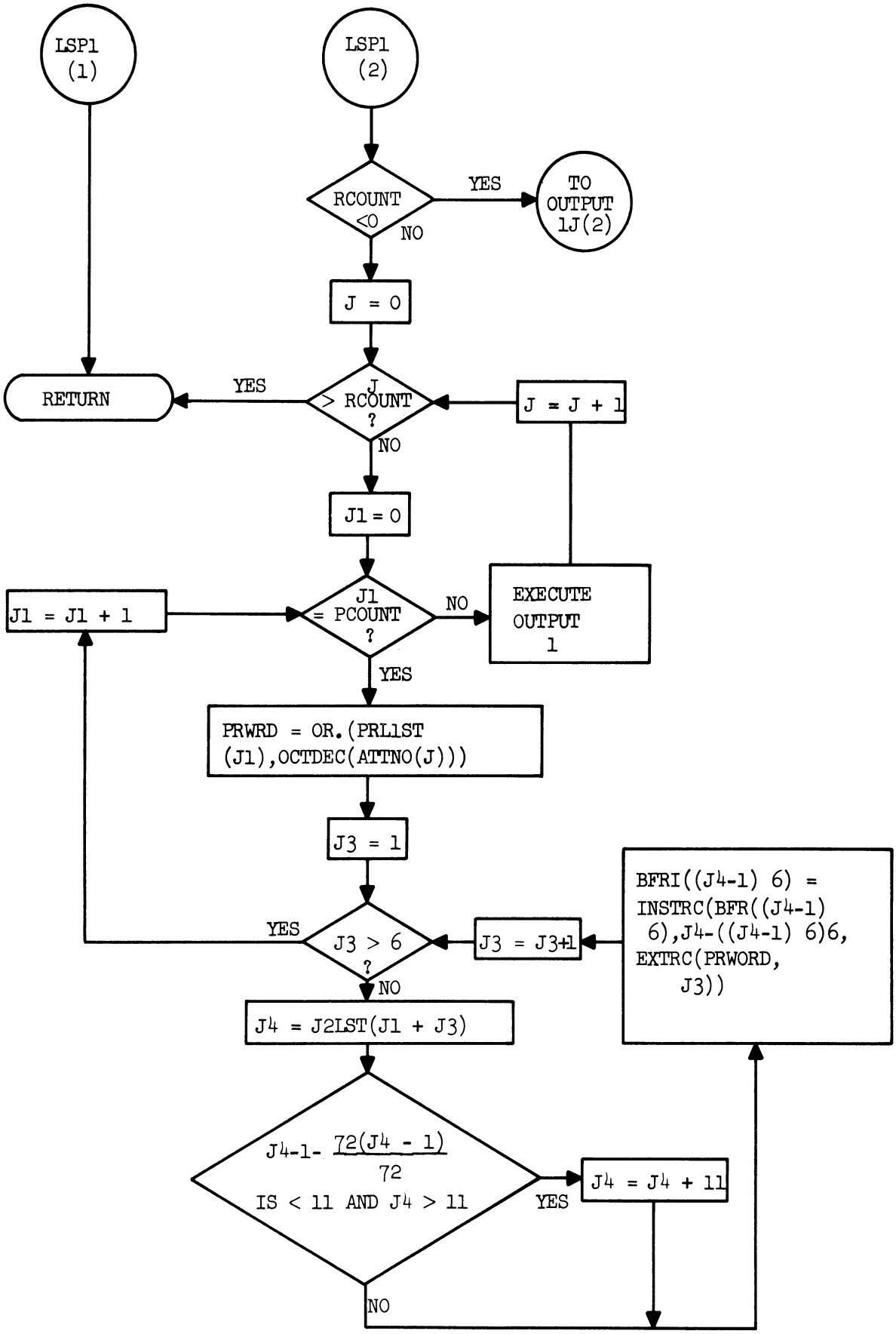
INTERNAL FUNCTION PRLOG



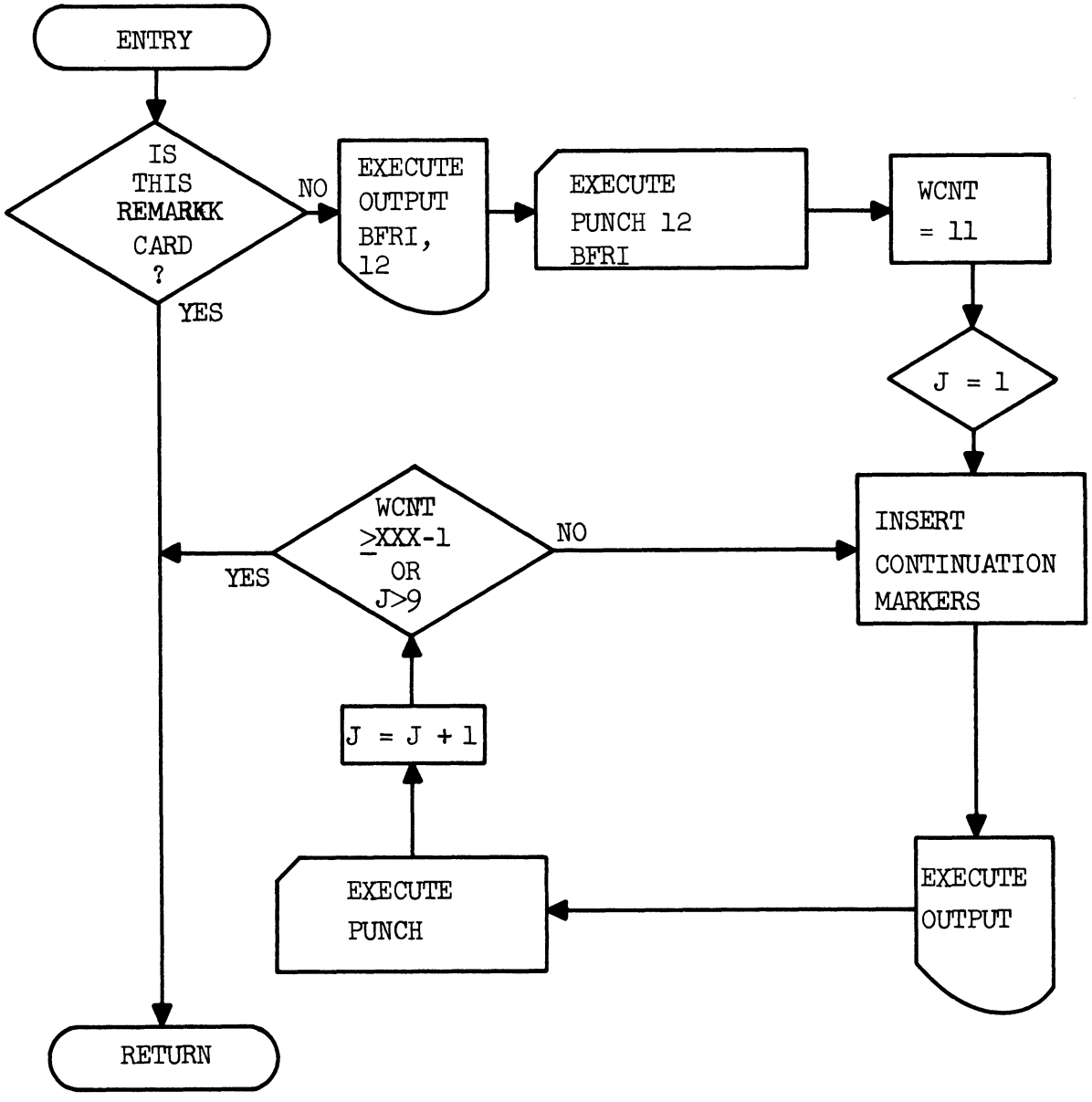

PSBK (0)
BRN = 2
PSBK (1)
READ DRUM 4, D4AD, BFR(12+III) ..... BFR(12+III+IIJ)
D4AD=D4AD+12
ENTRY
BRN = QIQ
III = 0
IIJ = 0
TO PSBK (BRN)
INTERNAL FUNCTION
PSCAN (QIQ)
CHAR = EXTRC (BFR(12+III + 1), 5)
READ BINARY 4
PSBK (1)
REMARK CARD ?
YES
NO
EXECUTE OUTPUT BFR(12III,12)
EXECUTE PUNCH 12. BFR(12III)
IIJ = IIJ + 1
IIJ + III + 1 > RCNO ?
YES
NO
TO PSBK (BRN)
FIRST CARD OF NEXT STATEMENT ?
YES
NO
BACK-SPACE TAPE 4
NO
BRN=2 ?
YES
D4AD=D4AD-12
COLI = EXTRC BFR(12III,1)
III = III + 1
III = III - 1
LAST CARD IN EPILOGUE ?
YES
PLSFG = 1B
NO
LAST CARD BEFORE CALCULATIONS OR EPILOGUE ?
YES
MINFLG = 1B
NO
III = III + 1
BFR(1)=INSRTC (BFRCI), 5,□.
BRN = BRN - 2BRN/2
III = III+ IIJ
RETURN


INTERNAL FUNCTIONS SSCAN, SSCANI
PAGE 1
LSPI (0)
TO LSPI (2-SW)
TO α
J>J1
NO
YES
J=J+1
J2=J2+1
BFR((J2-6)/6)= INSRTC(BFR((J2-1)/6), J2-6(J2-J1)/6), CHAR
EXECUTE SKIP (J2)
LSIN
EXECUTE QPQ (K INDEX)
CHAR IS &
YES
NO
TO LSPI (SW)
TO LSPI (0)
EXECUTE FSUB (QFQ)
CHAR IS 0
YES
NO
ENTRY, SSCAN
QFFQ=FSUB1
QPQ=PSUB1
SW=1
ENTRY, SSCANI
QFFQ=FSUB2
QPQ=PSUB2
PCOUNT=-1
PCOUNT=0
SW=0
DO β, J=1, 1, J>120
BFR1(J)=□
β
J1=72(II1+1)
IS THIS a REFERENCE CARD ?
YES
NO
TO LSP1 (1)
BFRI=INSRTC(BFRI,5,R)
J$1
α
J-1-72(J-1)/72 =0 AND J>1
YES
NO
J = J + 11
CHAR = EXTRC (BFR (J-1)/6, J-6(J-1)/6)
J<12
NO
YES
J2=J-1
CHAR IS 0
NO
YES
EXECUTE FSUB(QFFQ) J=11
J2=J2-1 CHAR =□
TO LSIN

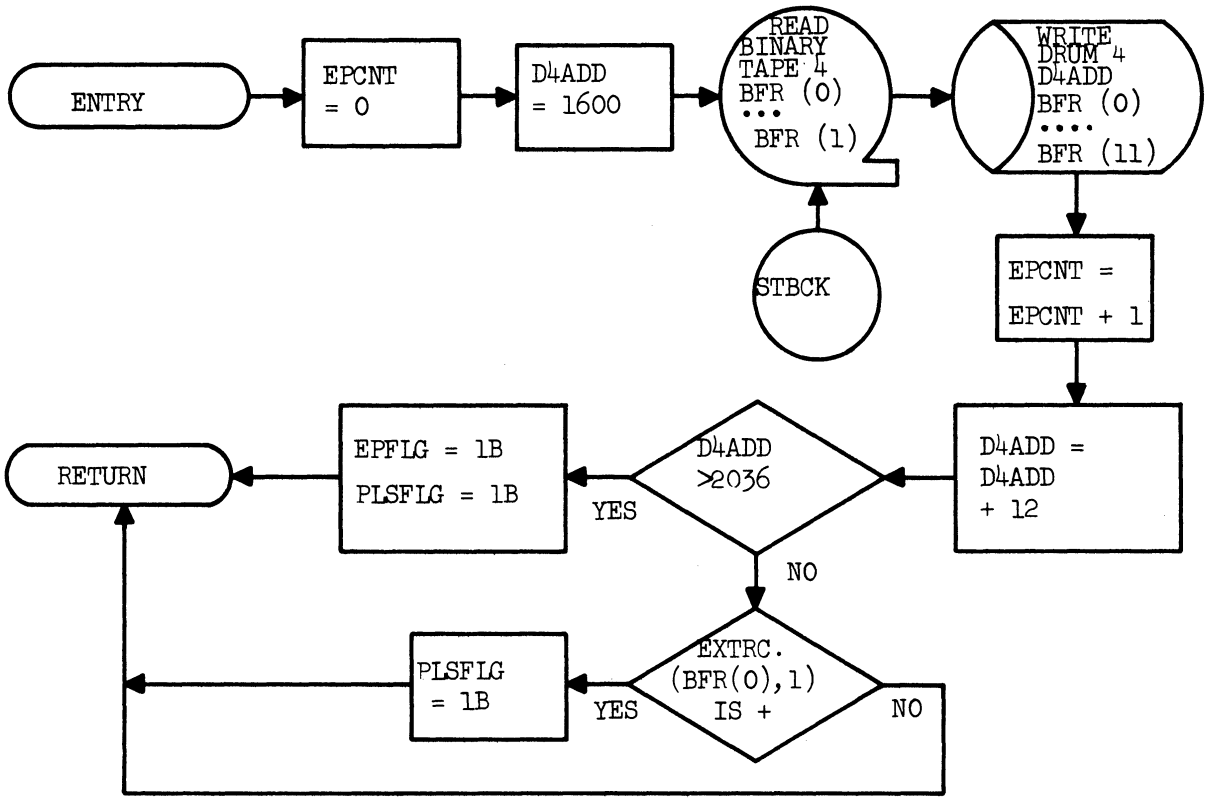
INTERNAL FUNCTIONS SSCAN, SSCAN1



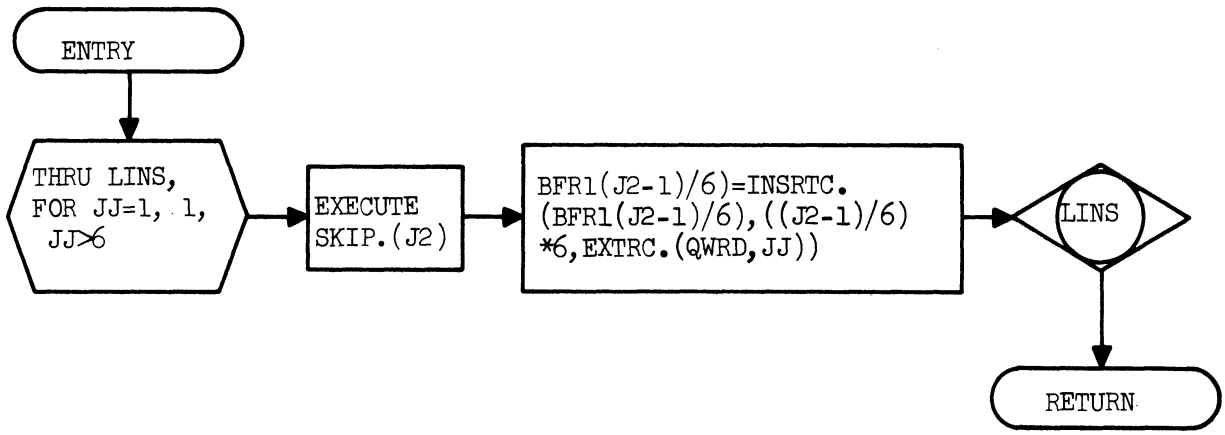
INTERNAL FUNCTION OUTPT1. (XXX)



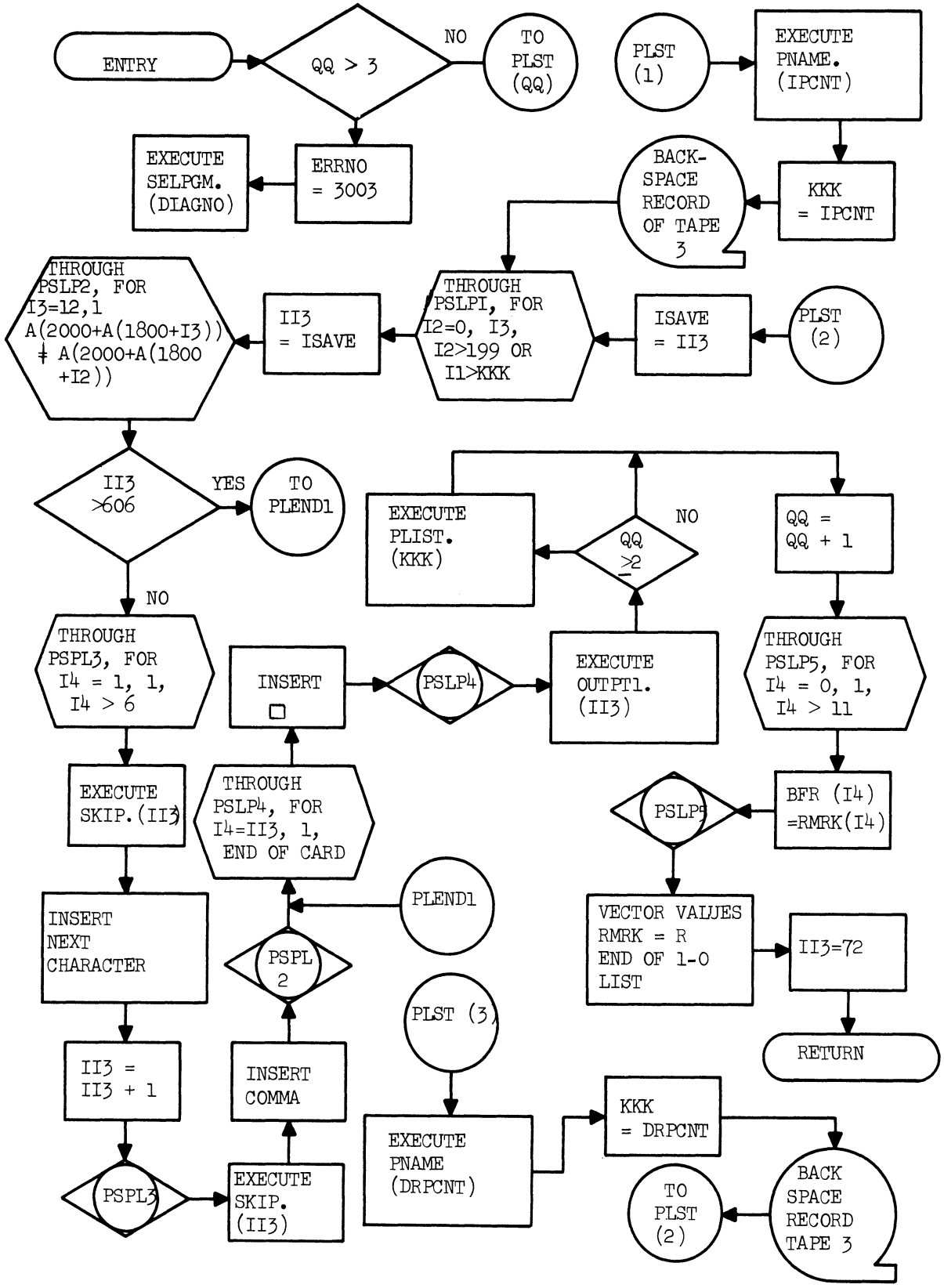
INTERNAL FUNCTION STORER.



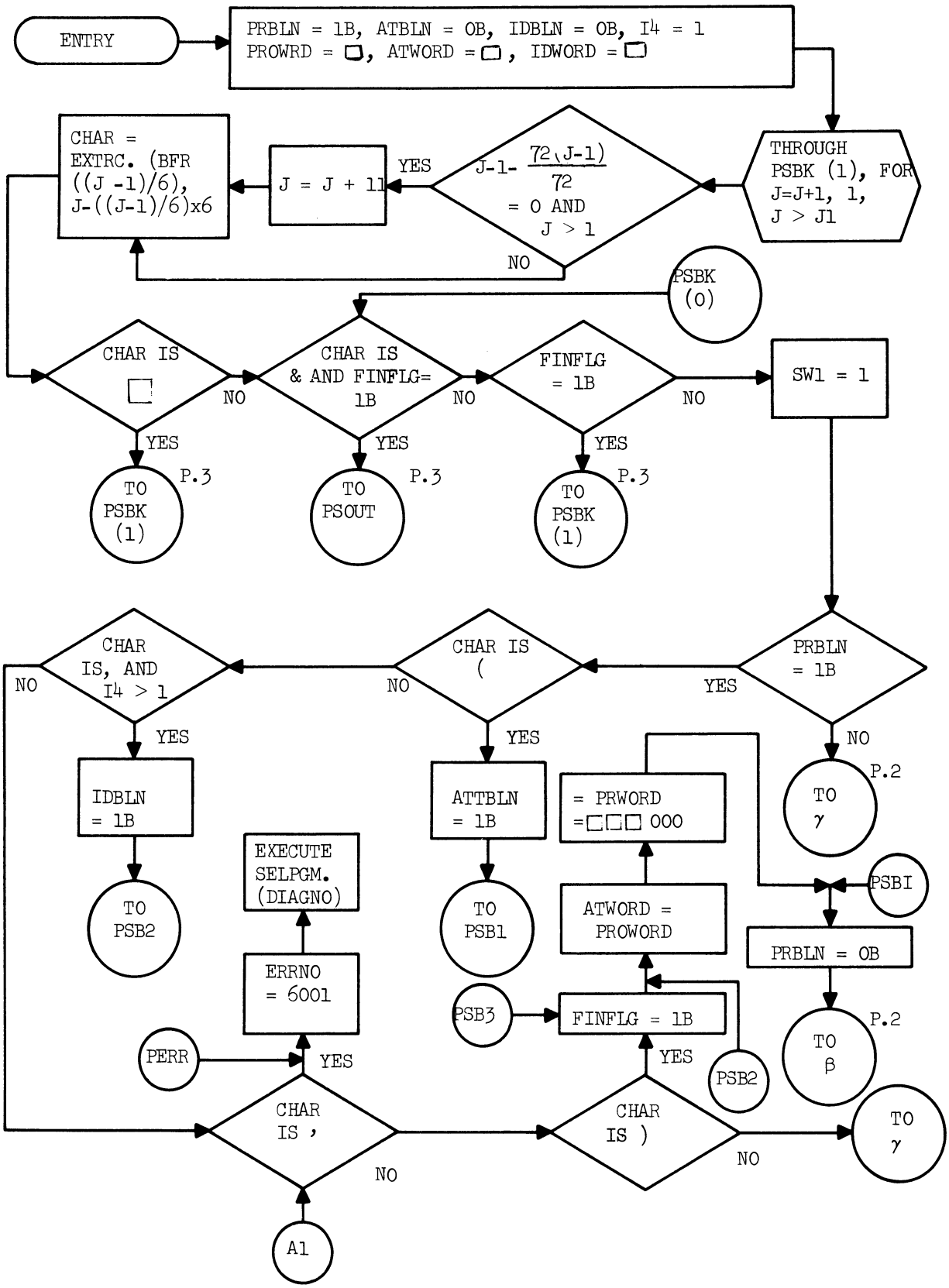
INTERNAL FUNCTION INSRT.



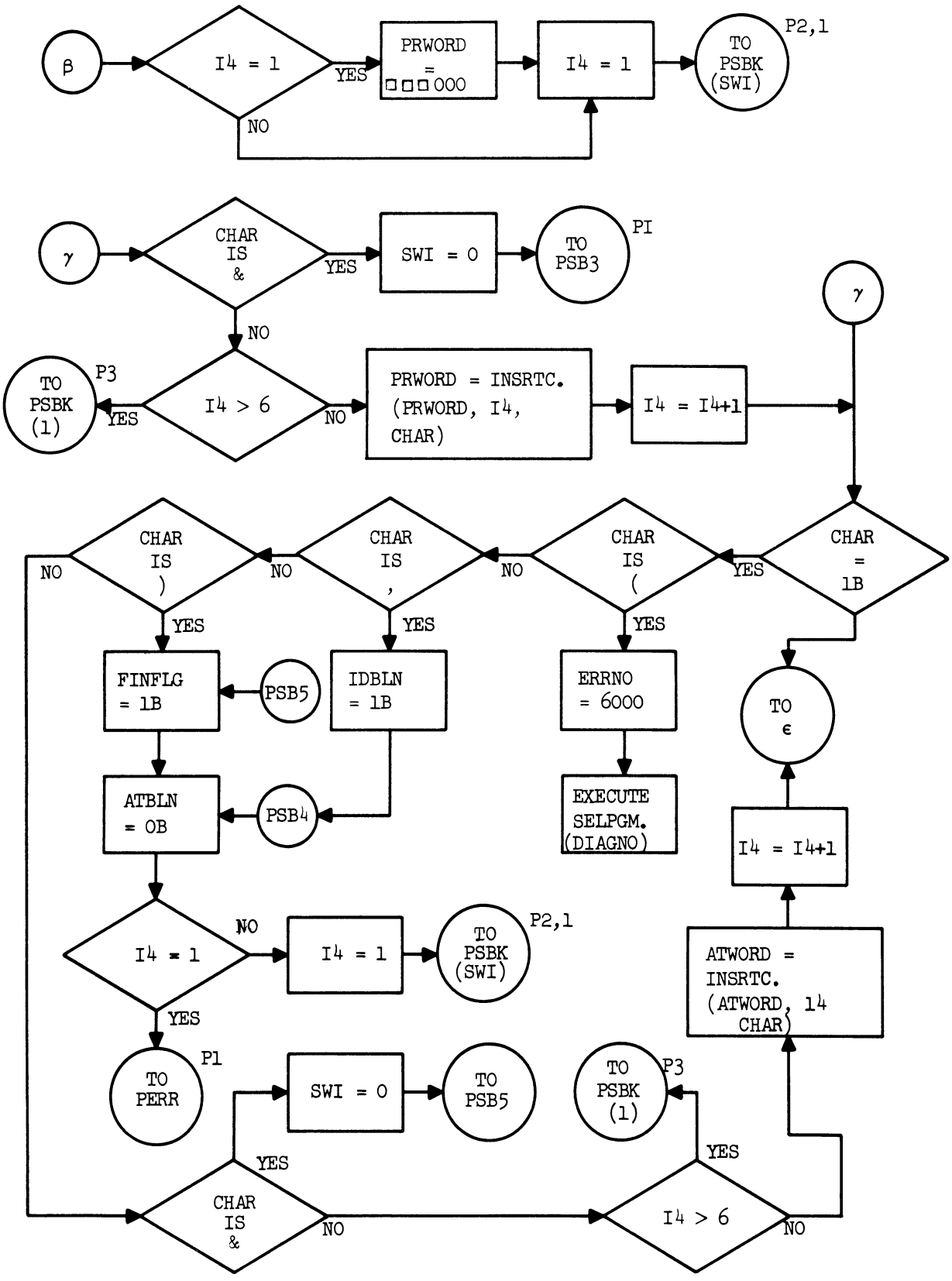
INTERNAL FUNCTION PSUB1 (QQ)



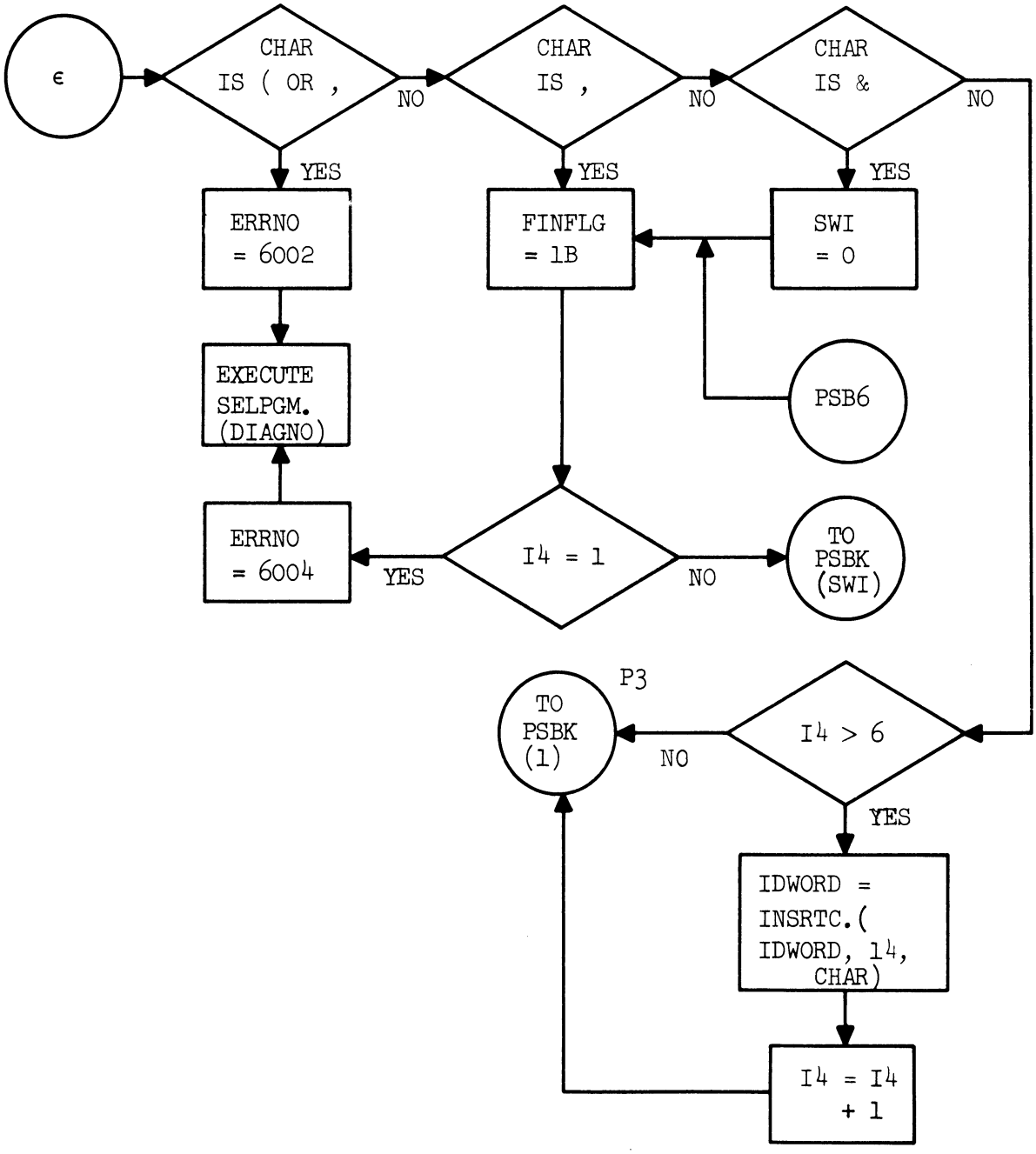
INTERNAL FUNCTION PSUB2. Page 1



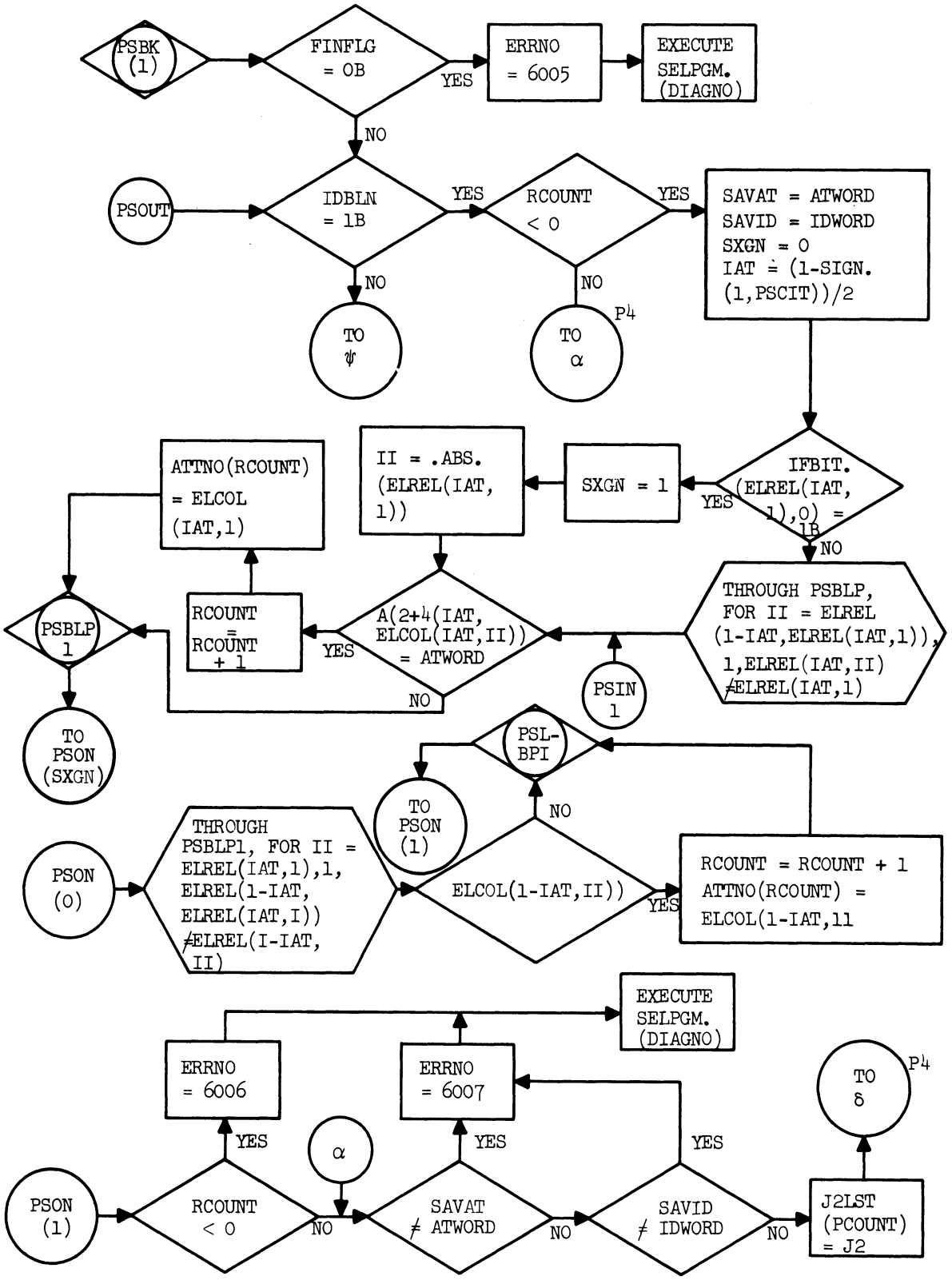
INTERNAL FUNCTION PSUB2



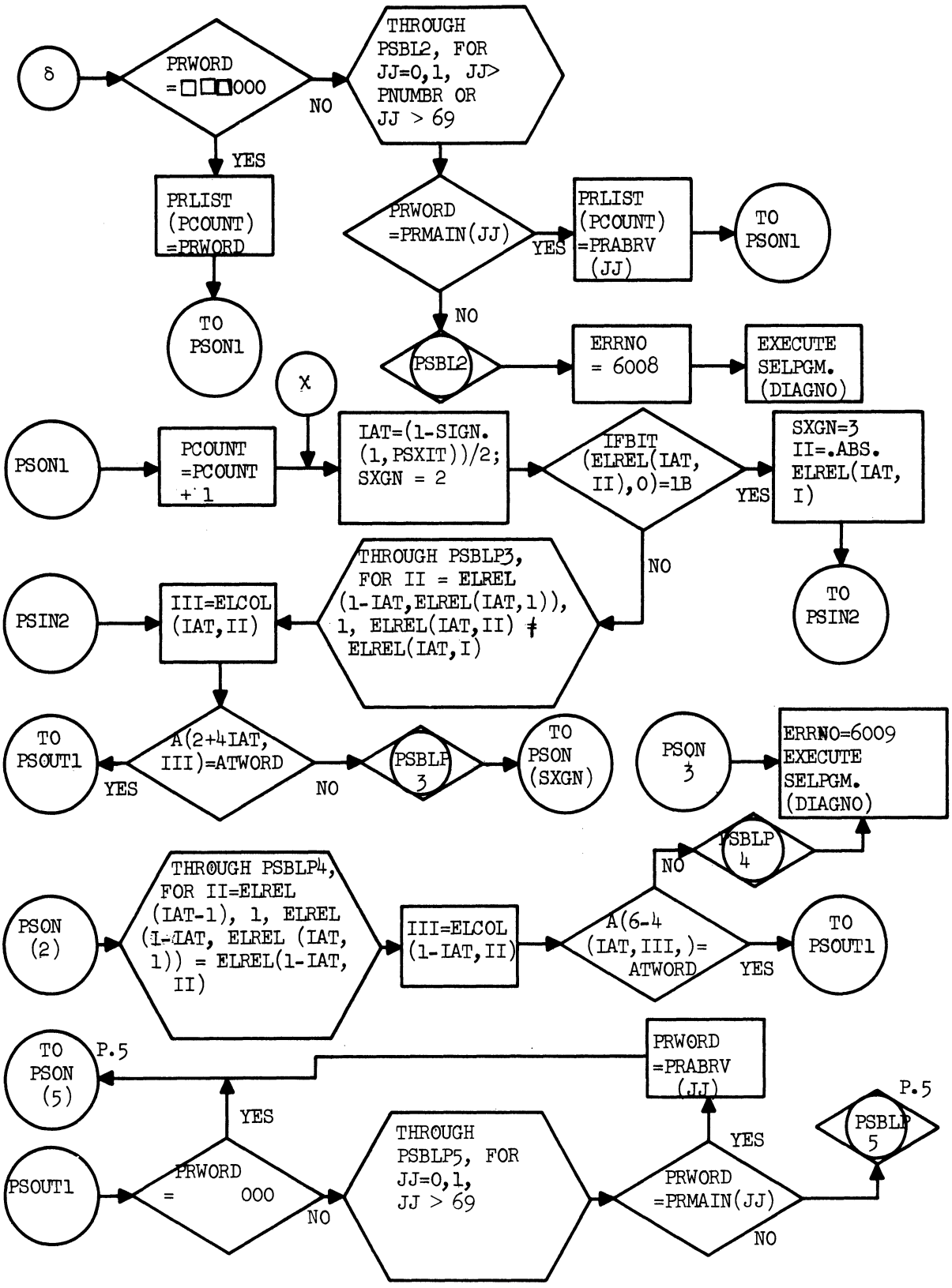
INTERNAL FUNCTION PSUB2

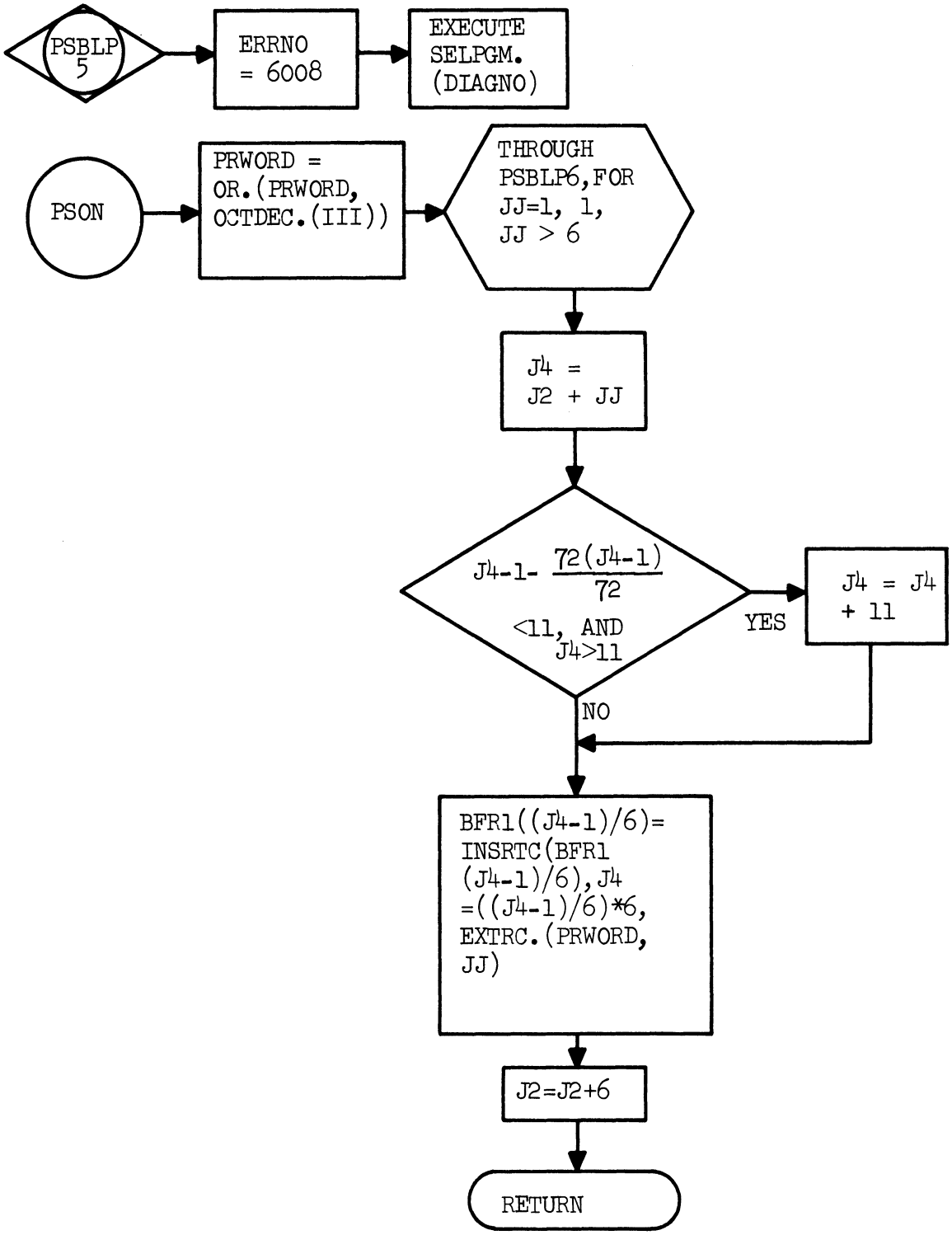


INTERNAL FUNCTION PSUB2

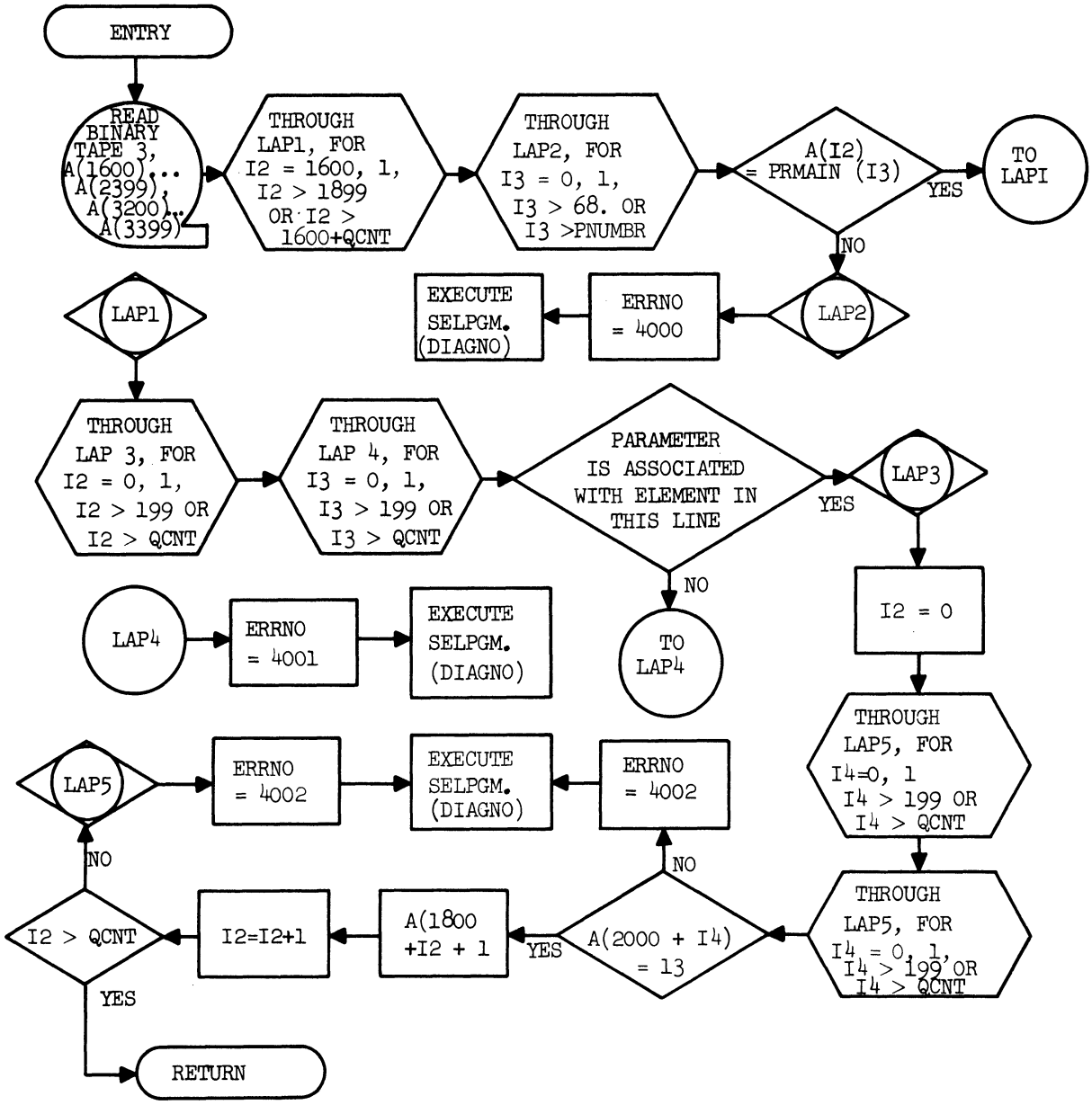


INTERNAL FUNCTION PSUB2 Page 4

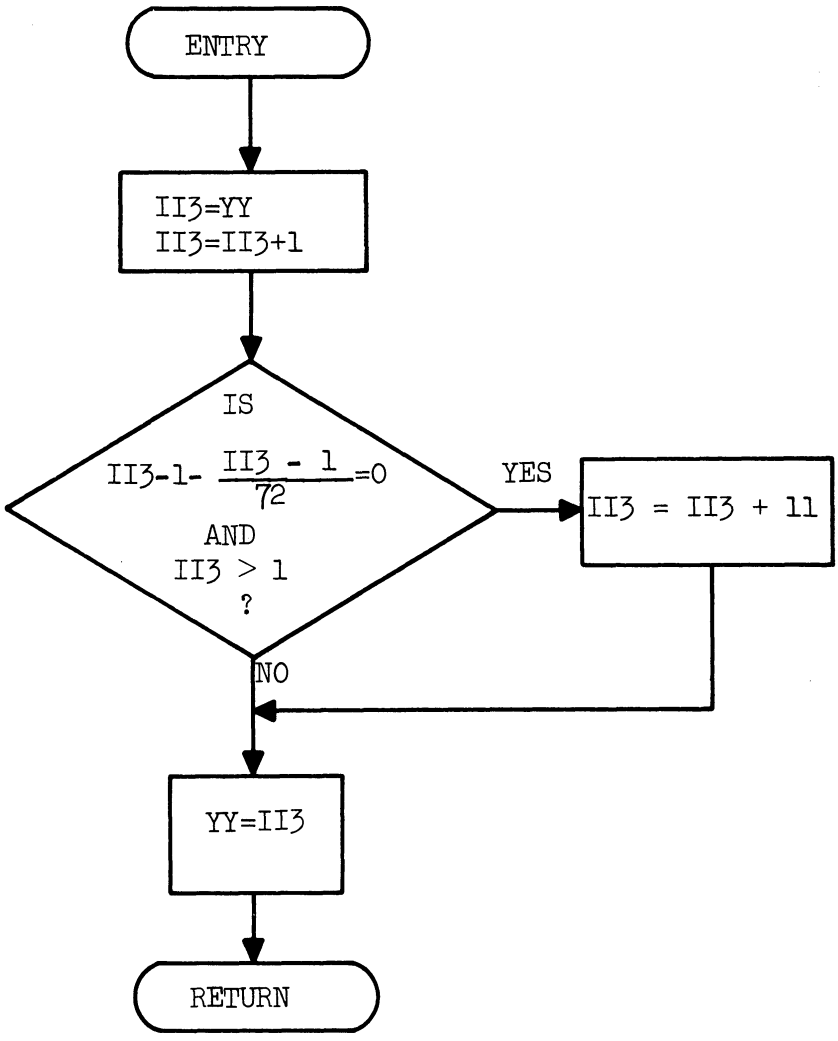



PSBLP 5
ERRNO = 6008
EXECUTE SELPGM. (DIAGNO)
PSON
PRWORD = OR.(PRWORD, OCTDEC.(III))
THROUGH PSBLP6,FOR JJ=1, 1, JJ > 6
J4 = J2 + JJ
J4-1- 72(J4-1)/72 <11, AND J4>11
YES
J4 = J4 + 11
NO
BFR1((J4-1)/6)= INSRTC(BFR1 (J4-1)/6),J4 =((J4-1)/6)*6, EXTRC.(PRWORD, JJ)
J2=J2+6
RETURN

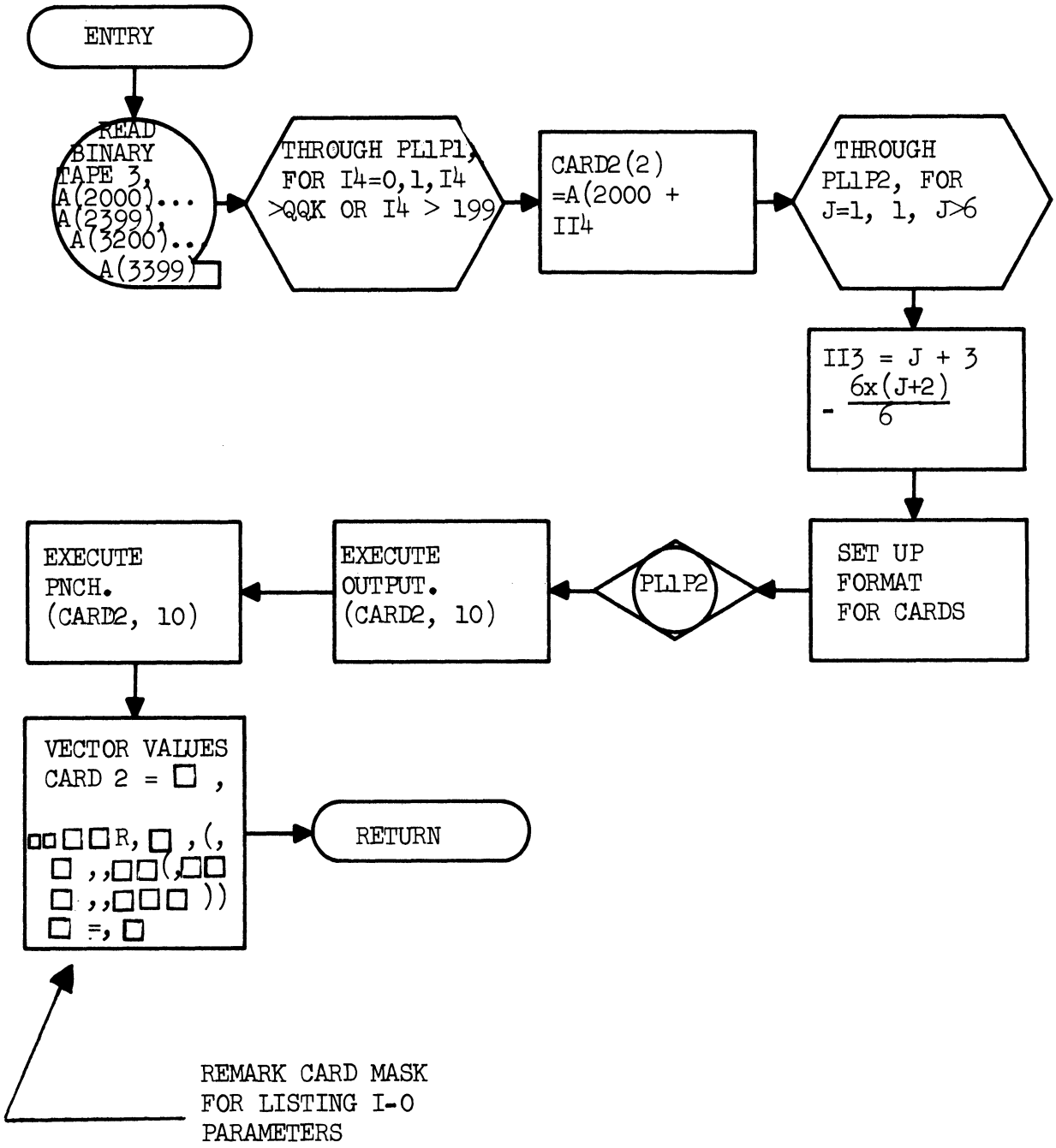
INTERNAL FUNCTION PNAME. (QCNT)



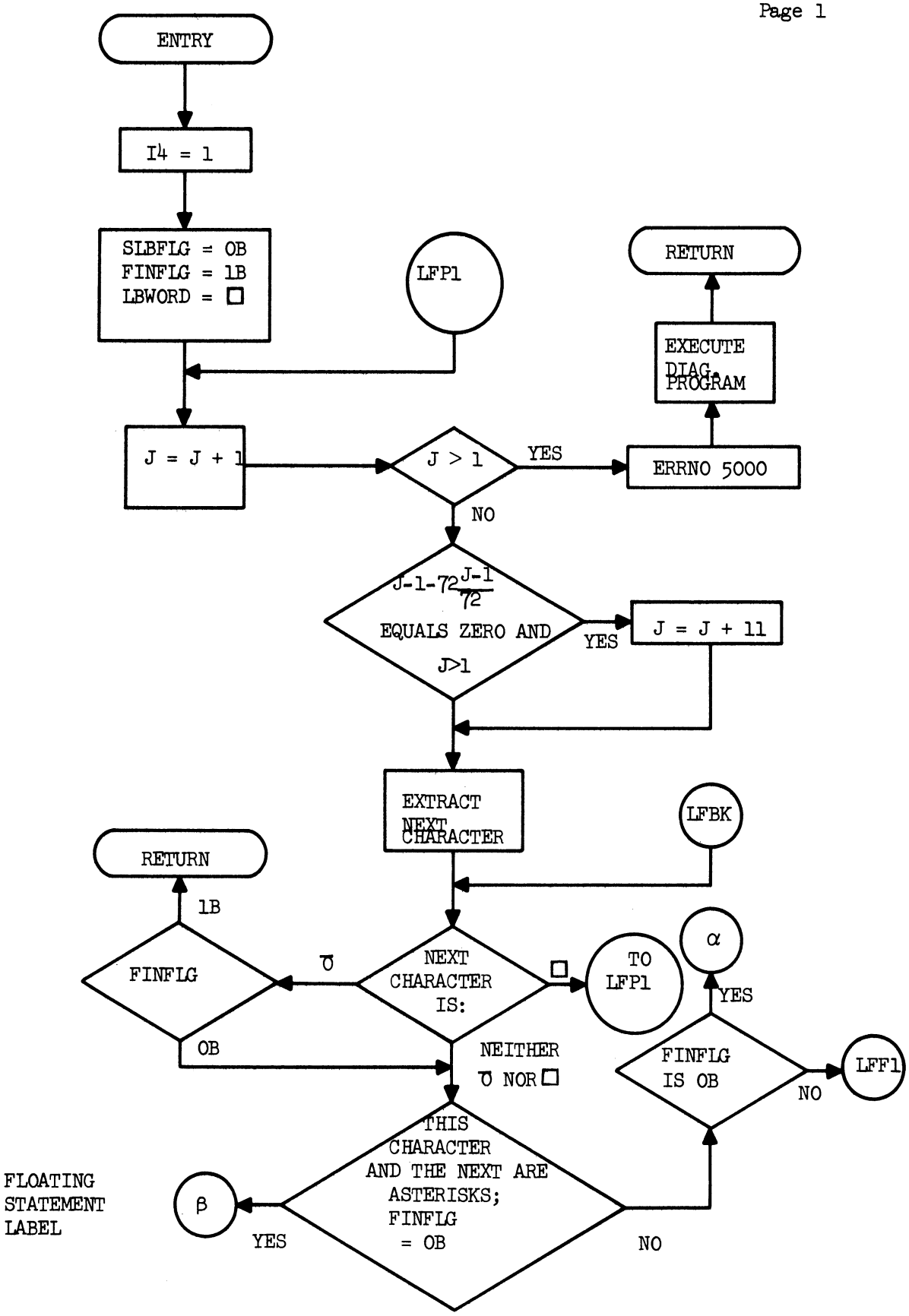
INTERNAL FUNCTION - SKIP (YY)



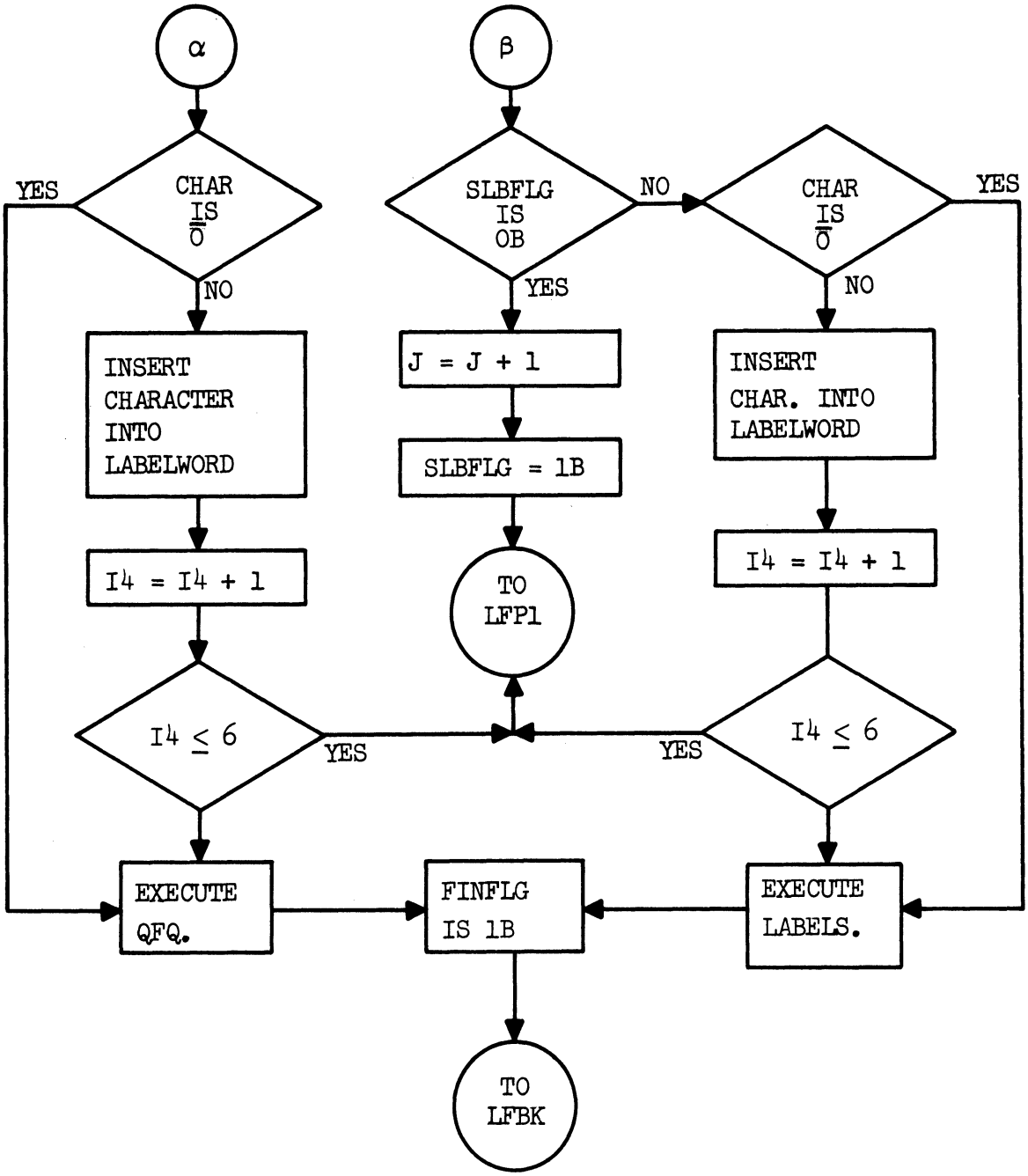
INTERNAL FUNCTION PLIST. (QQK')



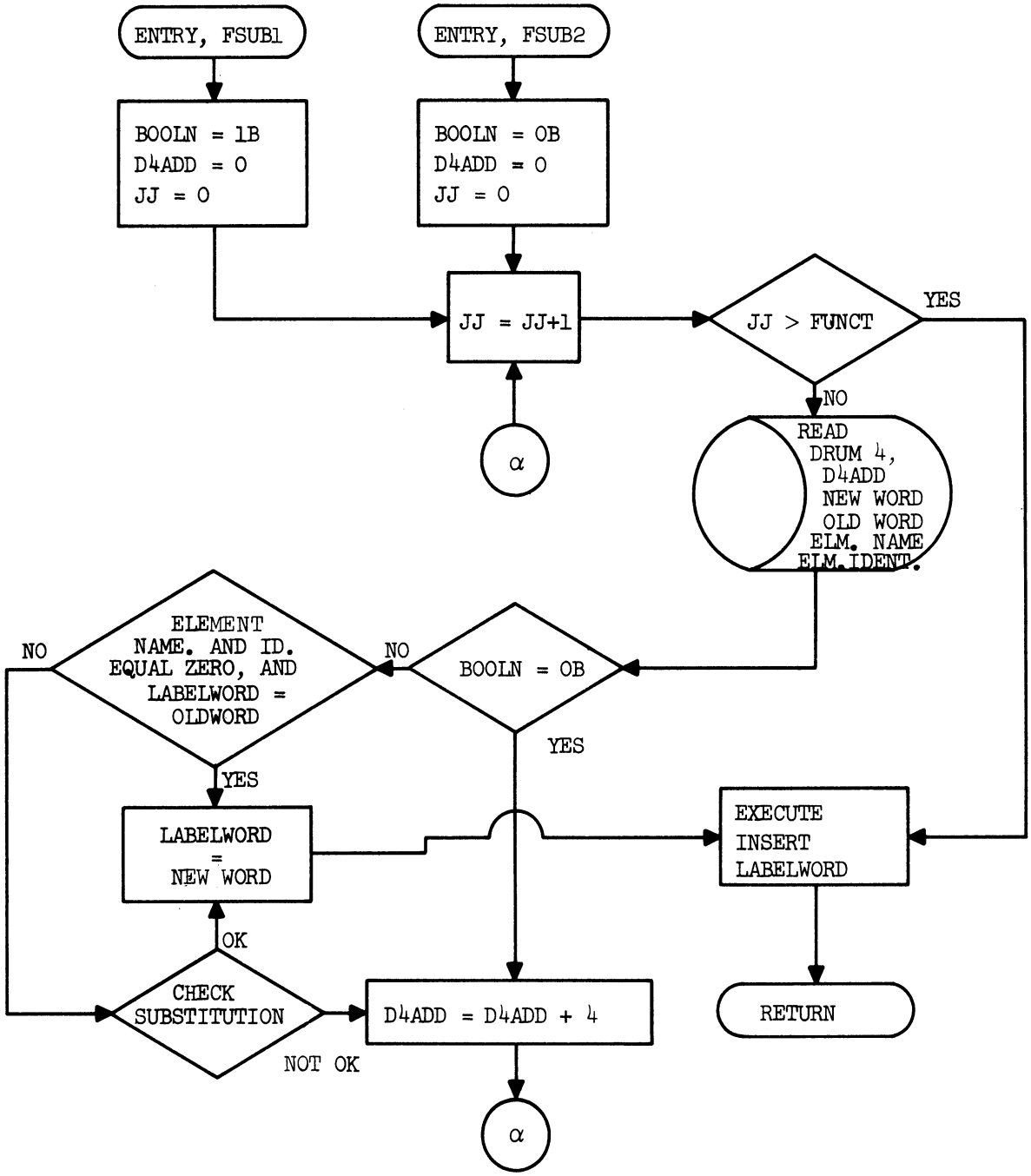
INTERNAL FUNCTION - FSUB (QFQ.)



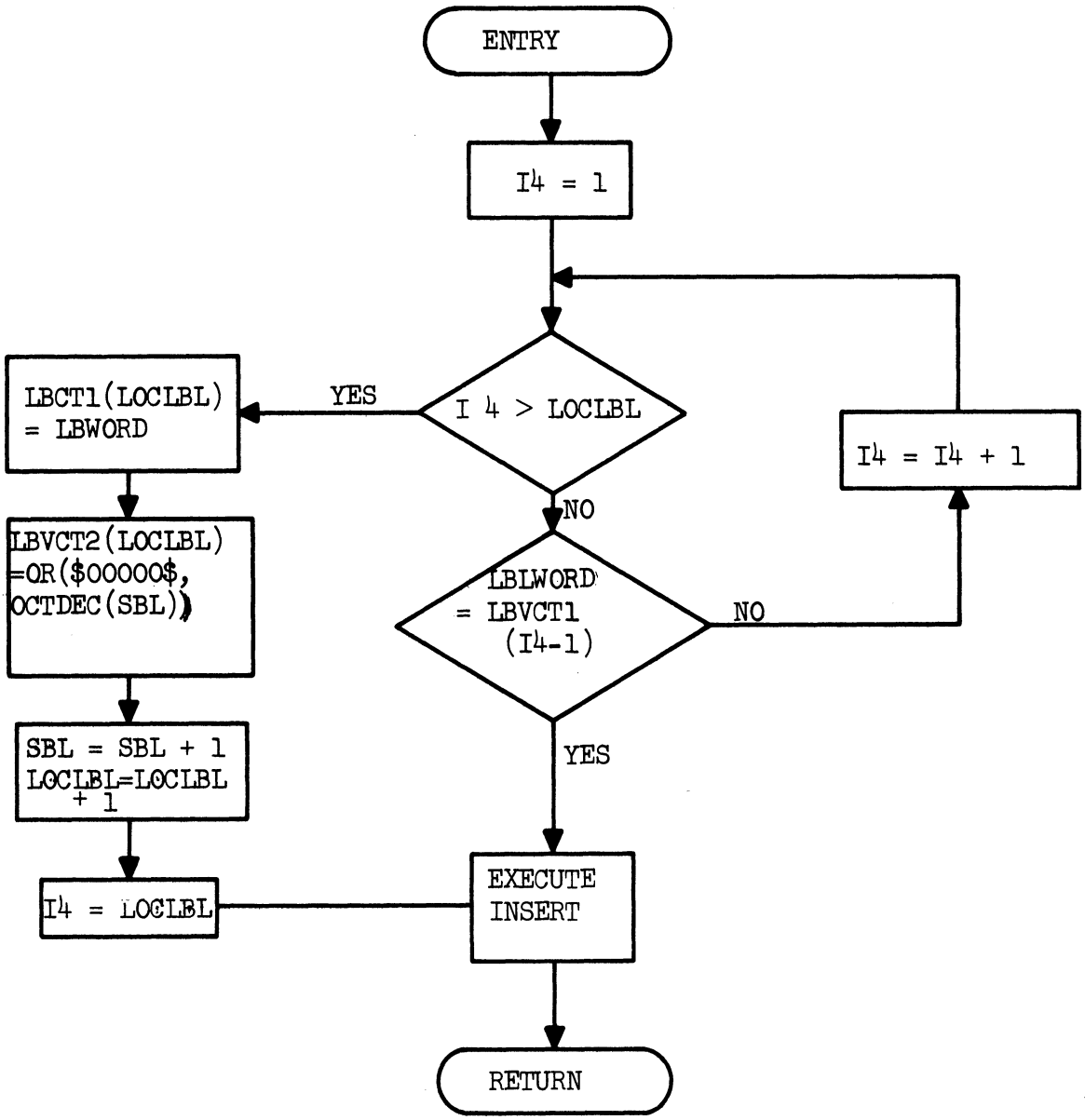
INTERNAL FUNCTION FSUB (Q.F.Q.)



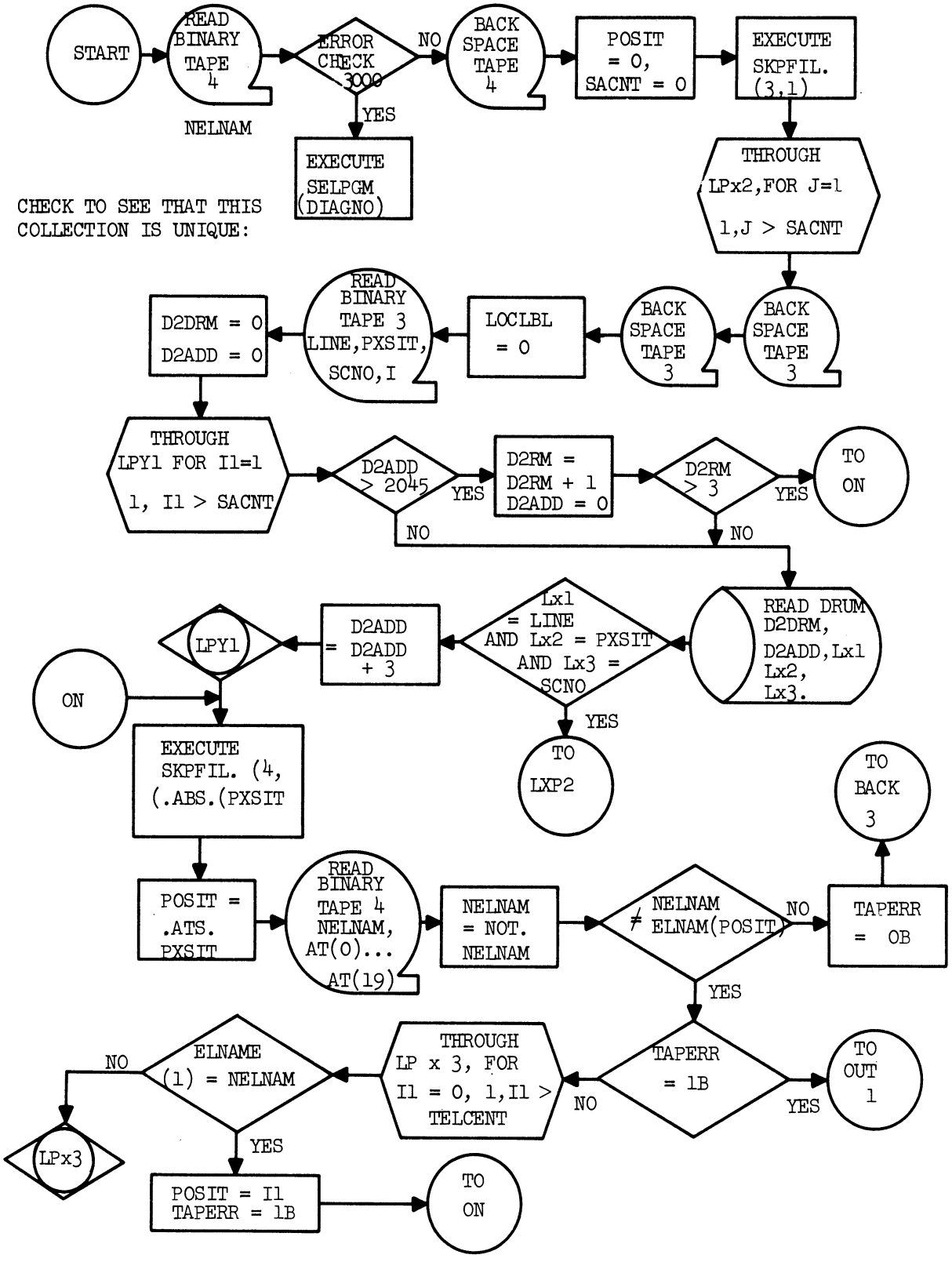
INTERNAL FUNCTIONS--FSUB1 FSUB2



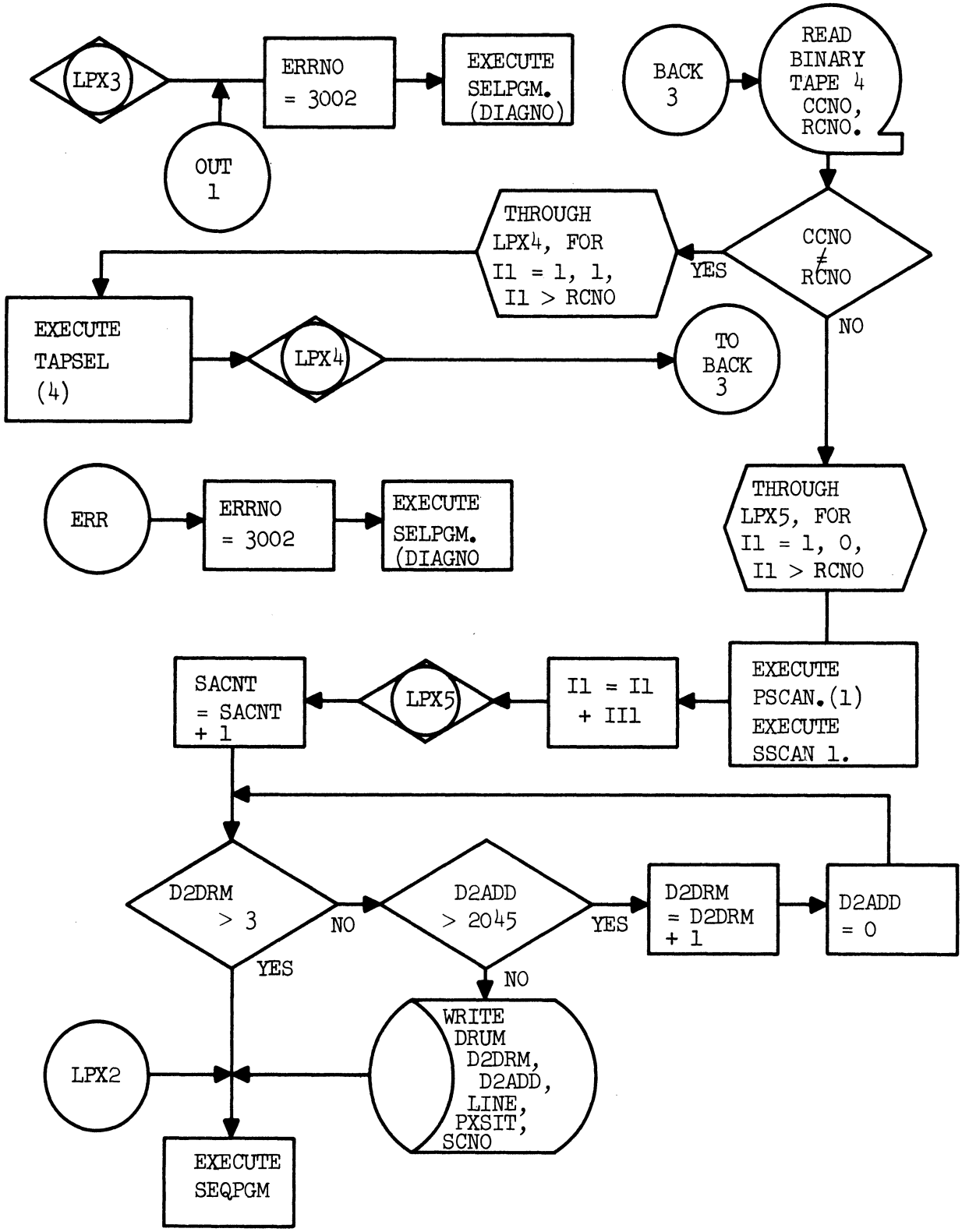
INTERNAL FUNCTION - LABELS



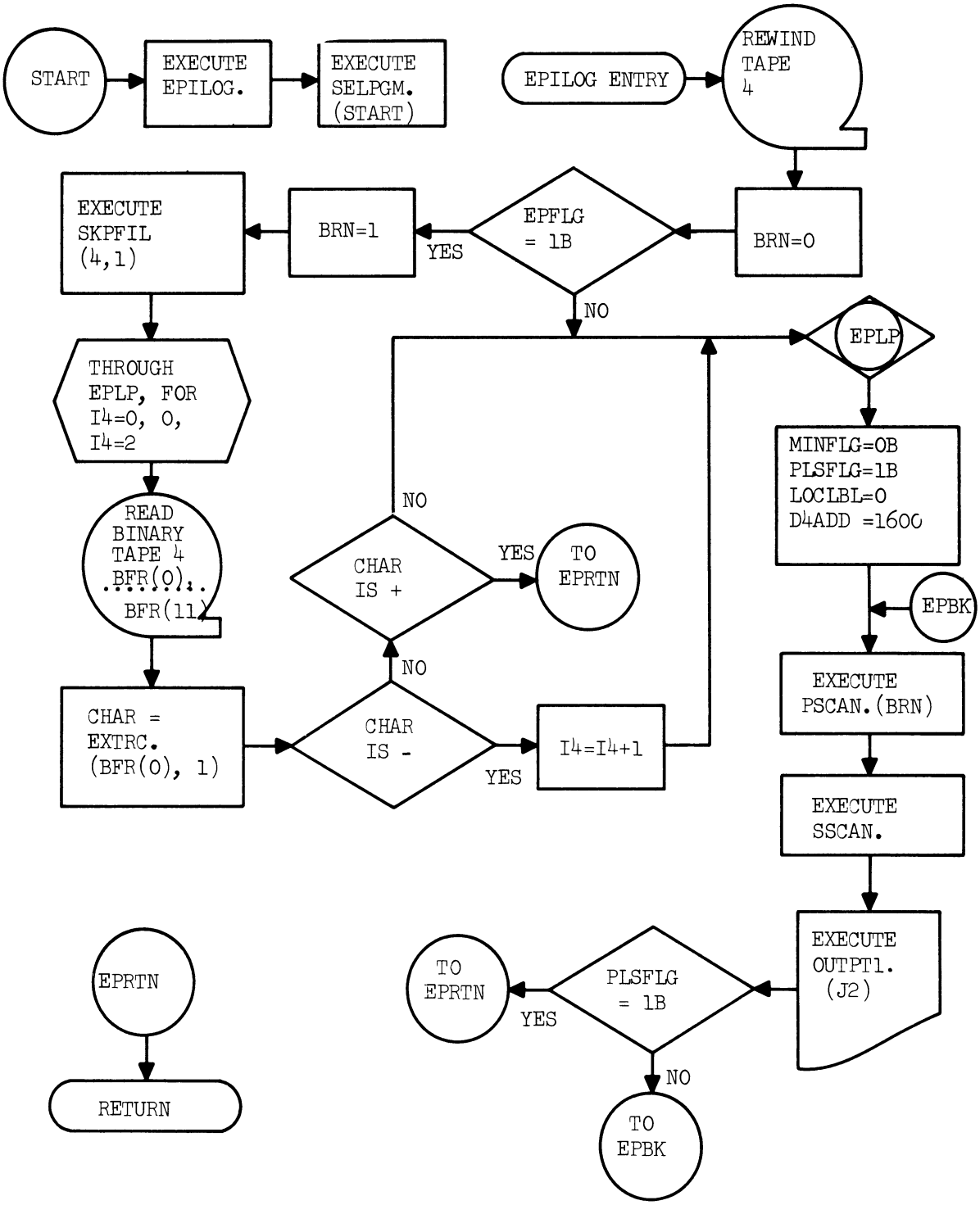
PROGEN SECTION



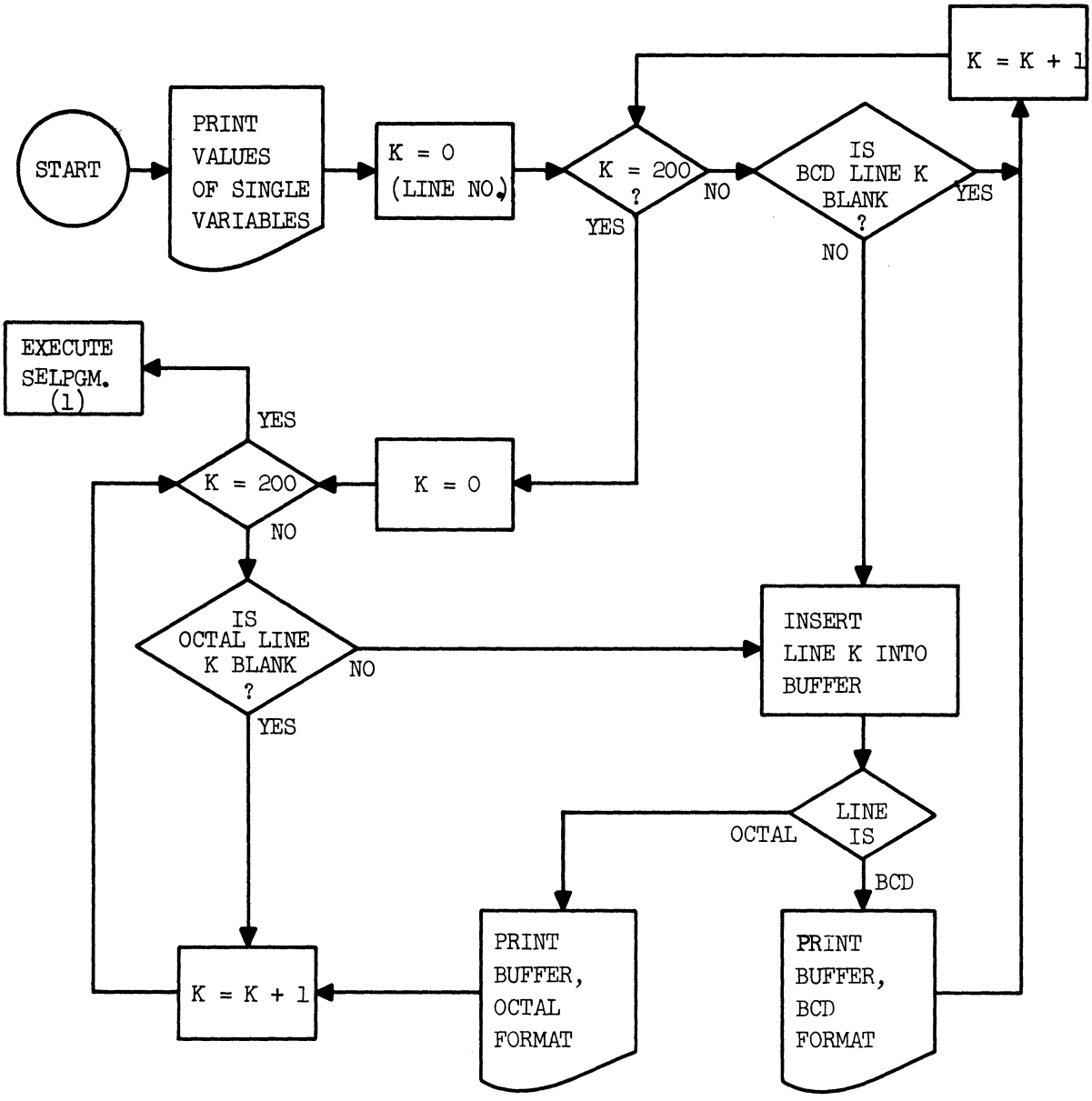
PROGEN SECTION



EPILOGUE; MAIN PROGRAM AND INTERNAL FUNCTION



DIAGNOSTIC CORE



STEPWISE REGRESSION

FLOW DIAGRAM AND CORE LAYOUTS

CORE LAYOUTS

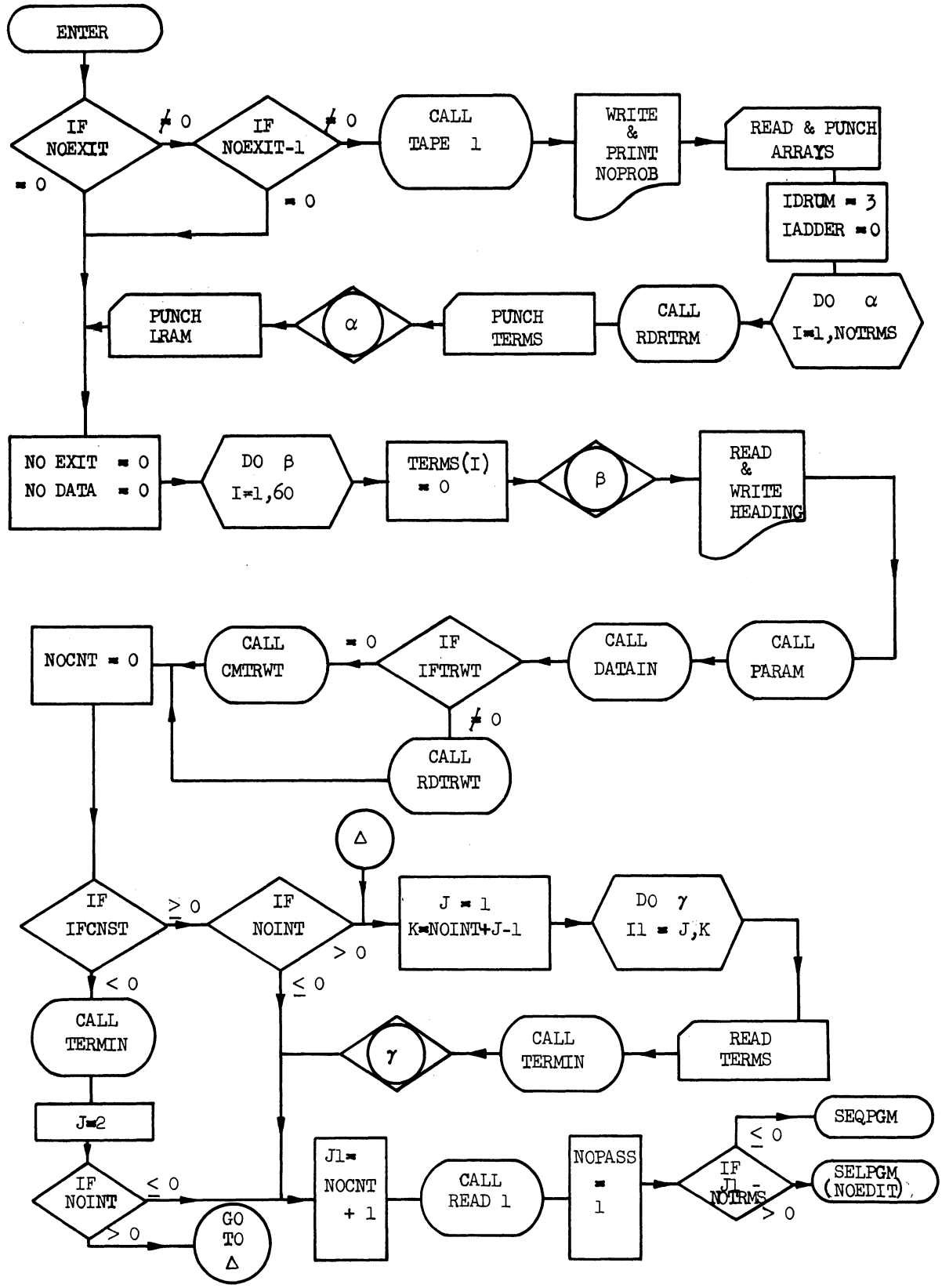
(MAIN)		(SUB)	SUB-SUB	(FUNCTION)	SUB-SUB-SUB
STARTER	1/2	TAPE 1	FORMAT	SEQPGM	
		RDRTRM	TRMCHK	SELPGM	
		PARAM	ENTRM		
		DATAIN			
		RDTRWT			
		CMTRWT			
		TERMIN			
		READ 1			
STARTER	2/2				
		PKTRM	XLOG	SELPGM	
		TERMIN	RAM2A		TRMCHK
			PICKV		ENTRM
			PICKS		
			RAM2C		
			RAM2B		
			PICKE		
STUDENT					
		TAPE 1	EXP(3	SELPGM	
		RDRTRM	TRMCHK		
		GRADER	ENTRM		
		NRML			
		PRINT 4			
		ENTRM			
		PKTRM			
		TERMIN			
EDITOR					
		RDRTRM	PFNCT	SELPGM	
		TAPE 1	EXP(3	ZFNCT	
			EXP(2		
REGRESSION	1/2				
		SUMSQ	TAPE 1	SELPGM	
		RSDSUM	PRINT 2		
		PRCRCN	PRINT 3		
			SQRT		
REGRESSION	2/2				
		RGRSSN	DGFRDM	SELPGM	SQRT
		WINDUP	VARSRT	SEQPGM	PRINT 1
			VARCHK		PREDCT
			MATRAN		TSTLVL
			RGRTRM		FLVL
					DTAB
					XTAB
					ERROR
					TAPE 1

CORE LAYOUTS CONT'D

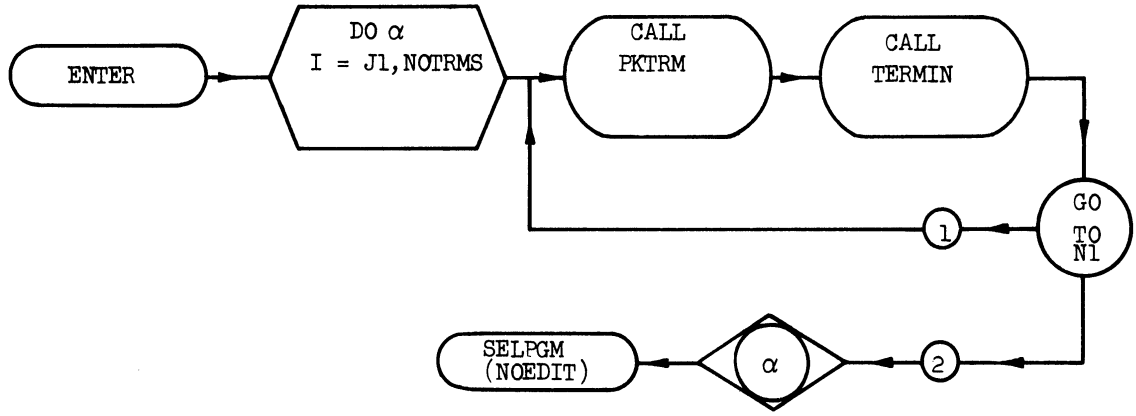
(MAIN)	(SUB)	SUB-SUB	(FUNCTION)	SUB-SUB-SUB
STATEMENT GENERATOR				
	PROCES	ERROR	SELPGM	
	PROLOG	GTRM		
	DIMNSN	EXPNT		
	CTERM			
	RDRTRM			
	GENTRM			
	SUMEQN			

STARTER PROGRAM

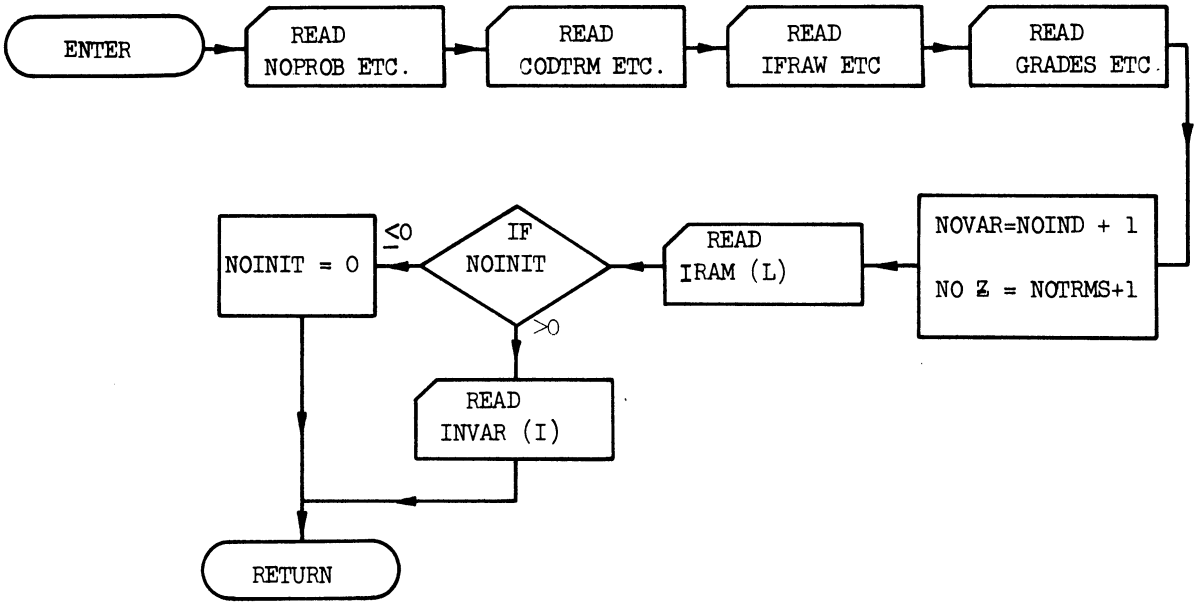
First Part



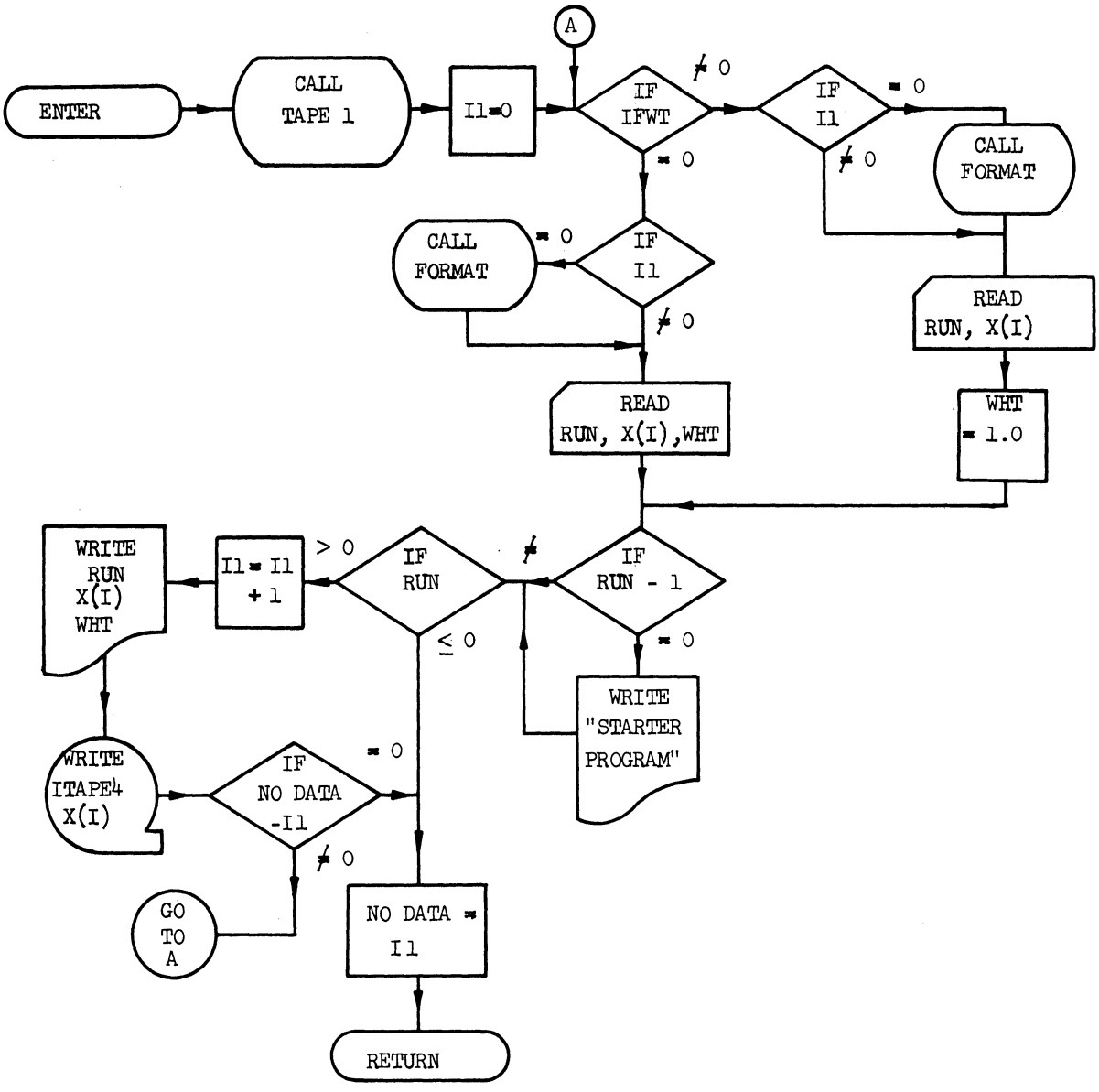
STARTER PROGRAM
Second Part



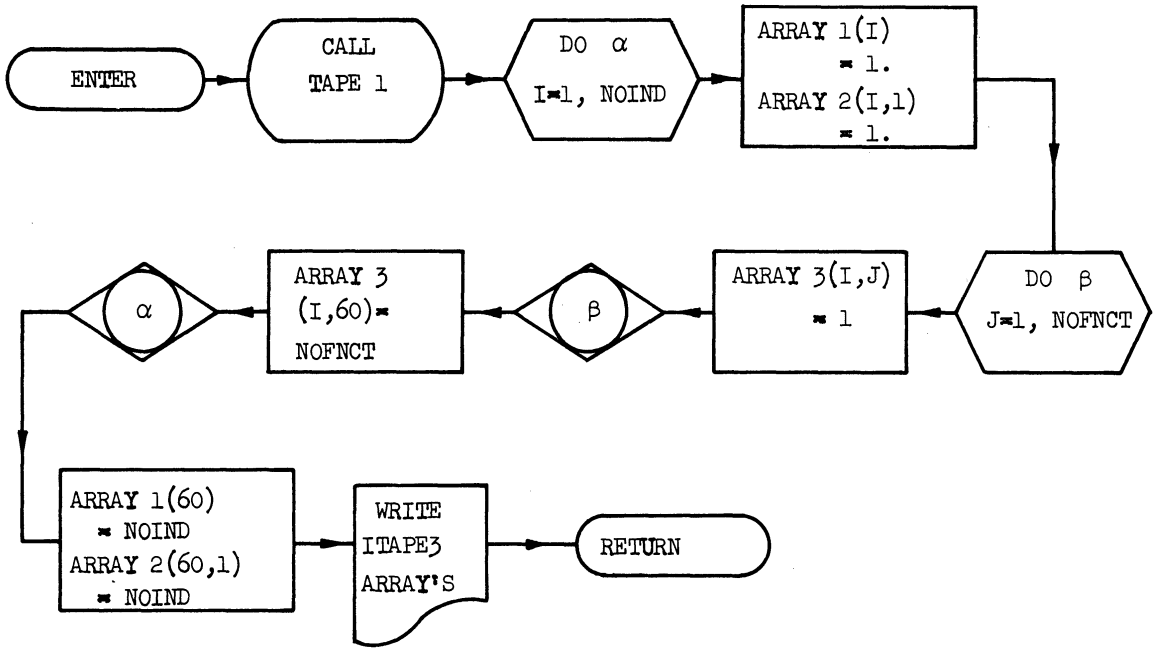
PARAM



DATAIN

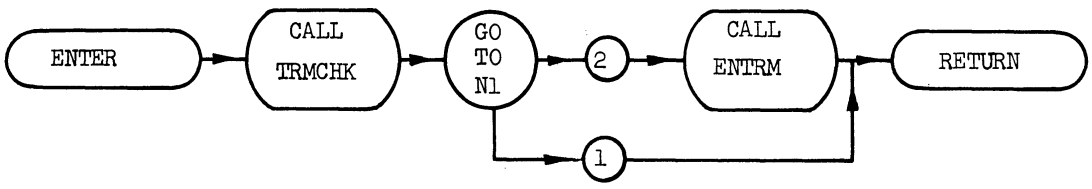

A
ENTER
CALL TAPE 1
I1=0
IF IFWT
≠ 0
IF I1
= 0
≠ 0
CALL FORMAT
= 0
IF I1
CALL FORMAT
≠ 0
READ RUN, X(I)
WHT = 1.0
READ RUN, X(I),WHT
IF RUN - 1
≠
= 0
WRITE "STARTER PROGRAM"
IF RUN
> 0
≤ 0
I1 = I1 + 1
WRITE RUN X(I) WHT
WRITE ITAPE4 X(I)
IF NO DATA -I1
= 0
≠ 0
GO TO A
NO DATA = I1
RETURN

CMTRWT

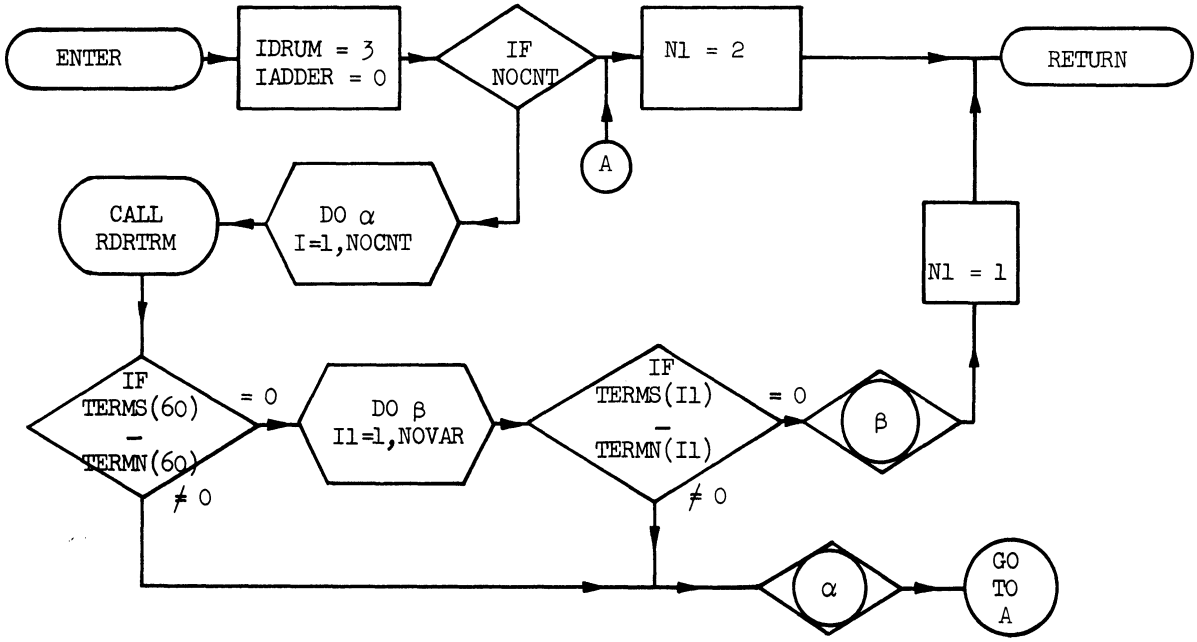


RDTRWT

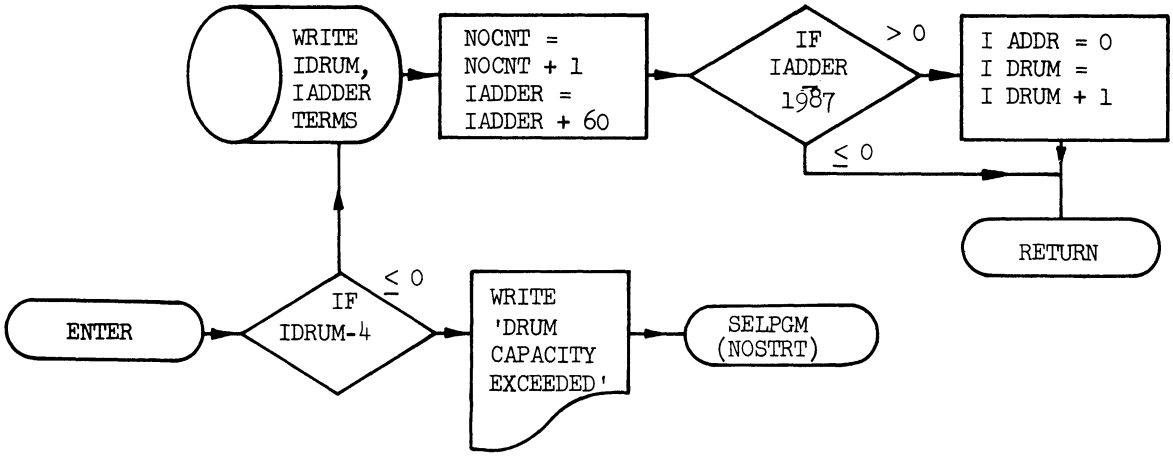
TERMIN



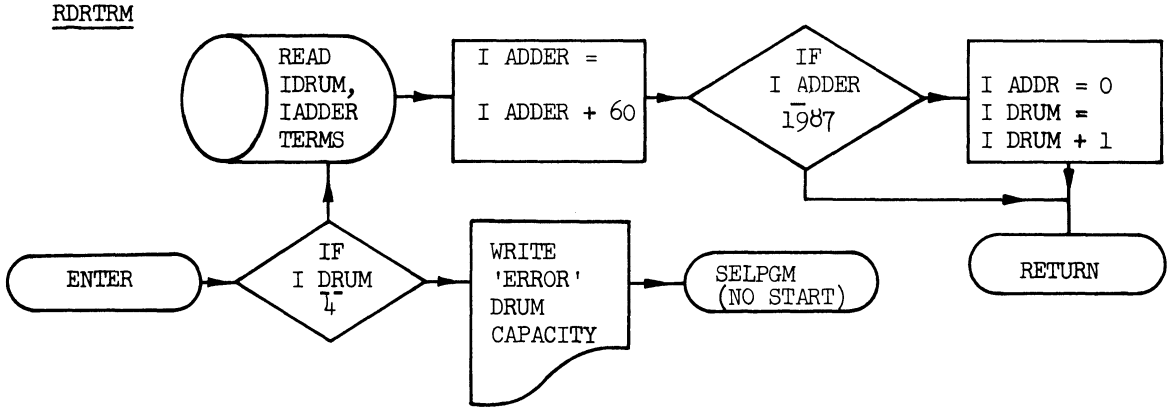
TRMCHK.



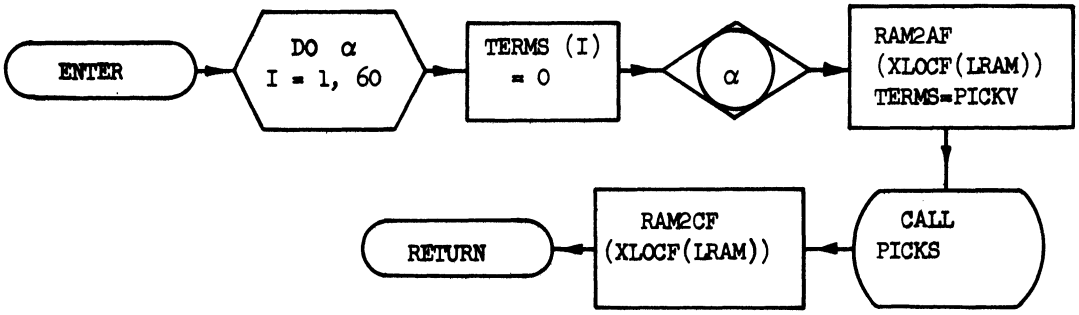
ENTRM



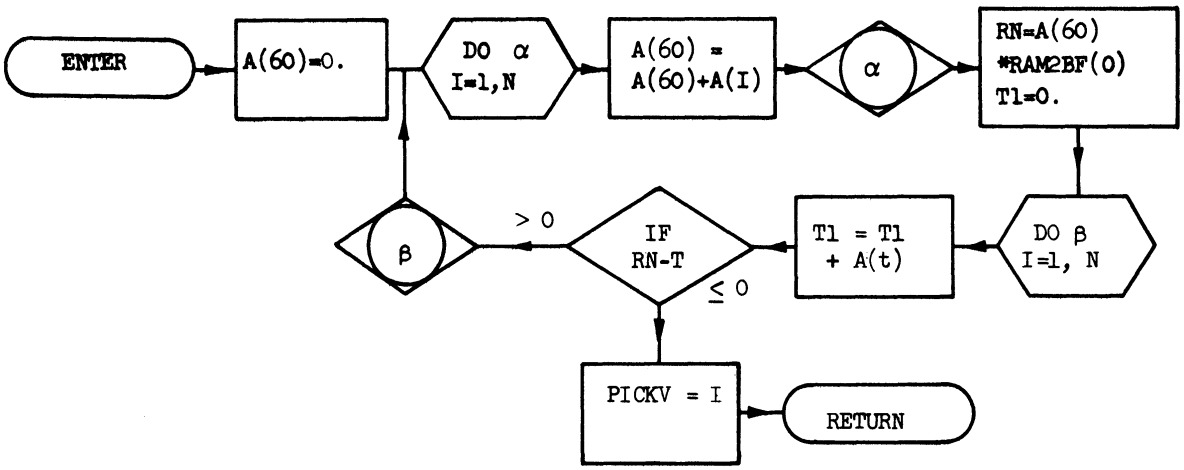
RDRTRM



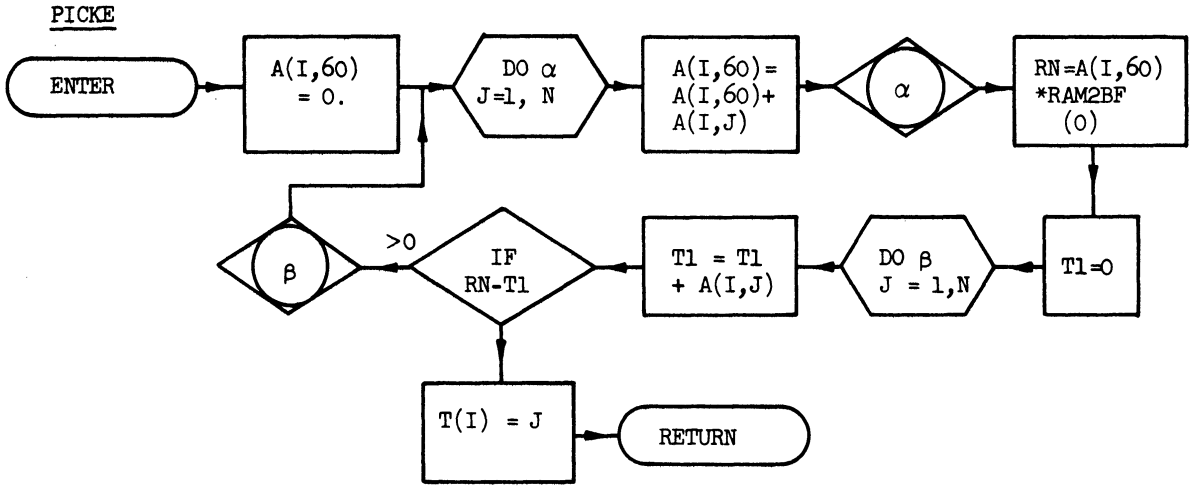
·PKTRM



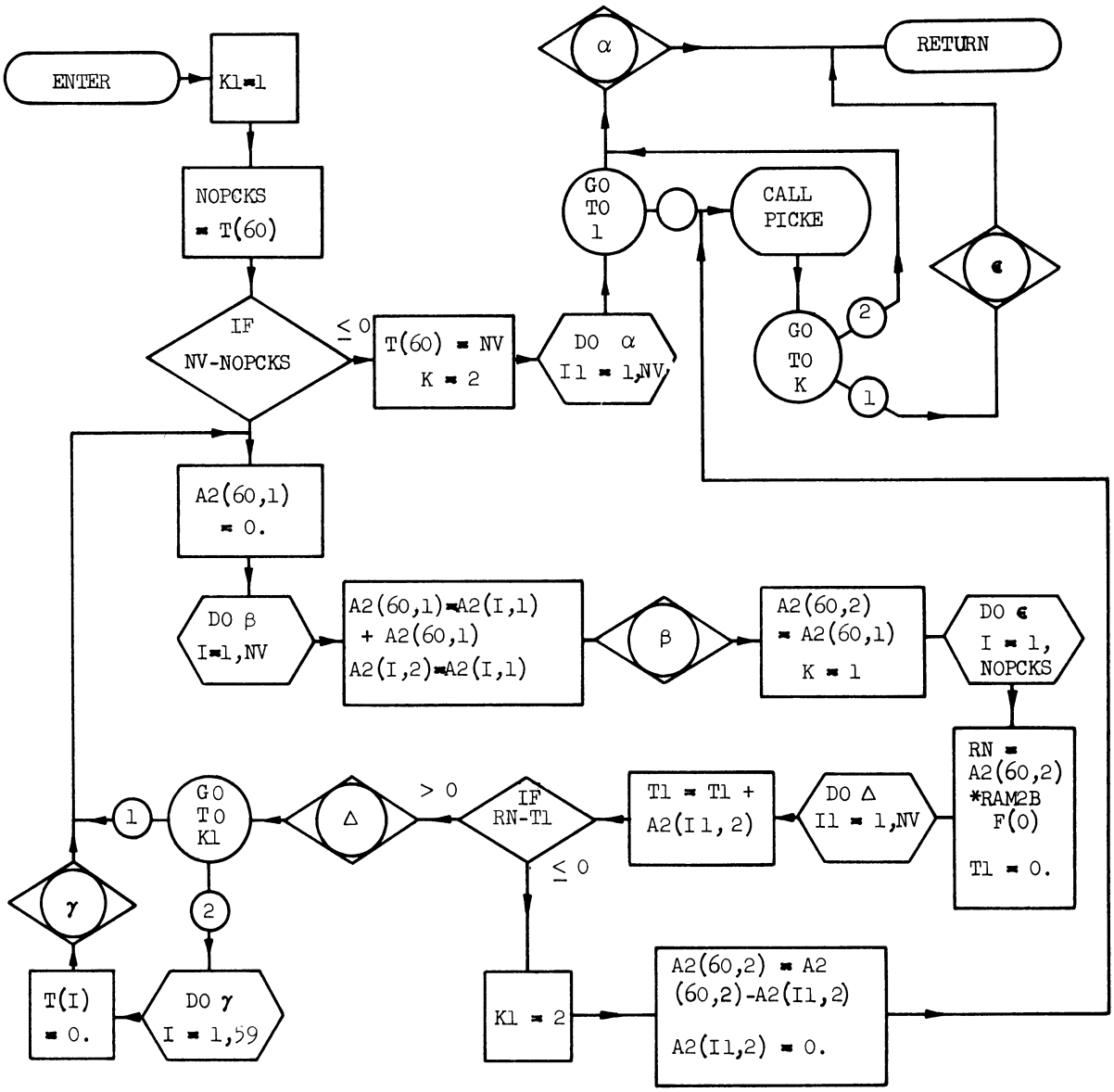
PICKV
FUNCTION



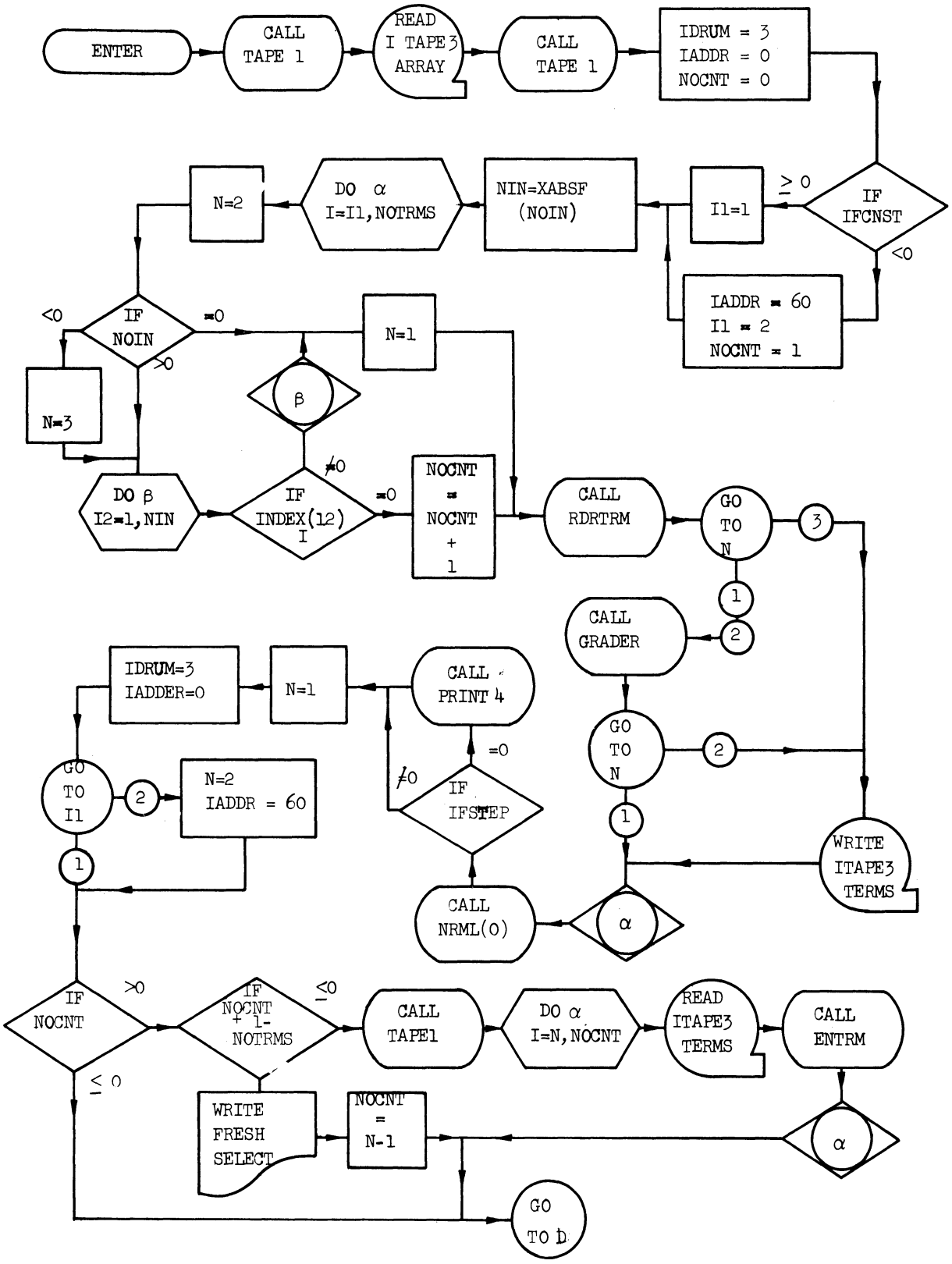
PICKE



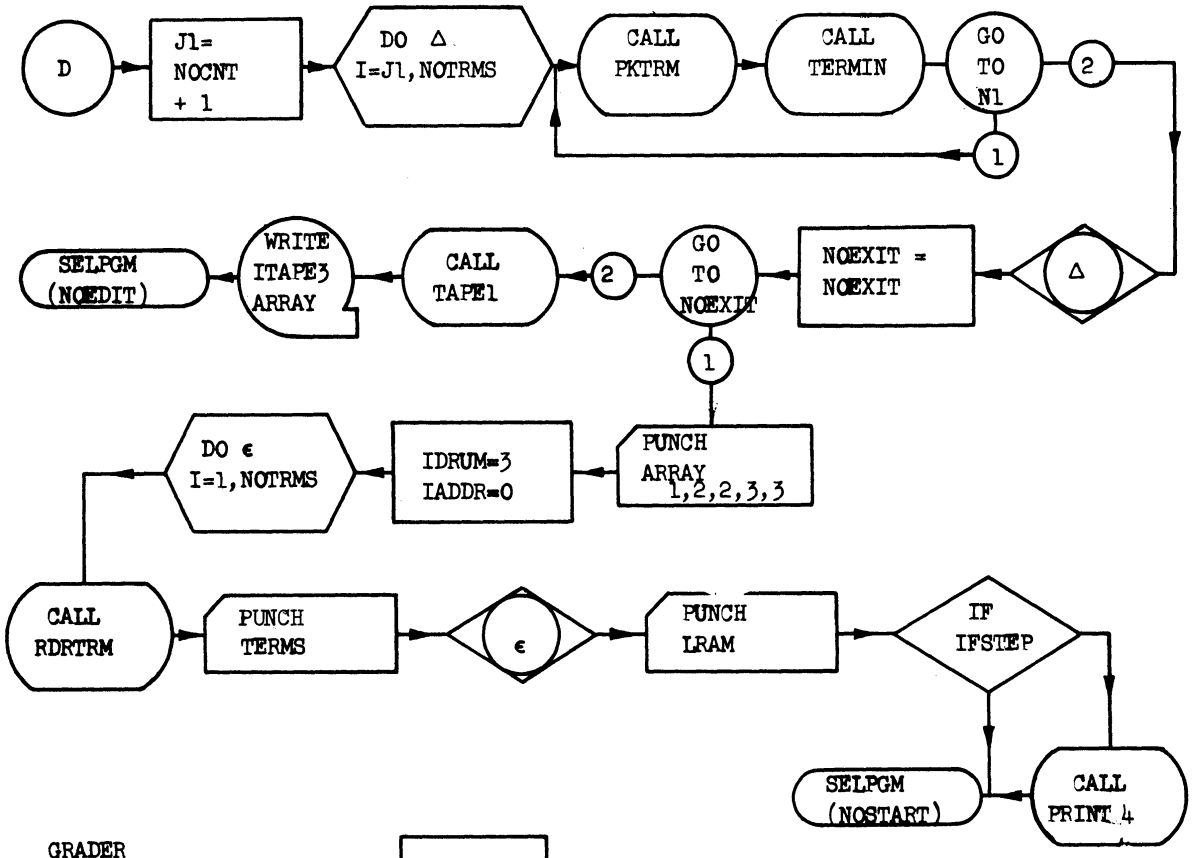
PICKS

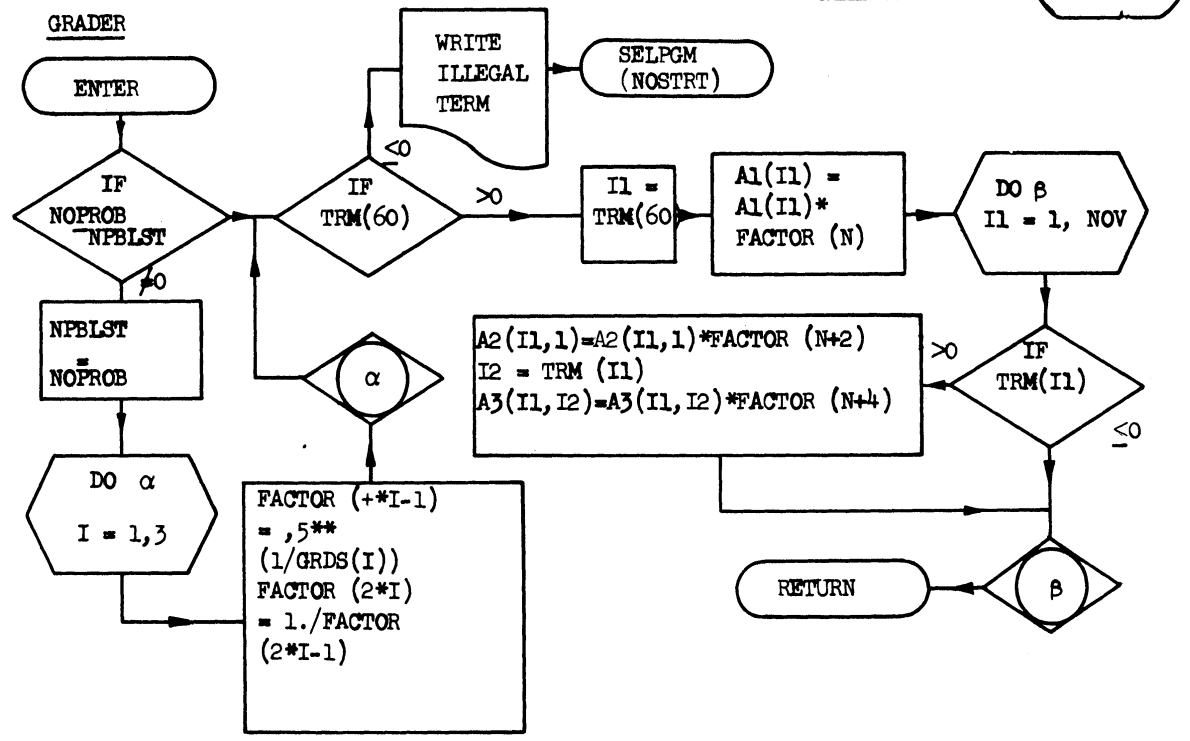

ENTER
K1=1
NOPCKS = T(60)
IF NV-NOPCKS
≤ 0
T(60) = NV
K = 2
DO α I1 = 1,NV
GO TO 1
α
RETURN
CALL PICKE
GO TO K
2
1
ε
A2(60,1) = 0.
DO β I=1,NV
A2(60,1)=A2(I,1) + A2(60,1)
A2(I,2)=A2(I,1)
β
A2(60,2) = A2(60,1)
K = 1
DO ε I = 1, NOPCKS
RN = A2(60,2) *RAM2B F(0)
T1 = 0.
DO Δ I1 = 1,NV
T1 = T1 + A2(I1,2)
IF RN-T1
> 0
≤ 0
Δ
GO TO K1
1
2
γ
T(I) = 0.
DO γ I = 1,59
K1 = 2
A2(60,2) = A2 (60,2)-A2(I1,2)
A2(I1,2) = 0.

STUDENT PROGRAM

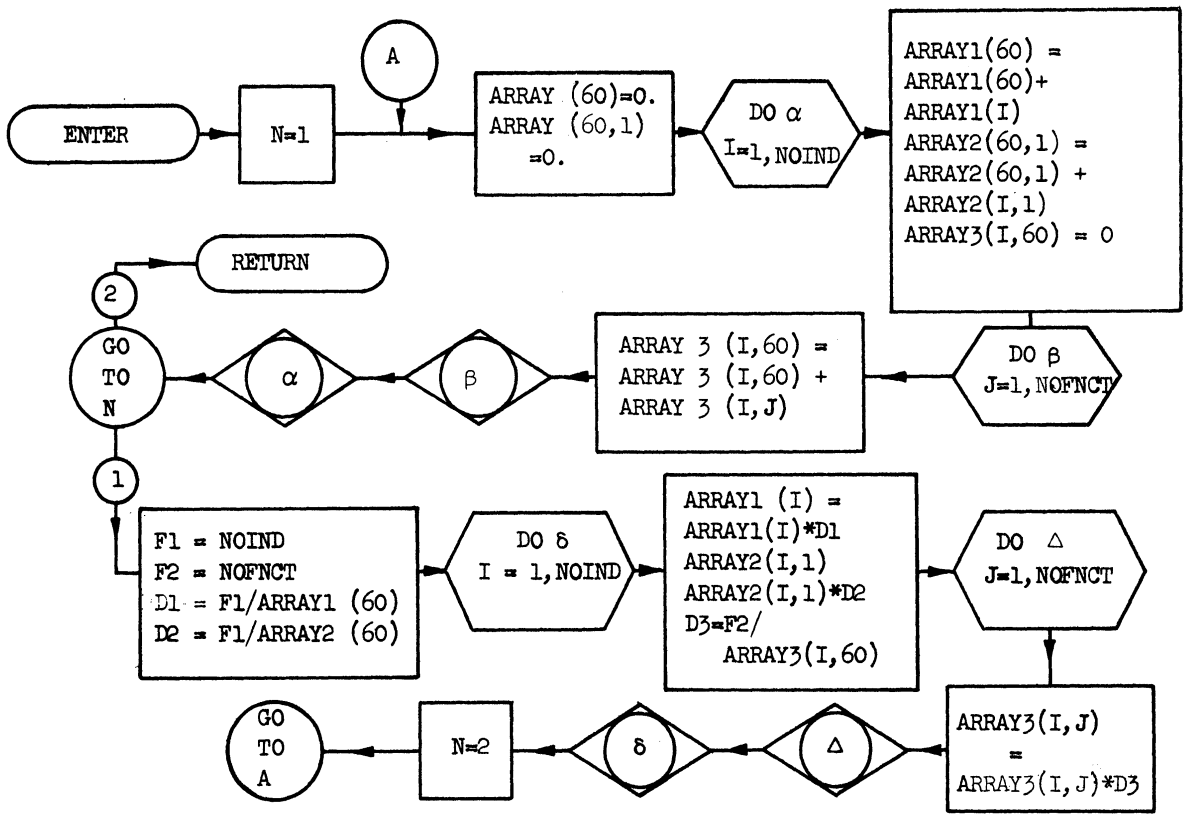


STUDENT PROGRAM (CONTINUED)

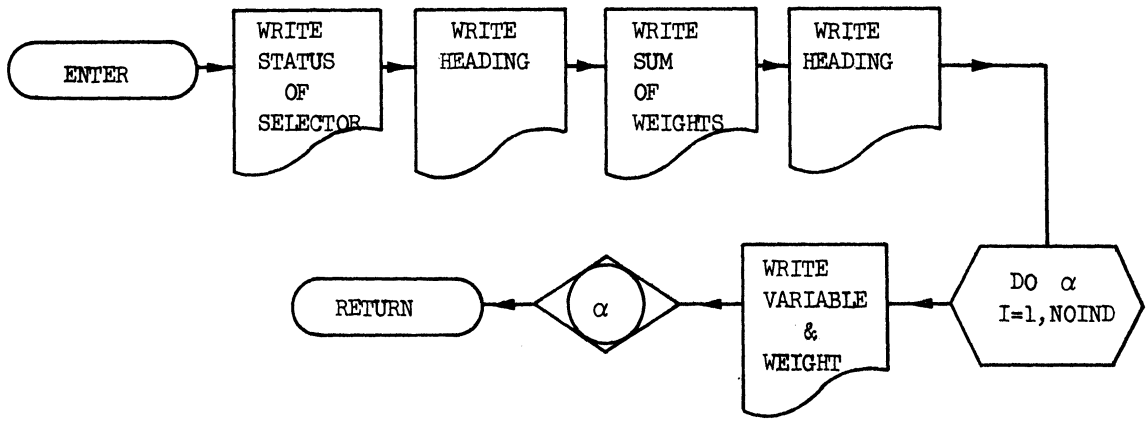




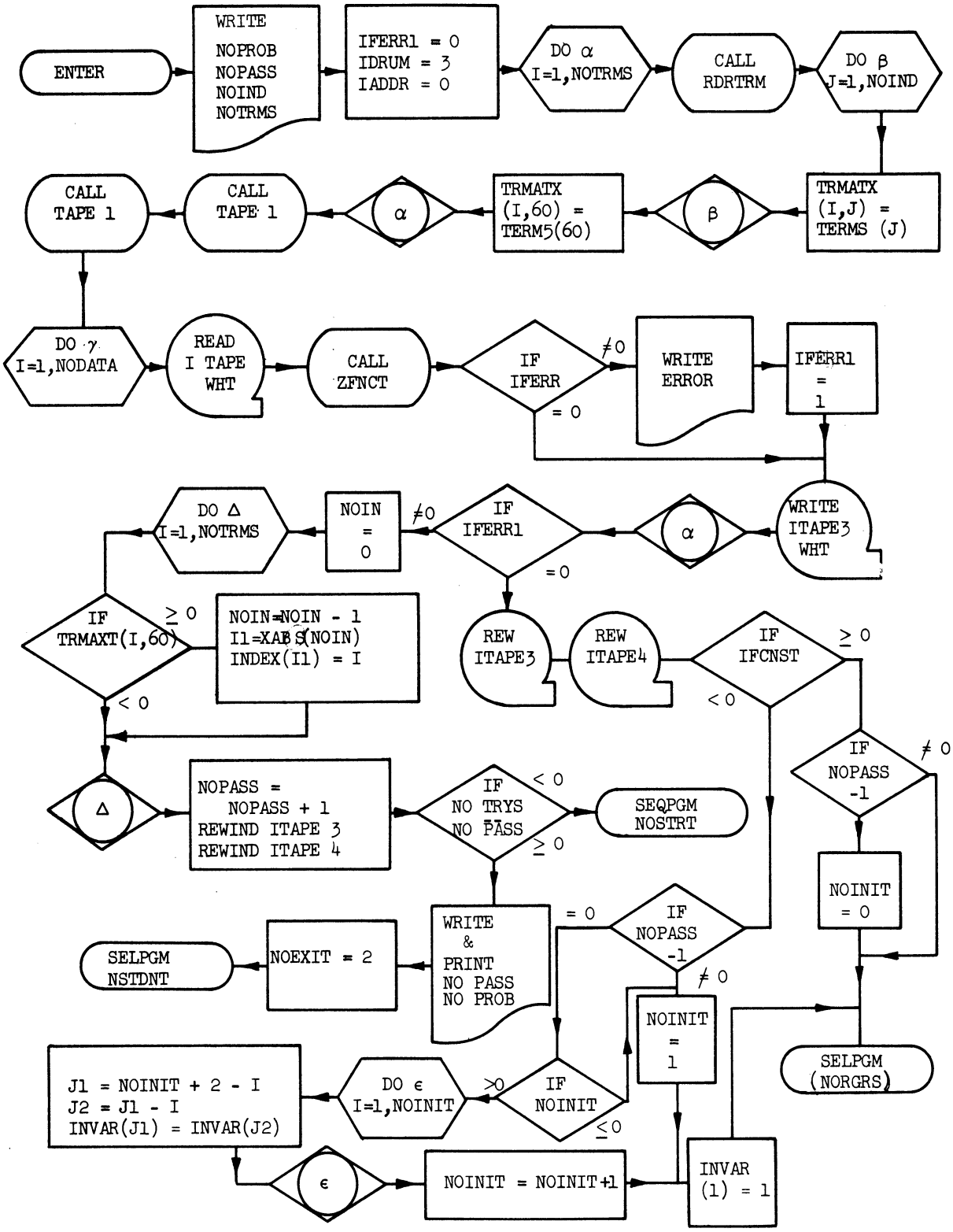
NRML


ENTER
N=1
A
ARRAY (60)=0.
ARRAY (60,1)
=0.
DO α
I=1,NOIND
ARRAY1(60) =
ARRAY1(60)+
ARRAY1(I)
ARRAY2(60,1) =
ARRAY2(60,1) +
ARRAY2(I,1)
ARRAY3(I,60) = 0
DO β
J=1,NOFNCT
ARRAY 3 (I,60) =
ARRAY 3 (I,60) +
ARRAY 3 (I,J)
α
β
GO
TO
N
2
RETURN
1
F1 = NOIND
F2 = NOFNCT
D1 = F1/ARRAY1 (60)
D2 = F1/ARRAY2 (60)
DO δ
I = 1,NOIND
ARRAY1 (I) =
ARRAY1(I)*D1
ARRAY2(I,1)
ARRAY2(I,1)*D2
D3=F2/
ARRAY3(I,60)
DO Δ
J=1,NOFNCT
ARRAY3(I,J)
=
ARRAY3(I,J)*D3
Δ
δ
N=2
GO
TO
A

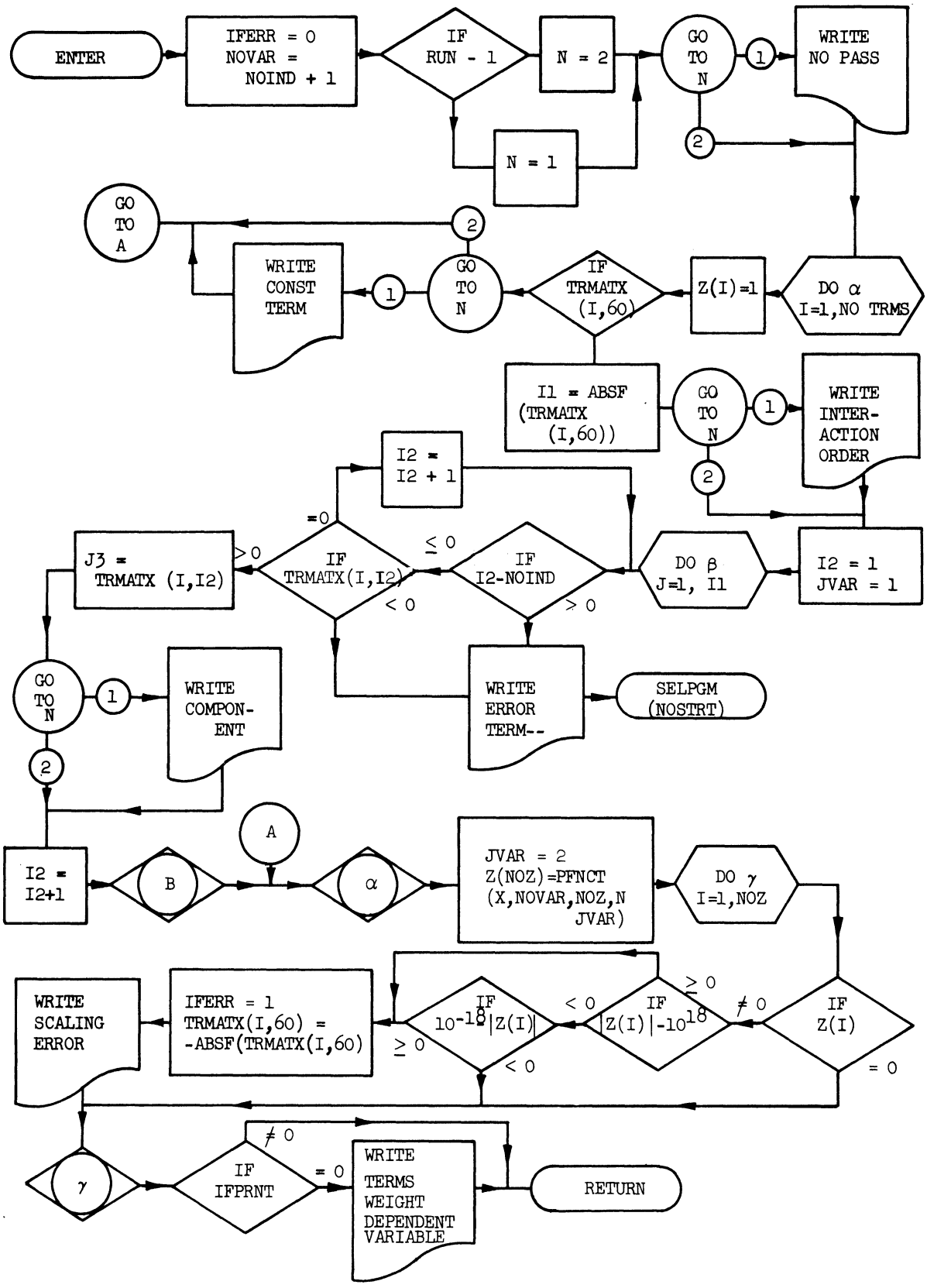
PRINT 4


ENTER
WRITE
STATUS
OF
SELECTOR
WRITE
HEADING
WRITE
SUM
OF
WEIGHTS
WRITE
HEADING
DO α
I=1,NOIND
WRITE
VARIABLE
&
WEIGHT
α
RETURN

EDITOR PROGRAM

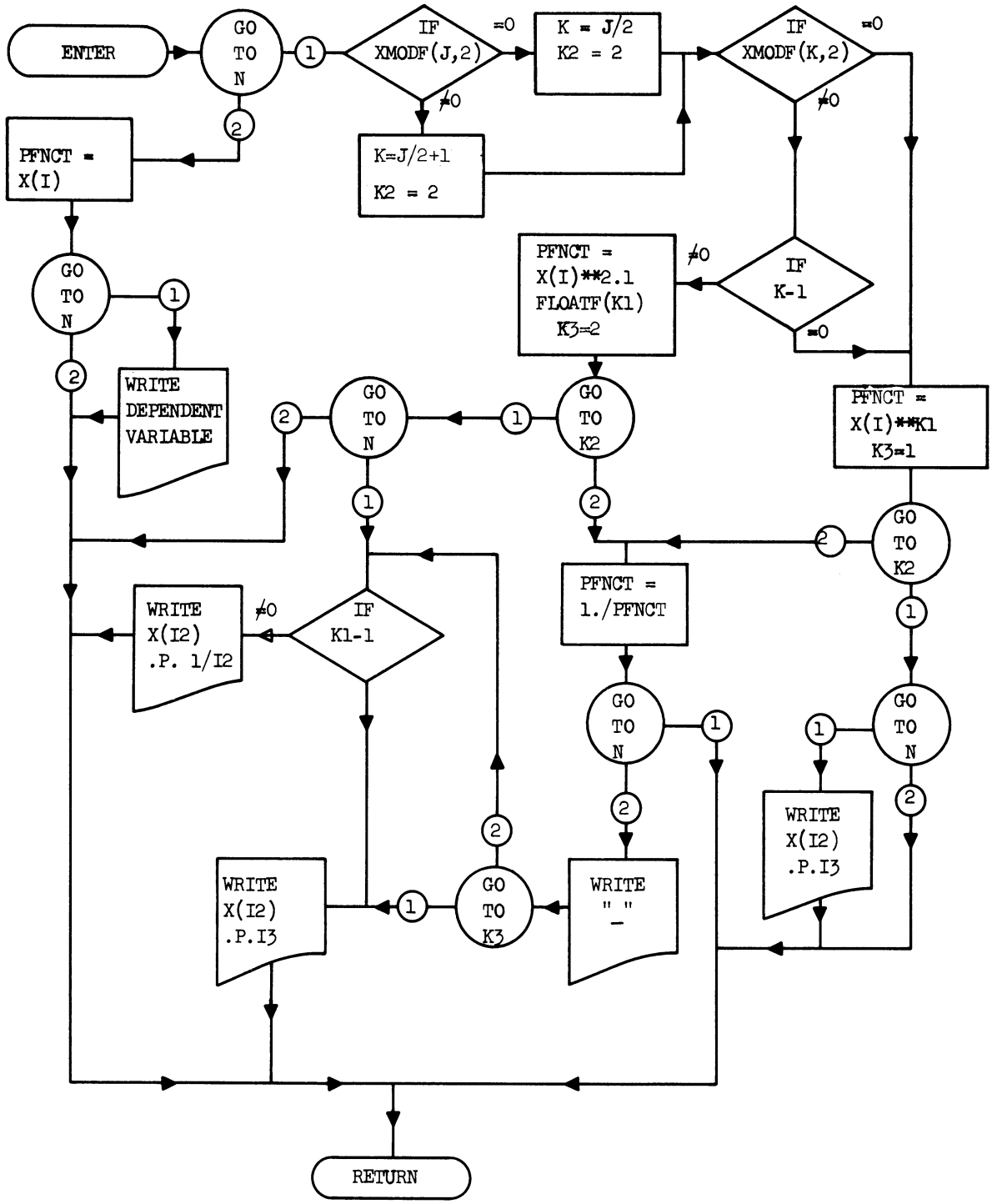


ZFNCT

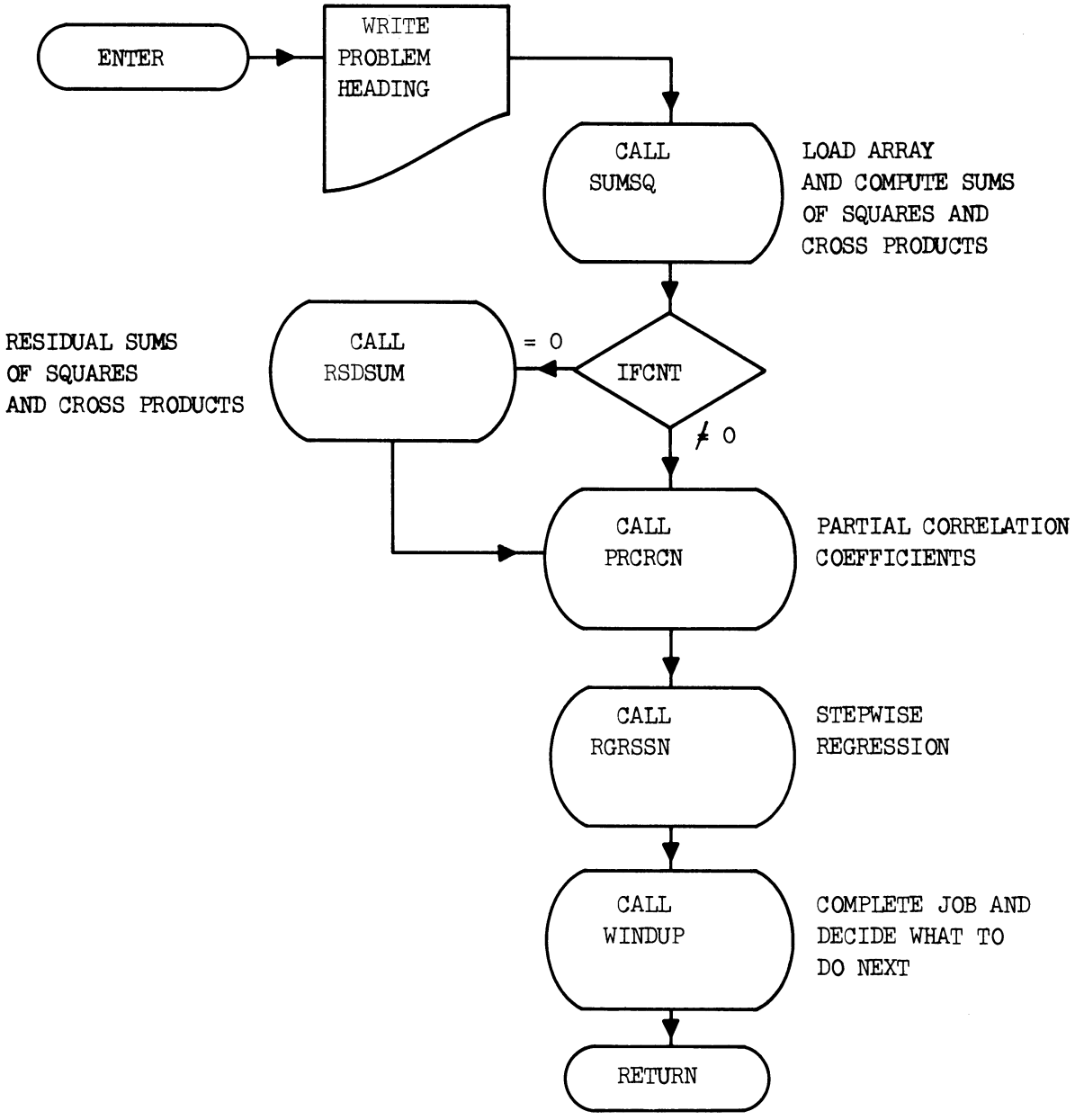

ENTER
IFERR = 0
NOVAR = NOIND + 1
IF RUN - 1
N = 2
N = 1
GO TO N
1
2
WRITE NO PASS
GO TO A
WRITE CONST TERM
IF TRMATX (I,60)
Z(I)=1
DO α I=1,NO TRMS
I1 = ABSF (TRMATX (I,60))
WRITE INTER-ACTION ORDER
I2 = I2 + 1
I2 = 1 JVAR = 1
DO β J=1, I1
IF I2-NOIND
IF TRMATX(I,I2)
J3 = TRMATX (I,I2)
WRITE ERROR TERM--
SELPGM (NOSTRT)
WRITE COMPON-ENT
A
B
α
I2 = I2+1
JVAR = 2 Z(NOZ)=PFNCT (X,NOVAR,NOZ,N JVAR)
DO γ I=1,NOZ
IF Z(I)
IF |Z(I)|-10^18
IF 10^-18|Z(I)|
IFERR = 1 TRMATX(I,60) = -ABSF(TRMATX(I,60)
WRITE SCALING ERROR
γ
IF IFPRNT
WRITE TERMS WEIGHT DEPENDENT VARIABLE
RETURN

PFNCT
FUNCTION

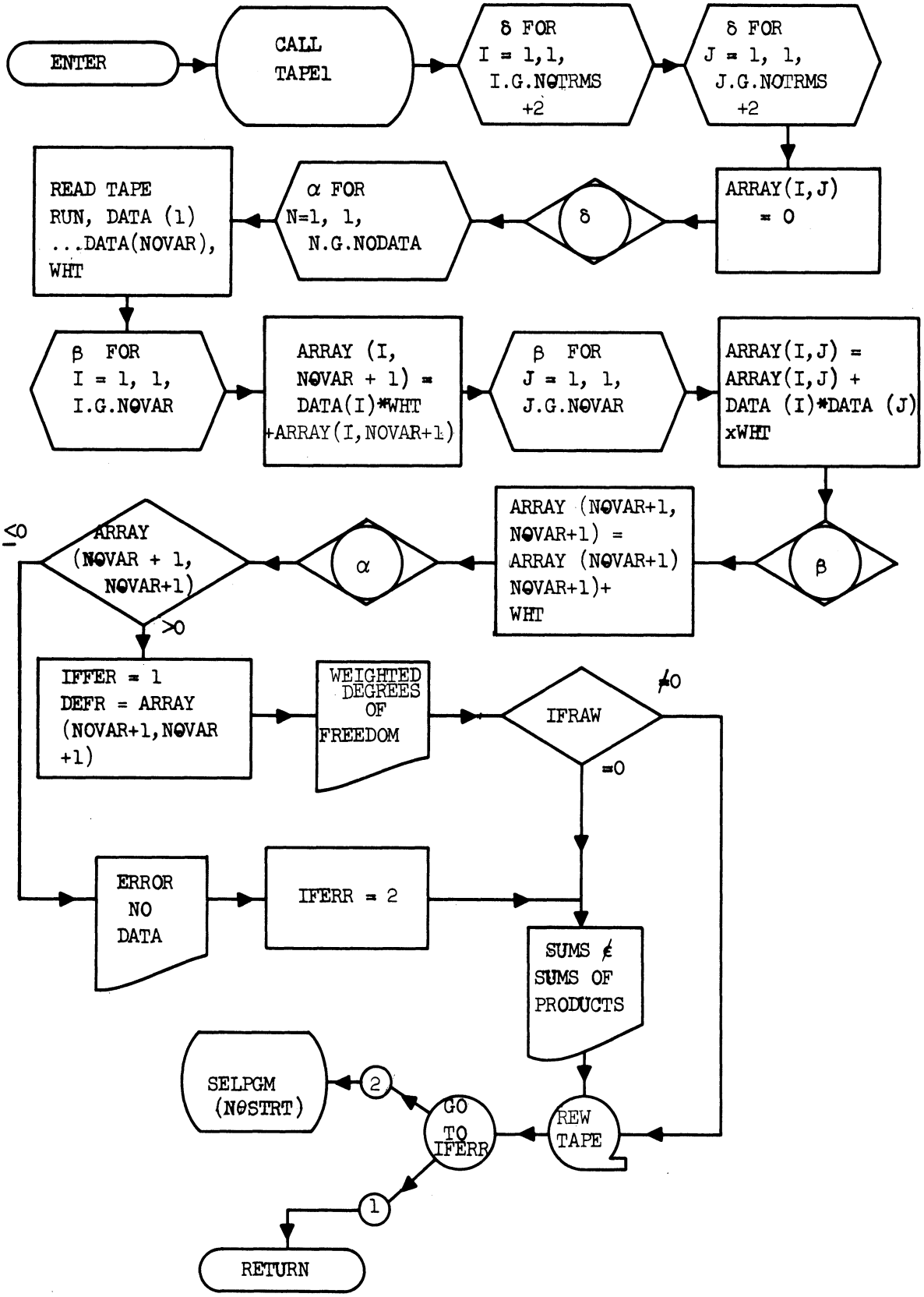
FOR INTEGER POWERS ROOTS AND RECIPROCALS.

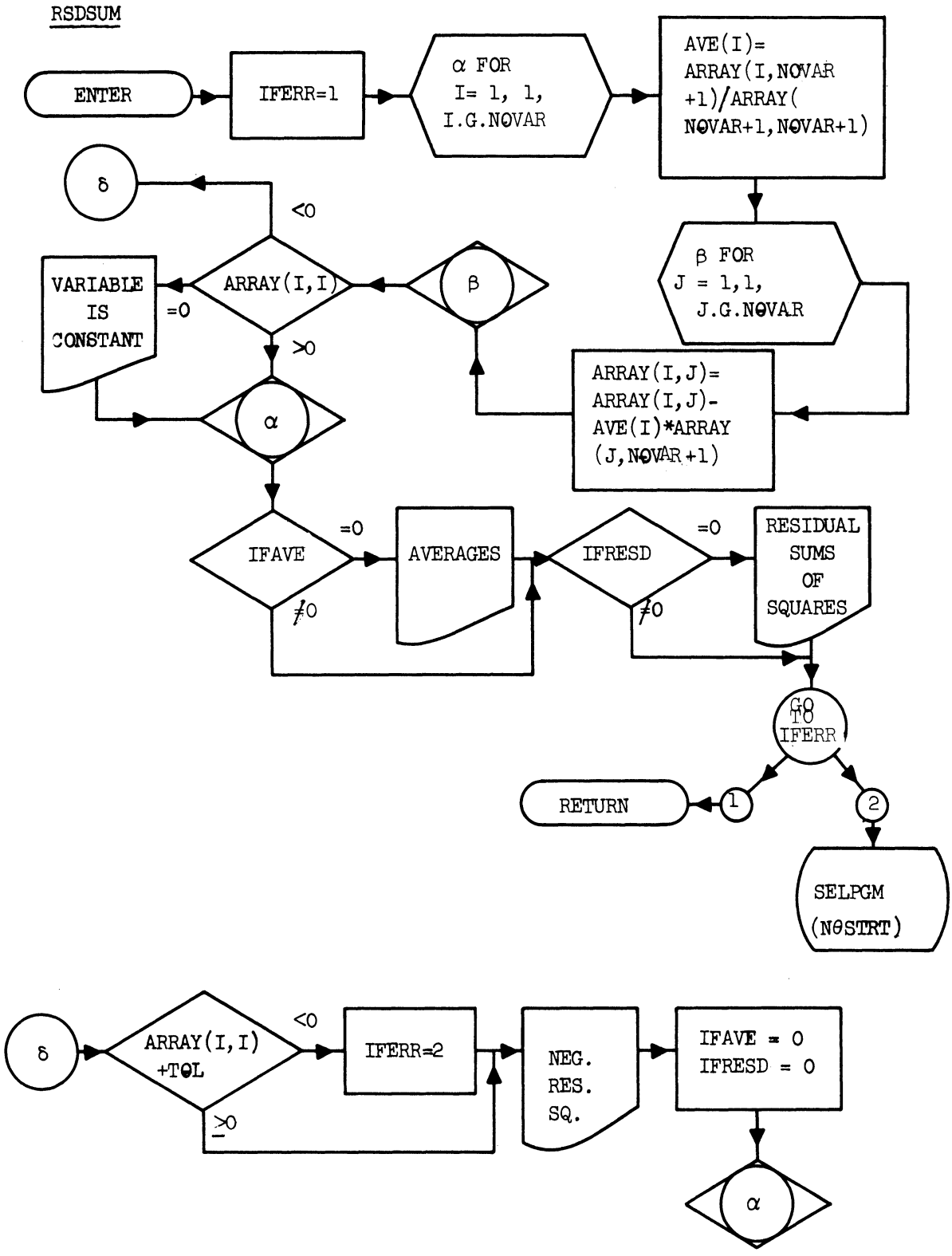


REGRESSION PROGRAM
FOR THE STEPWISE
FITTING OF DATA



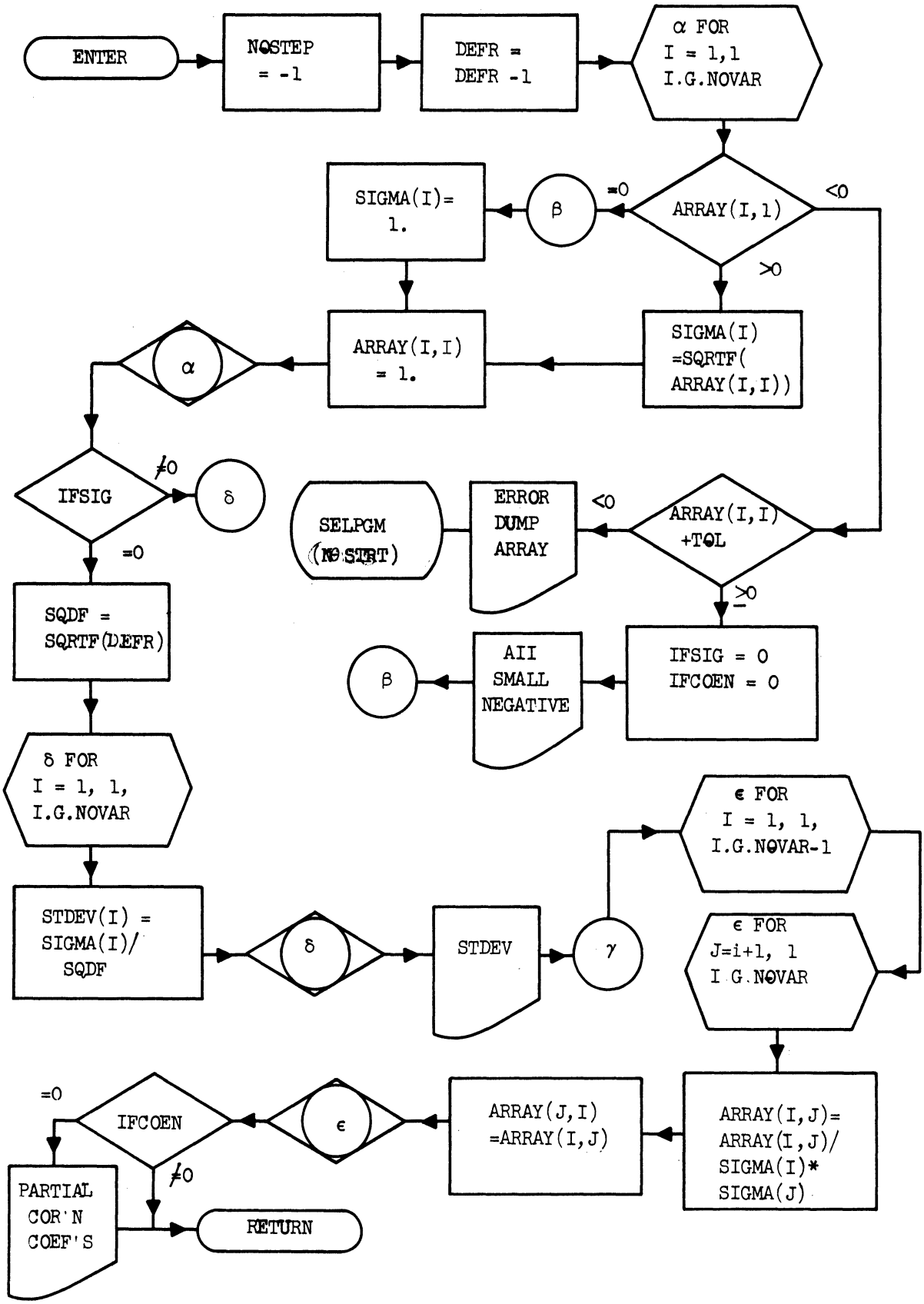
SUMSQ


ENTER
CALL TAPE1
δ FOR I = 1,1, I.G.NOTRMS +2
δ FOR J = 1, 1, J.G.NOTRMS +2
ARRAY(I,J) = 0
δ
α FOR N=1, 1, N.G.NODATA
READ TAPE RUN, DATA (1) ...DATA(NOVAR), WHT
β FOR I = 1, 1, I.G.NOVAR
ARRAY (I, NOVAR + 1) = DATA(I)*WHT +ARRAY(I,NOVAR+1)
β FOR J = 1, 1, J.G.NOVAR
ARRAY(I,J) = ARRAY(I,J) + DATA (I)*DATA (J) xWHT
β
ARRAY (NOVAR+1, NOVAR+1) = ARRAY (NOVAR+1) NOVAR+1)+ WHT
α
ARRAY (NOVAR + 1, NOVAR+1)
≤0
>0
IFFER = 1 DEFR = ARRAY (NOVAR+1,NOVAR +1)
WEIGHTED DEGREES OF FREEDOM
IFRAW
≠0
=0
ERROR NO DATA
IFERR = 2
SUMS & SUMS OF PRODUCTS
REW TAPE
GO TO IFERR
2
1
SELPGM (NOSTRT)
RETURN

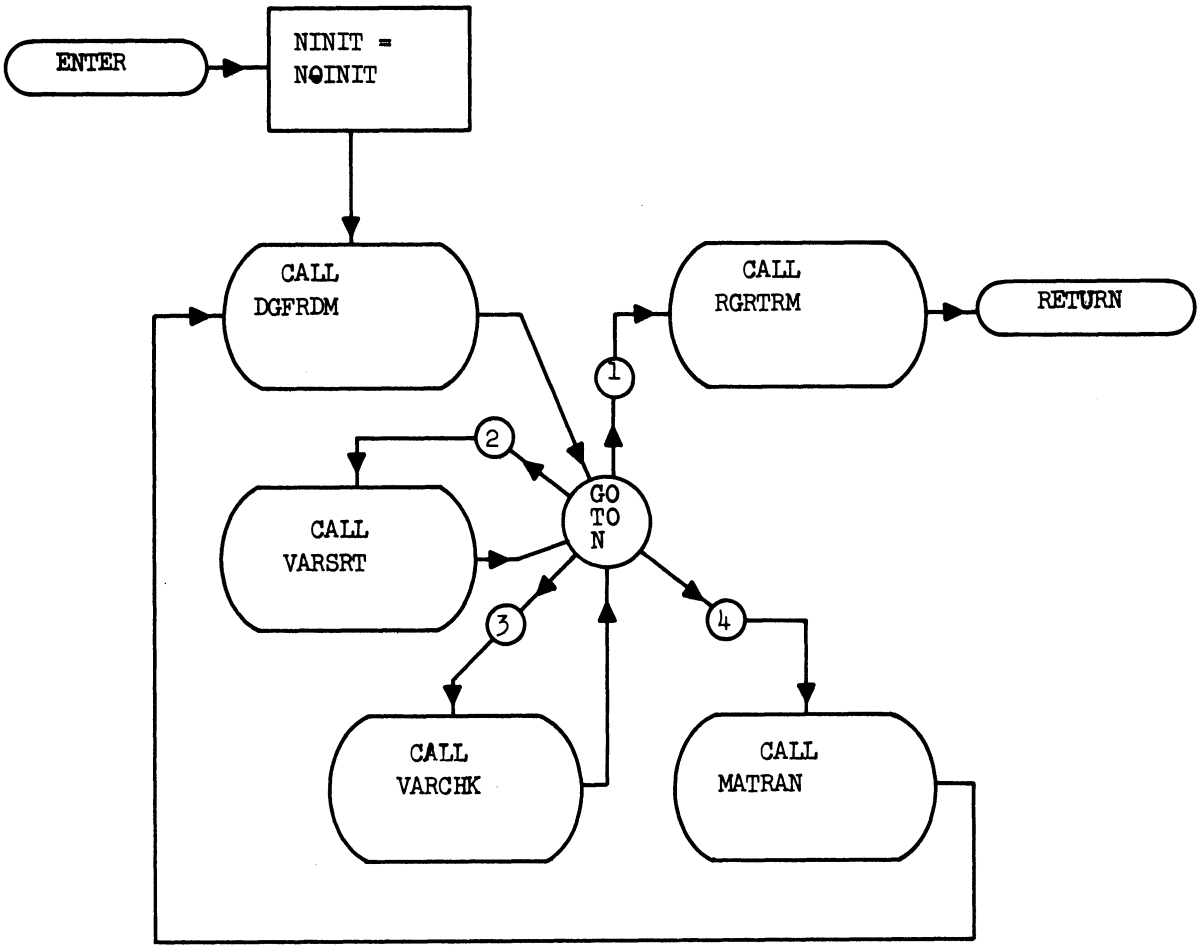

RSDSUM
ENTER
IFERR=1
α FOR
I= 1, 1,
I.G.NOVAR
AVE(I)=
ARRAY(I,NOVAR
+1)/ARRAY(
NOVAR+1,NOVAR+1)
β FOR
J = 1,1,
J.G.NOVAR
ARRAY(I,J)=
ARRAY(I,J)-
AVE(I)*ARRAY
(J,NOVAR+1)
δ
<0
ARRAY(I,I)
β
VARIABLE
IS
CONSTANT
=0
>0
α
IFAVE
=0
≠0
AVERAGES
IFRESD
=0
≠0
RESIDUAL
SUMS
OF
SQUARES
GO
TO
IFERR
RETURN
1
2
SELPGM
(NOSTRT)
δ
ARRAY(I,I)
+TOL
<0
≥0
IFERR=2
NEG.
RES.
SQ.
IFAVE = 0
IFRESD = 0
α

A-3

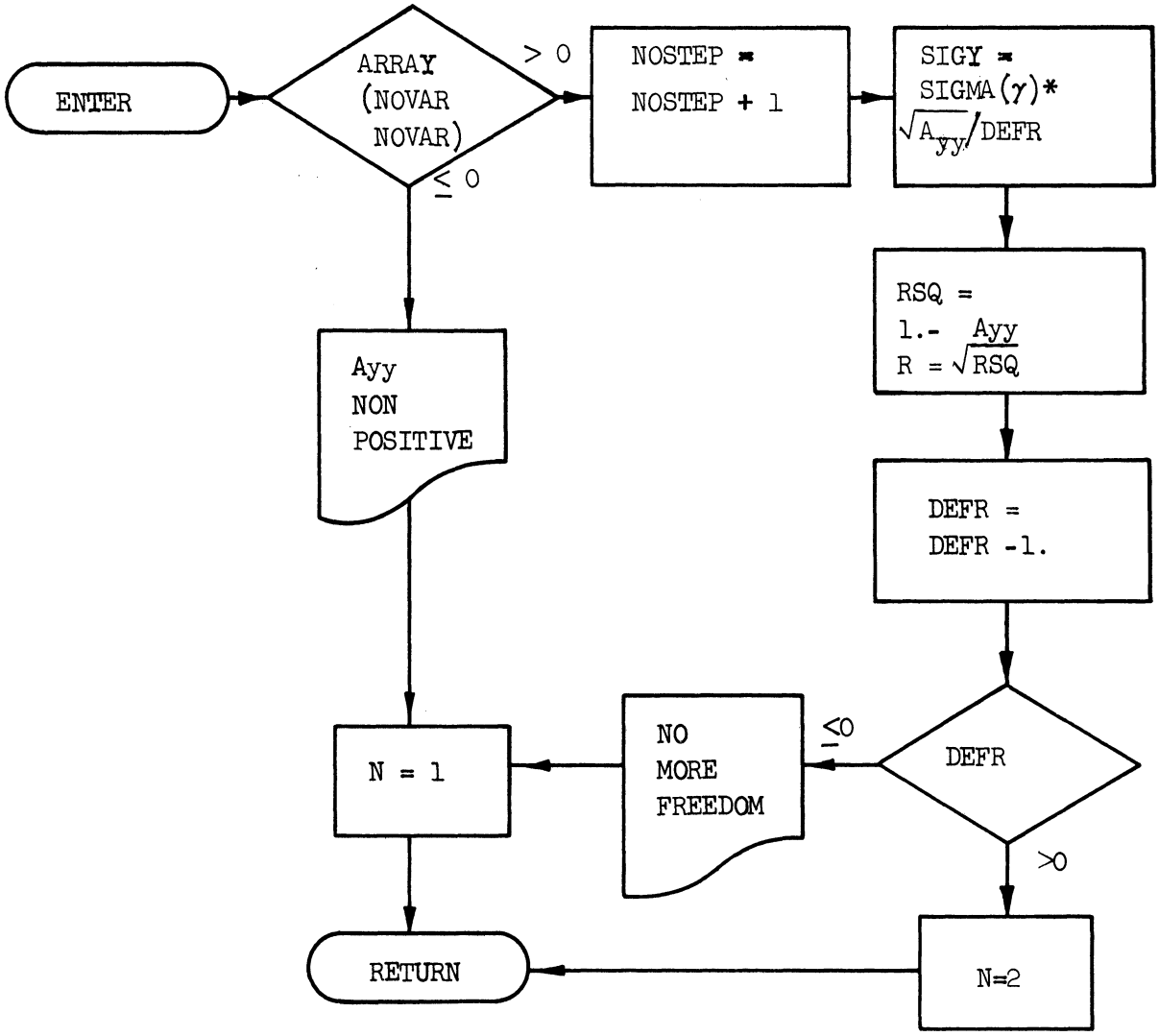
PRCRCN


ENTER
NOSTEP = -1
DEFR = DEFR -1
α FOR I = 1,1 I.G.NOVAR
ARRAY(I,1)
=0
<0
>0
β
SIGMA(I)= 1.
SIGMA(I) =SQRTF(ARRAY(I,I))
ARRAY(I,I) = 1.
α
IFSIG
≠0
δ
=0
SELPGM (NO START)
ERROR DUMP ARRAY
<0
ARRAY(I,I) +TOL
≥0
SQDF = SQRTF(DEFR)
IFSIG = 0 IFCOEN = 0
AII SMALL NEGATIVE
β
δ FOR I = 1, 1, I.G.NOVAR
ϵ FOR I = 1, 1, I.G.NOVAR-1
STDEV(I) = SIGMA(I)/ SQDF
δ
STDEV
γ
ϵ FOR J=i+1, 1 I G.NOVAR
ARRAY(I,J)= ARRAY(I,J)/ SIGMA(I)* SIGMA(J)
ARRAY(J,I) =ARRAY(I,J)
ϵ
IFCOEN
=0
≠0
PARTIAL COR'N COEF'S
RETURN

RGRSSN

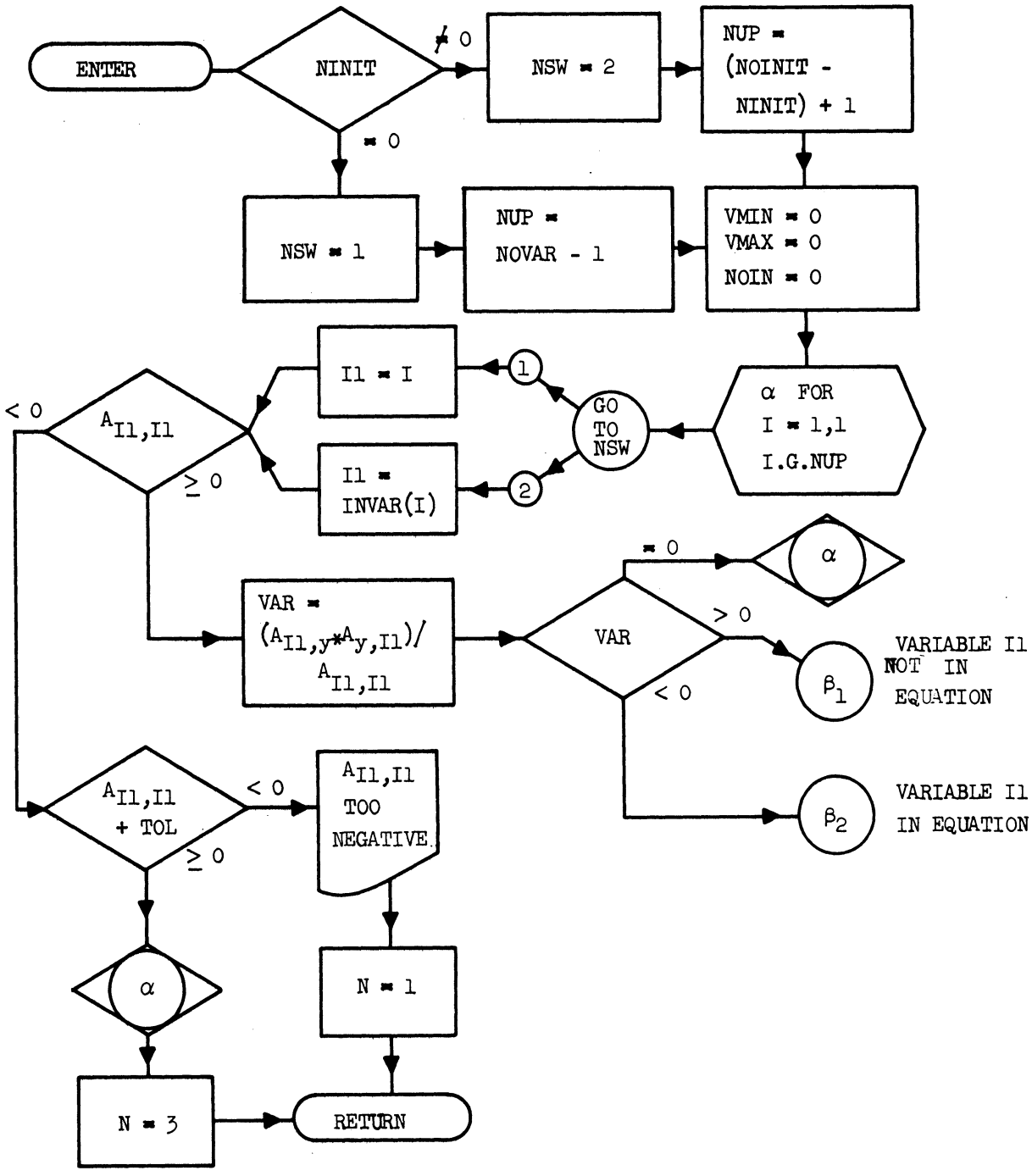

ENTER
NINIT =
NOINIT
CALL
DGFRDM
CALL
RGRTRM
RETURN
1
2
GO
TO
N
CALL
VARSRT
3
4
CALL
VARCHK
CALL
MATRAN

DGFRDM

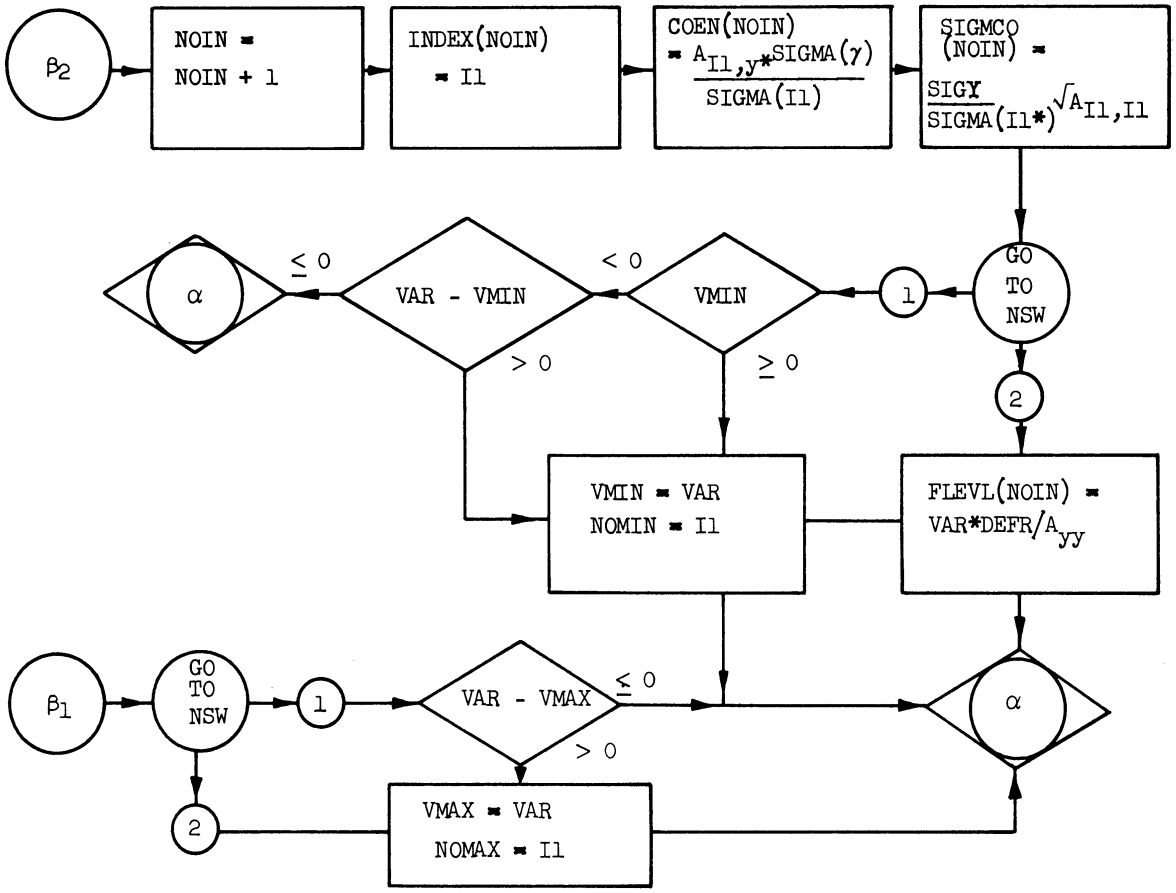

ENTER
ARRAY (NOVAR NOVAR)
> 0
≤ 0
NOSTEP = NOSTEP + 1
SIGY = SIGMA(γ)* √A_yy/DEFR
RSQ = 1.- Ayy
R = √RSQ
DEFR = DEFR -1.
DEFR
≤0
>0
NO MORE FREEDOM
Ayy NON POSITIVE
N = 1
N=2
RETURN

COMBINED WITH VINSRT TO ALLOW INSERTATION
OF TERMS, OR STANDARD OPERATION, AS DESIRED.

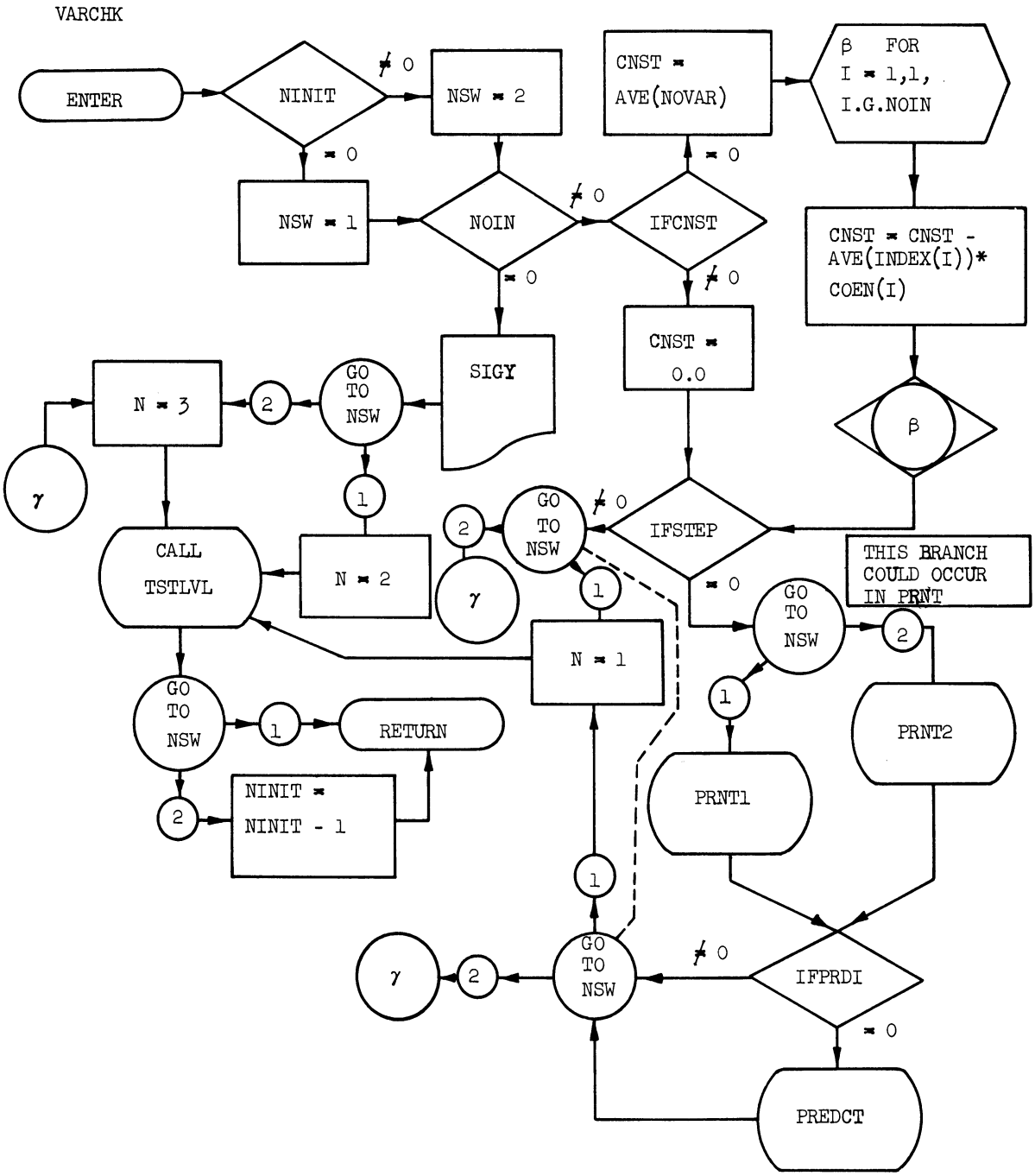
VARSRT

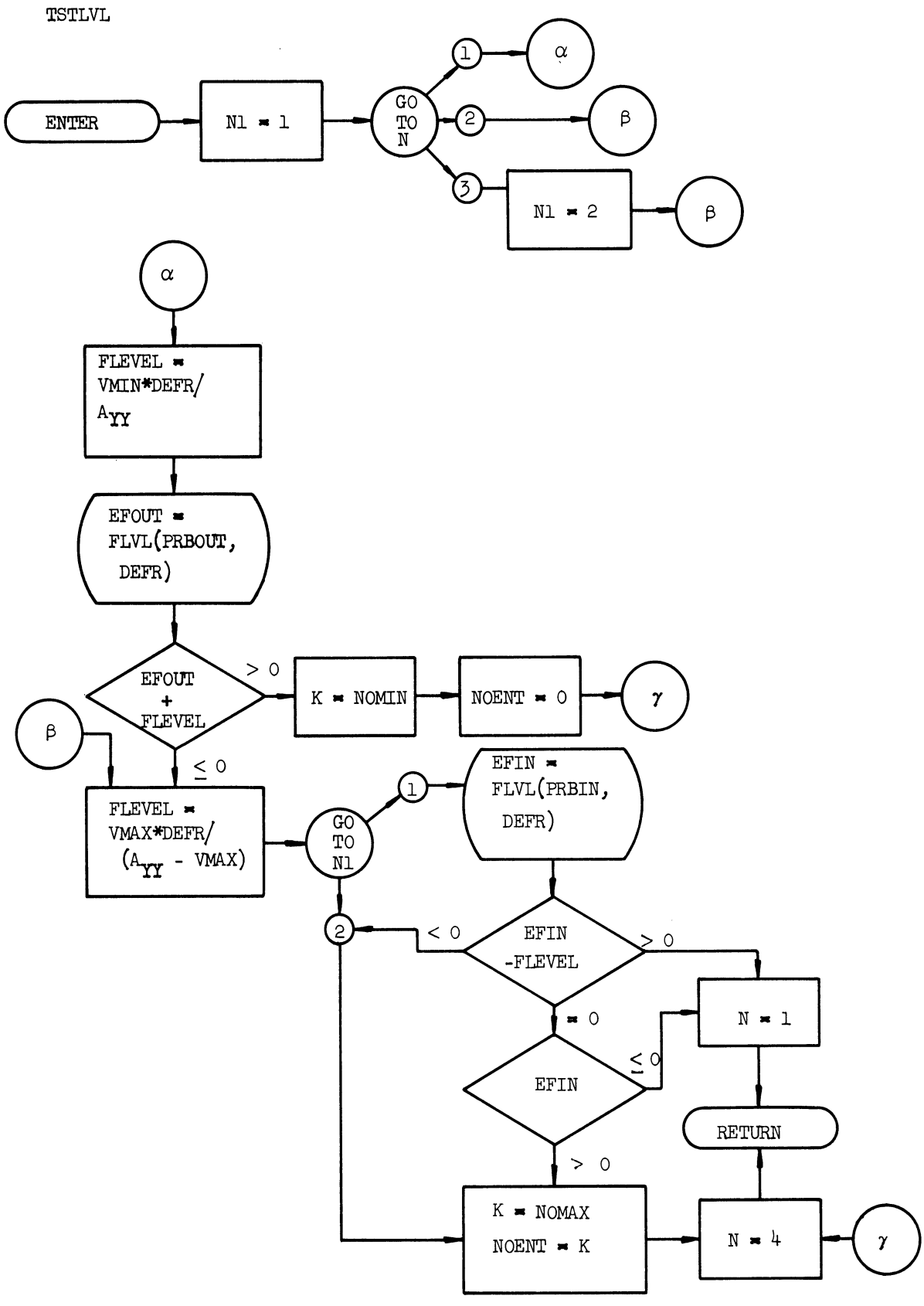


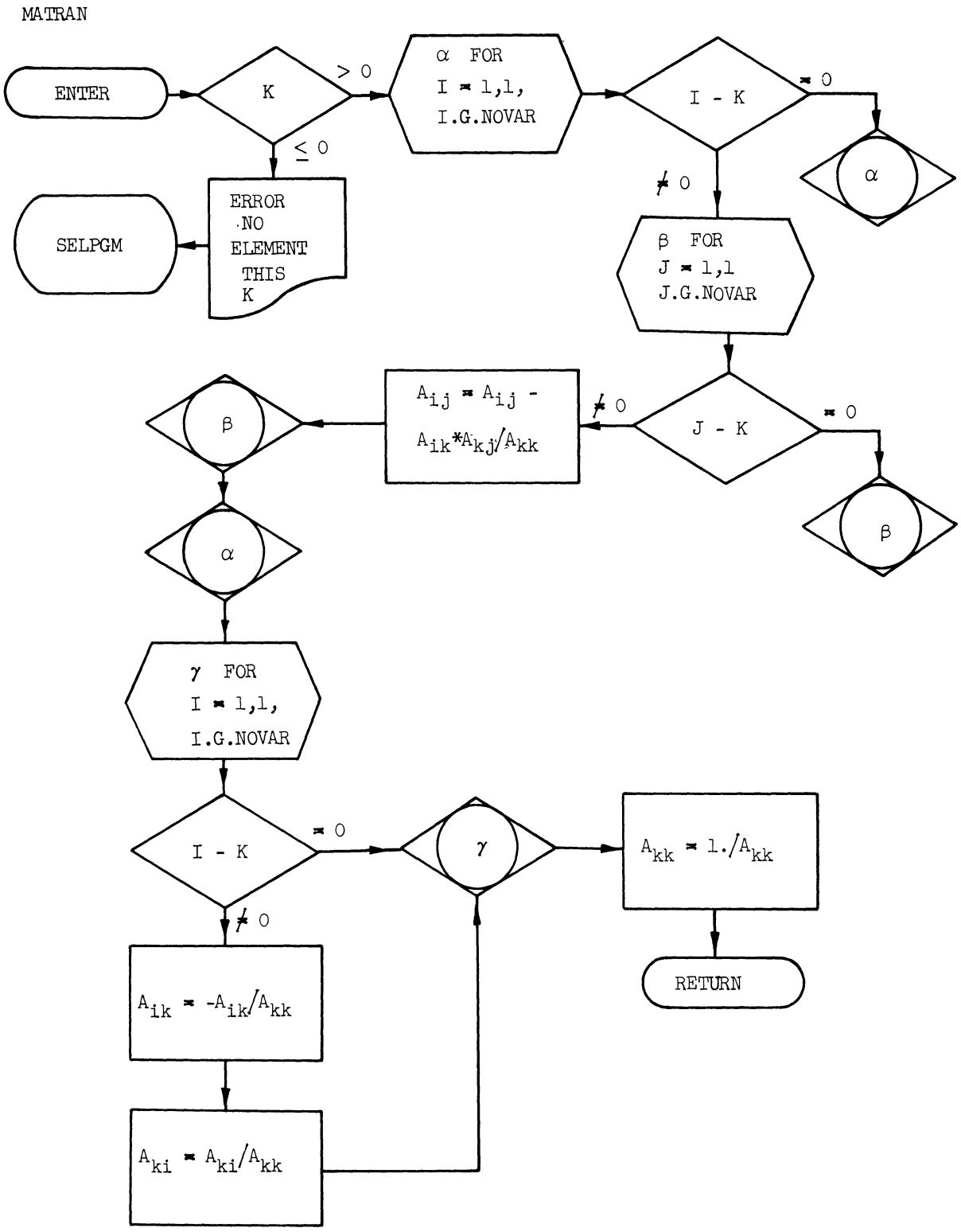
VARSRT (CONT'D)



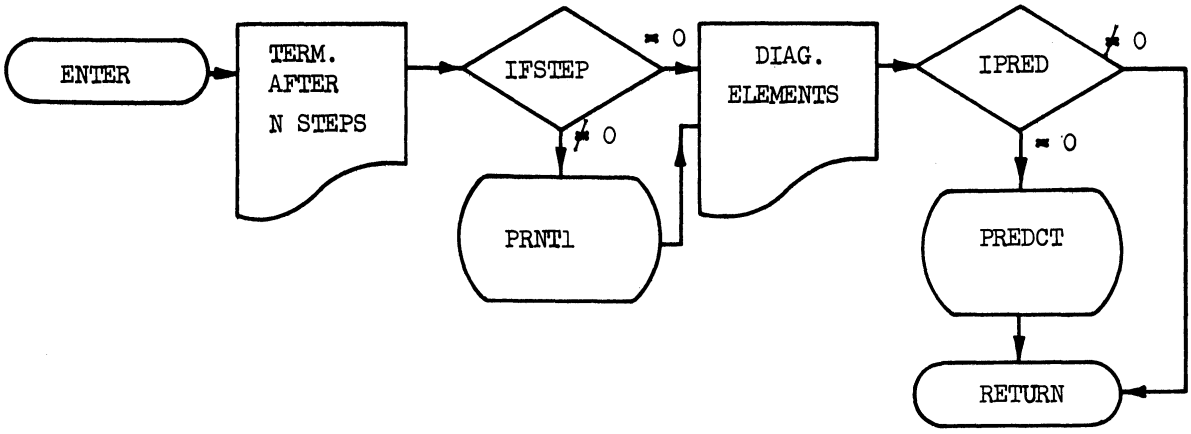
COMBINED WITH VINCHK TO ALLOW INSERTION
OF TERMS, OR STANDARD OPERATION, AS DESIRED.



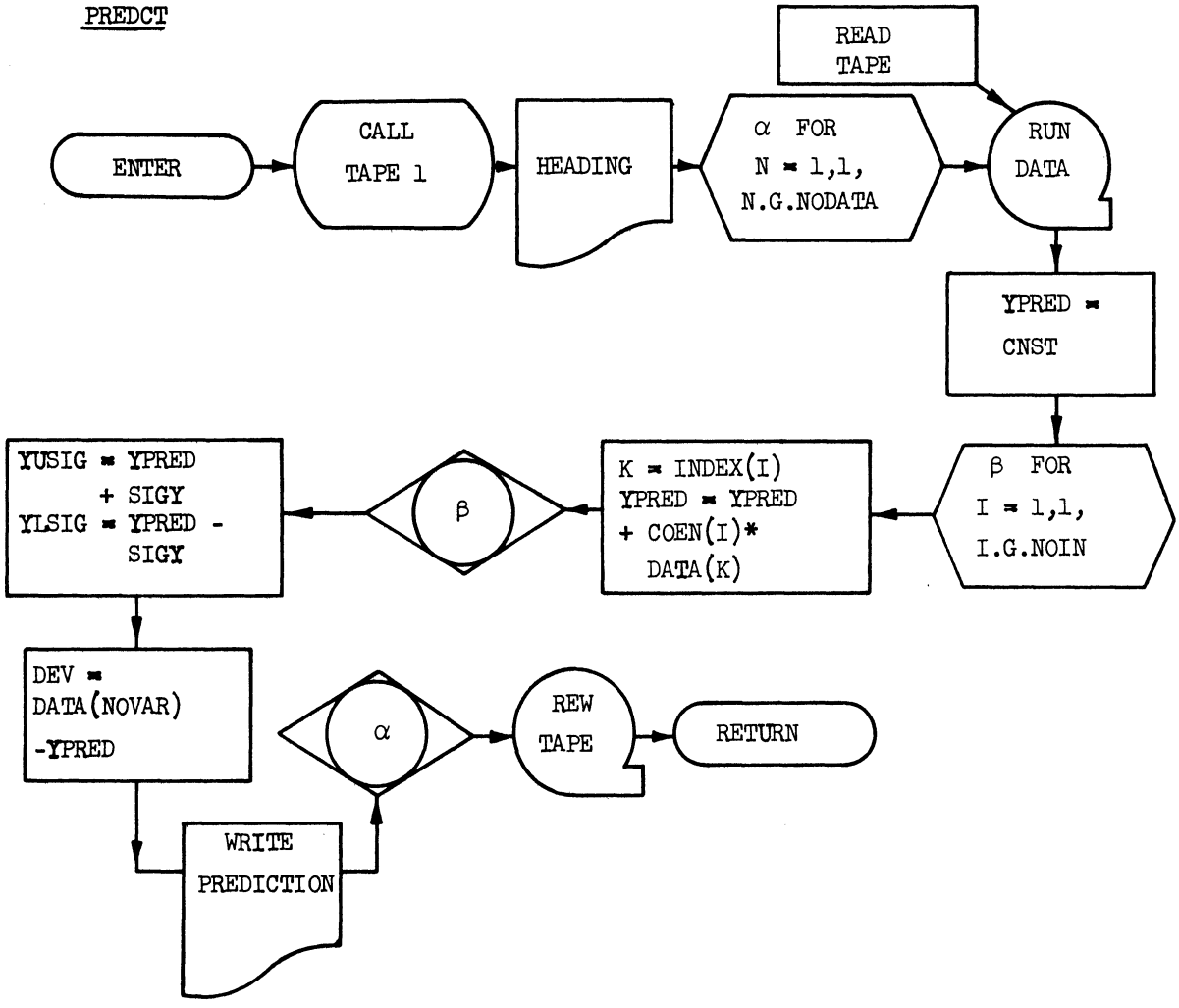

TSTLVL
ENTER
N1 = 1
GO TO N
1
2
3
α
β
N1 = 2
β
α
FLEVEL = VMIN*DEFR/ A_{YY}
EFOUT = FLVL(PRBOUT, DEFR)
EFOUT + FLEVEL
> 0
K = NOMIN
NOENT = 0
γ
β
≤ 0
FLEVEL = VMAX*DEFR/ (A_{YY} - VMAX)
GO TO N1
1
EFIN = FLVL(PRBIN, DEFR)
2
< 0
EFIN -FLEVEL
> 0
= 0
N = 1
EFIN
≤ 0
RETURN
> 0
K = NOMAX
NOENT = K
N = 4
γ


MATRAN
ENTER
K
> 0
≤ 0
α FOR
I = 1,1,
I.G.NOVAR
I - K
= 0
≠ 0
α
ERROR
NO
ELEMENT
THIS
K
SELPGM
β FOR
J = 1,1
J.G.NOVAR
J - K
≠ 0
= 0
A_ij = A_ij -
A_ik*A_kj/A_kk
β
α
γ FOR
I = 1,1,
I.G.NOVAR
I - K
= 0
≠ 0
γ
A_kk = 1./A_kk
RETURN
A_ik = -A_ik/A_kk
A_ki = A_ki/A_kk

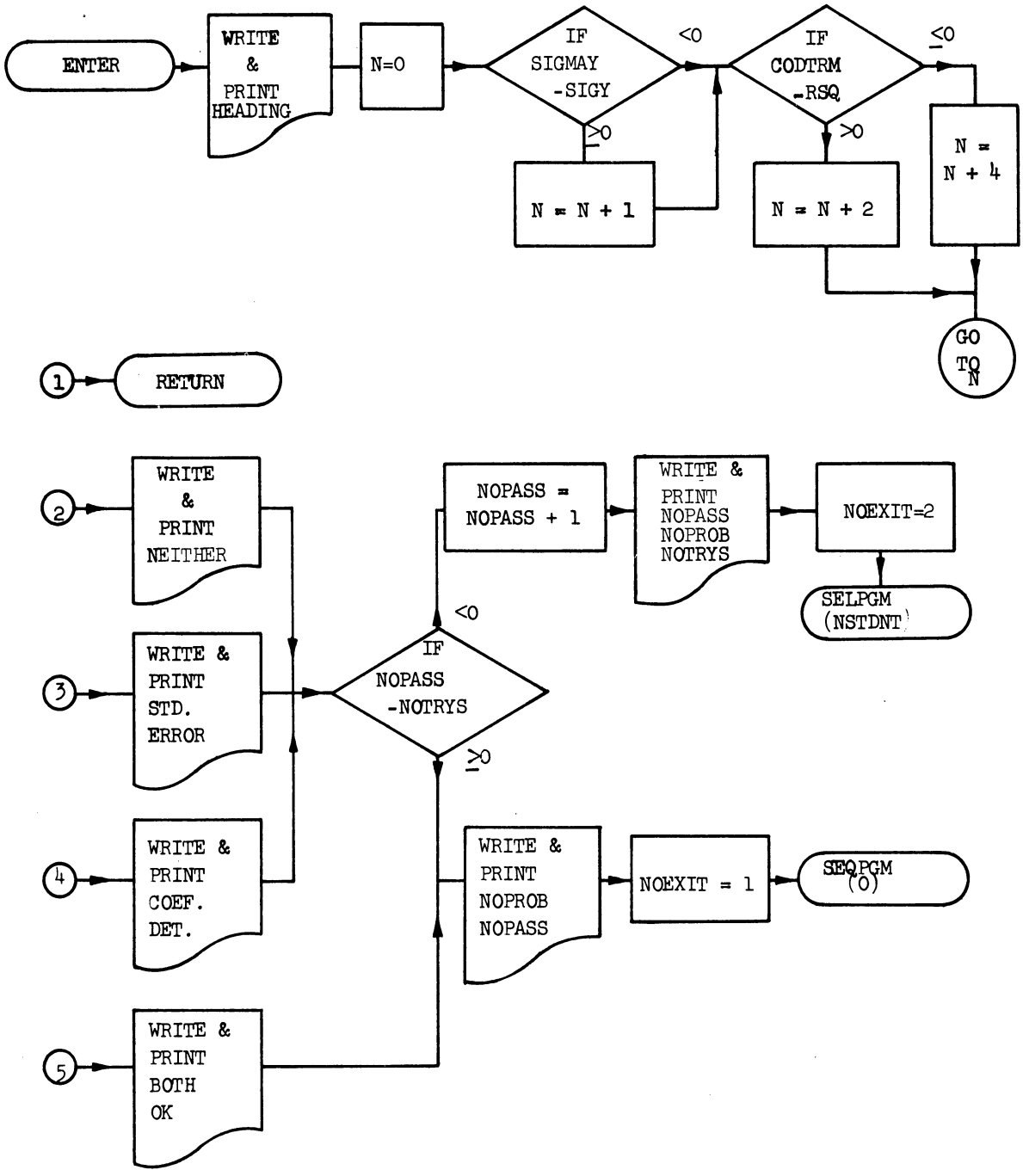
RGRTRM



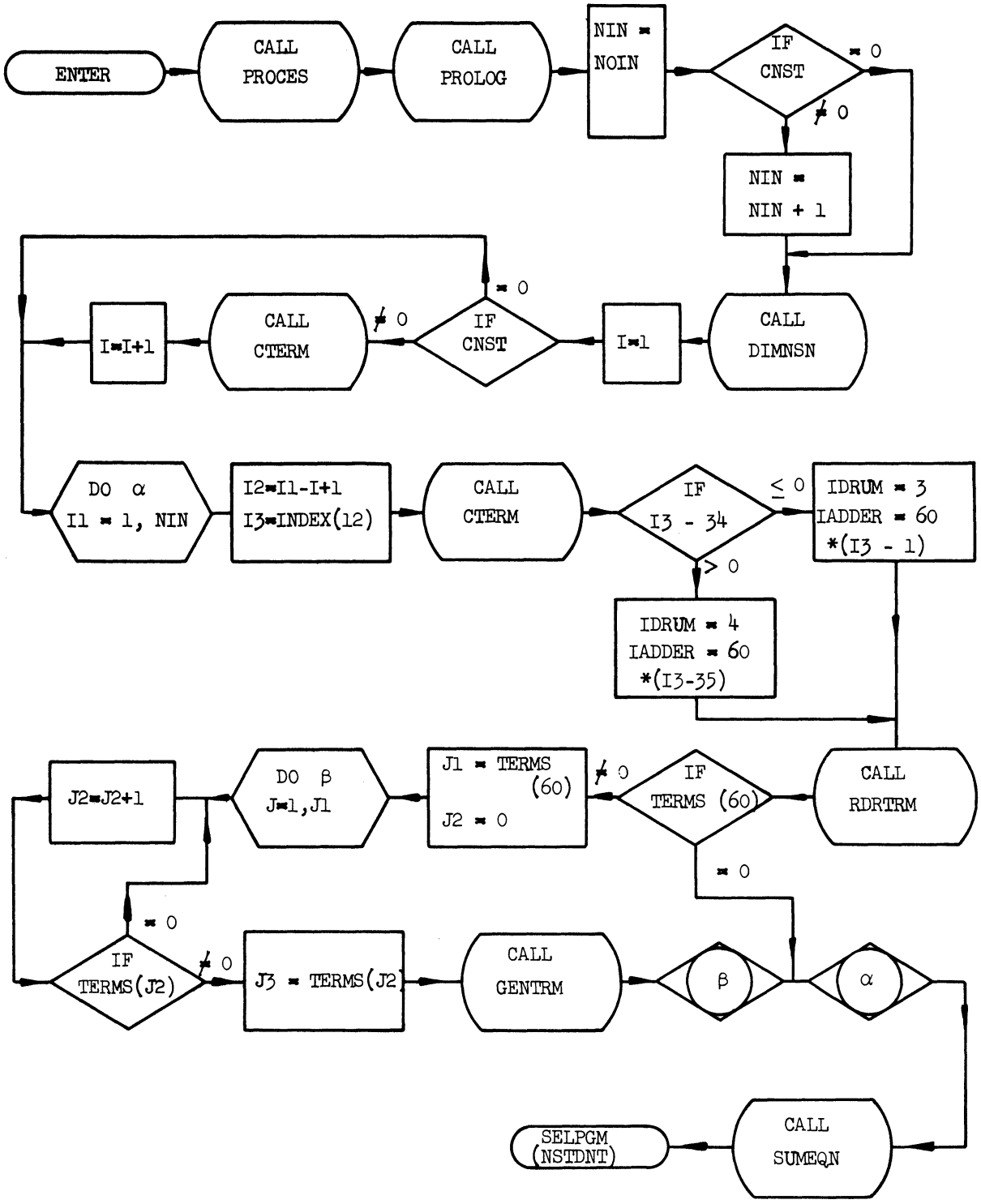
PREDCT



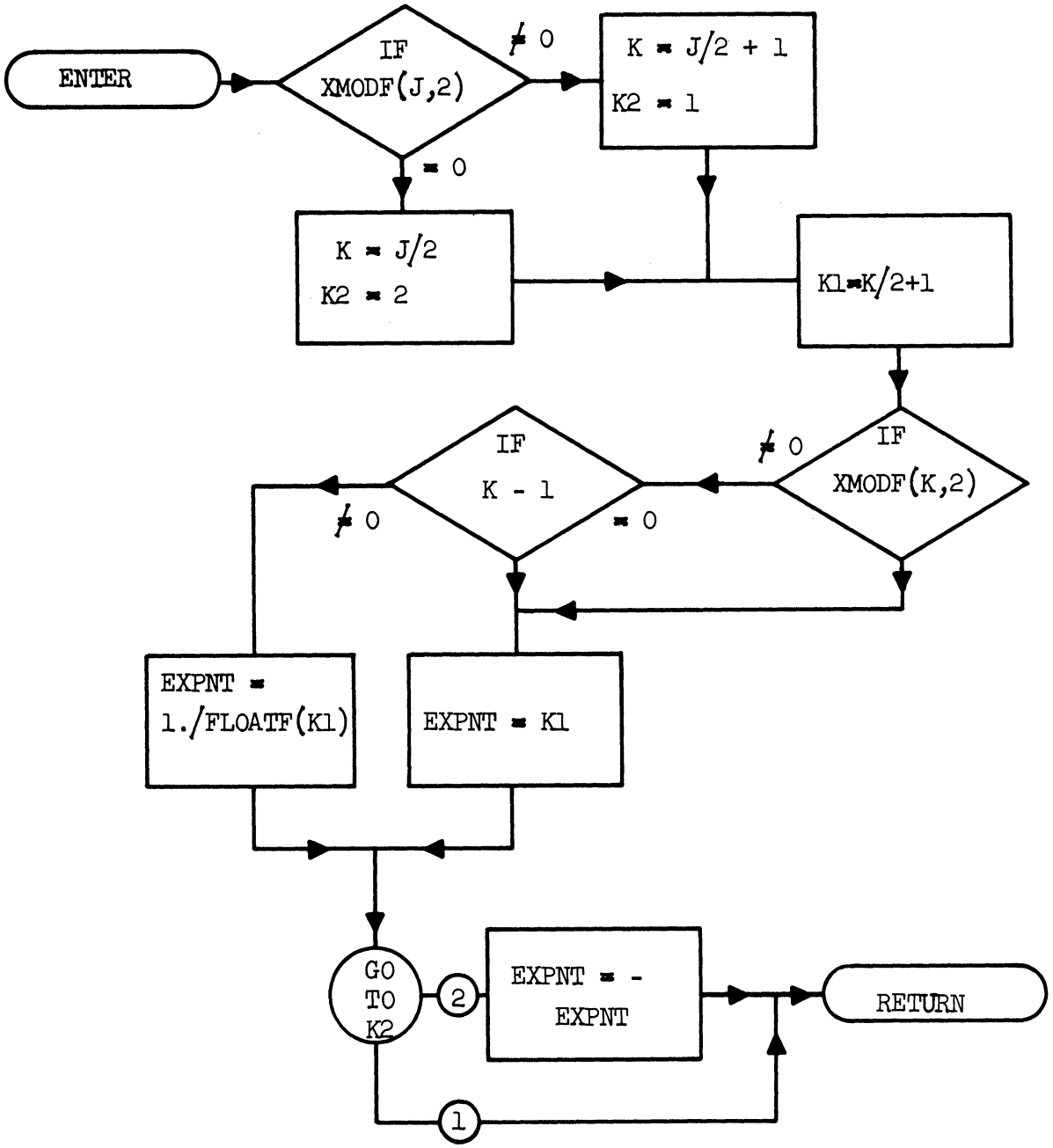
WINDUP


ENTER
WRITE & PRINT HEADING
N=0
IF SIGMAY -SIGY
<0
≥0
IF CODTRM -RSQ
≤0
>0
N = N + 1
N = N + 2
N = N + 4
GO TO N
1
RETURN
2
WRITE & PRINT NEITHER
3
WRITE & PRINT STD. ERROR
4
WRITE & PRINT COEF. DET.
5
WRITE & PRINT BOTH OK
IF NOPASS -NOTRYS
<0
≥0
NOPASS = NOPASS + 1
WRITE & PRINT NOPASS NOPROB NOTRYS
NOEXIT=2
SELPGM (NSTDNT)
WRITE & PRINT NOPROB NOPASS
NOEXIT = 1
SELPGM (0)

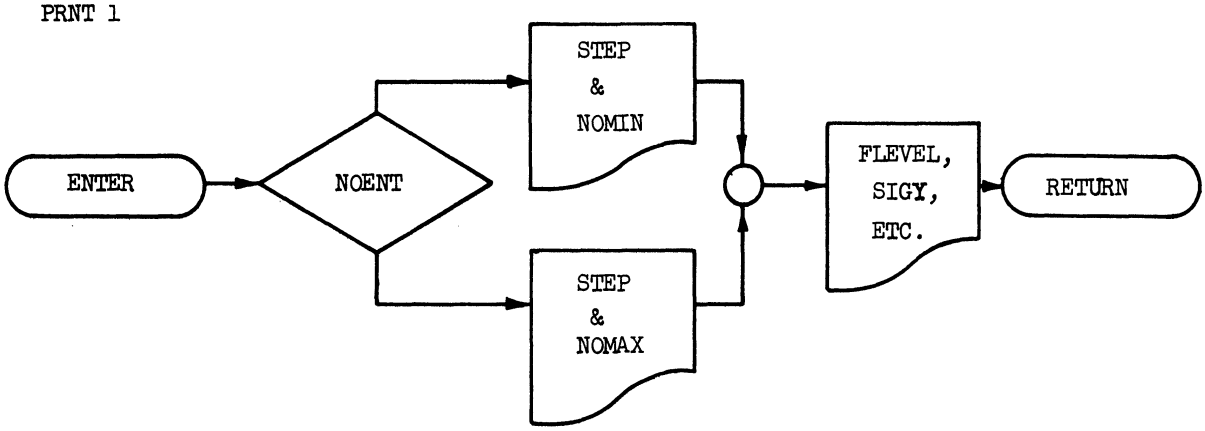
MAD OR FORTRAN STATEMENT GENERATING PROGRAM



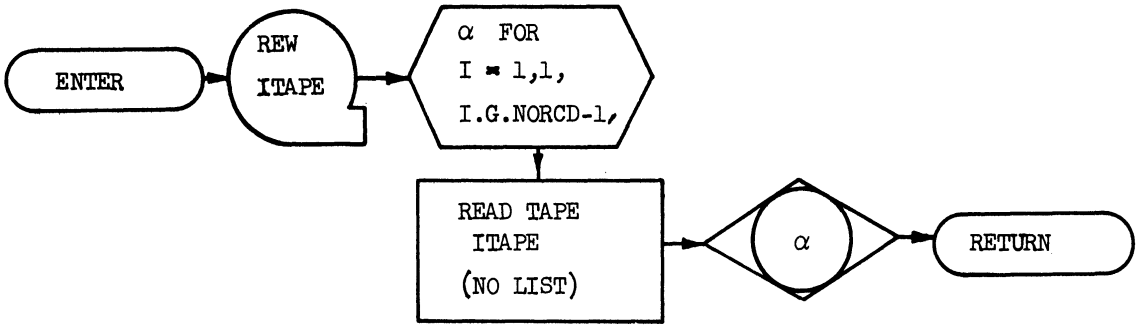
EXPNT


ENTER
IF
XMODF(J,2)
≠ 0
K = J/2 + 1
K2 = 1
= 0
K = J/2
K2 = 2
K1=K/2+1
IF
XMODF(K,2)
≠ 0
IF
K - 1
≠ 0
= 0
EXPNT =
1./FLOATF(K1)
EXPNT = K1
GO
TO
K2
2
EXPNT = -
EXPNT
RETURN
1

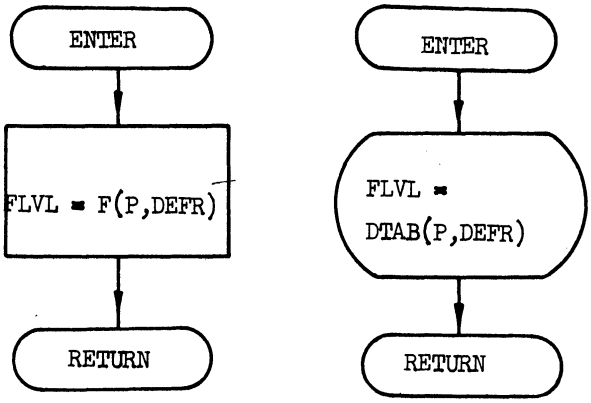
PRNT 1



TAPE 1



FLVL



TWO VERSIONS AVAILABLE

1) Functional Representation

2) Table Interpolation

COMMENTS ON THE SYSTEM SIMULATOR FLOW DIAGRAMS

The flow diagrams for the System Simulator may be more easily followed through the combined use of the comments presented here and the references given to the main text. Where page numbers are parenthesized the page refers to associated explanation in the main text of the paper.

The System Simulator Core 1 diagrammed on page 132 is charged with the initial translation of the source program presented by the user. (51) The first task is the initialization of the status of the machine including the determination of the condition of the tape units and the blanking of core and drums and the setting of switches and counters. Beginning then at RESET1 the Simulator reads card after card into memory one at a time. As each card is read into memory, the card information is scanned character by character using the procedures indicated in the scope of LOOP. Blanks are ignored and illegal punctuation is detected. Legal punctuation together with the action of the counter HOLCNT is used to establish the type of statement being scanned and the action to be taken. Whenever more than 6 characters have been found without finding legal punctuation, the Simulator anticipates that a Declaration of some kind may be at hand. By using indices K1 and K for the statement label array S, it is possible to transfer directly to the appropriate section of the program for any declaration and to the section of the declaration analysis appropriate for the punctuation found.

Because of the limitations of storage, the additional analysis required for an Element Description is done by ELDES in core 2. Otherwise,

the declarations for Connections, Input Parameters, Desired Results, Synonyms, Function Substitutions and the data following them can be treated entirely within core 1. The diagrams on page 134 indicate the settings of the switches for the processing of the various declarations. The Connection section on page 135 may be entered through S(6), S(7), S(8), or S(9) depending upon the punctuation encountered in the scanning loop for the counter index K. Within the section, the Boolean variables CTO, CEL, CAT, CAID, and so on are to retain the structure of the statement being treated and thus to direct the processing of the statement. As might be suspected from the mnemonic symbol names, CTO is associated with the connective TO (23), CEL is associated with Element name, CEID with Element Identifier, CAT with Attachment name and CAID with Attachment Identifier. The diagram on page 135 indicates the treatment of the special unary and binary elements. (24) The subroutine CON CK on page 139 is for the purpose of establishing the consistency of the Connection matrix and to assure in this way that analysis of ambiguously defined systems will not be attempted.

The Synonym section on page 136 analyzes the form of the synonym encountered (42) and saves the result for later elimination of synonyms from the Connection matrix. In a similiar but simpler way, the Input Parameters and Desired Results section on page 137 retains its information in core for later use while the Function Substitution section on page 138 records its information on magnetic tape.

The consistency check subroutine CON CK on page 139 is used to insure the detection of ambiguity in the Connection matrix. If more than one connection statement has been given for a given identified attachment point CON CK determines whether the associated attachment is such that

the connection is a duplicate of an earlier connection. If this is true, the subroutine removes the duplicate statement and compresses the matrix. Otherwise, the connection is ambiguous and the error is reported.

The Simulator Core 2 known as ELDES is the Section of program used whenever an Element Description is encountered. See page 140. After saving a section of the data on tape to provide space for the processing of the Element Description and after initializing counters and switches, the analysis of the Element Description begins. Since several assertions are recognized within the scope of the Element Description, a scanning occurs of each card to extract the information required. A somewhat different structure for handling the punctuation is employed. This structure is somewhat more convenient for the Element Description processing because of the periodic occurrence of the M.A.D. statements which must be passed untouched to the Library tape. Only the encounter of the Declaration DESCRIPTION FINISHED can return the program to core 1 after completing the processing at S(4) on page 143.

When the end of input data is detected by the occurrence of an End Of File mark or by the declaration NEXT SET OF DATA, the processing goes to the Setup Core 3. In the diagrams beginning on page 145, the Setup Core eliminates synonyms (SYN ELM subroutine), constructs the Boolean parameter words (51)(IPR ELM subroutine), constructs the indirect addressing lists for the connection matrix (52) (SS ORDR subroutine) and positions the tapes for further processing. The details of these subroutines may be followed by reference to the associated text as indicated. Several other small utility routines such as the tape mover routine TAPMV,

and the check out routines TAPEIN, TST DMP (Test Dump) and EL TP PR (Element Tape Printer) are shown for completeness.

The Desired Result Reduction Program (core 4) begins on page 154. After initialization, the search for parameters for which program must be generated begins. First, all parameters are removed that be so treated without introducing computation. This may be done by matching the requested parameter with an Input Parameter or by taking advantage of the Broad Scope concept. (27,56) If the request remains after the execution of EXT CHK (Scope Check) the associative memory (61) is set to receive entries for every pertinent Statement Collection. Since tape movement is very time consuming the flow diagram on page 155 computes the shortest path for the tape movement. The diagram on 156 analyzes each collection of statements to determine its utility in yielding the desired results and checks to be certain that any special conditions are also satisfied. The diagram on page 157 selects the collection to be used in the program and inserts the collection in the program (INSERT subroutine) and removes the yielded results and adjusts the connection matrix (REMOVE subroutine). The section on page 158 details the probabilistic selection mechanism. (60) Finally, the diagram on page 159 details the testing procedure for the completion of the program generation and the repetition of the process if required. The routines shown from page 160 through page 168 detail the various subroutines indicated in the main core 4.

When the program has been completed by core 4, the remaining task is the production of the object program itself. (63) The generation is accomplished in three core loads of program. The first of these

generates the Prologue section. (47) As a part of this, the unique parameter code is constructed by PCODE on page 170. (35) The routine PRLOG accomplishes the actual prologue generation using the primary scan PSCAN and the secondary scan SSCAN. Since multiple copies of some statements must be generated for input-output requirements the use of two scanning routines to analyze an input buffer BFR and generate an output buffer BFR1 proves to very effective in saving generation time. The routines PSCAN on page 172 and SSCAN and SSCAN1 on 173 along with the smaller routines for output (OUTPT1), saving the Epilogue section (STORER), the word insertion routine (INSRT), the input-output statement generator (PSUB1), the general statement generator (PSUB2)(33,40), the parameter name generator (PNAME), the continuation card routine (SKIP) and the parameter dictionary output routine (PLIST) are structured to be included wherever they are required in any of the program generation cores. The routine FSUB is charged with the determination of the contents of the $\overline{0}$ symbols (33) and produces either floating statements labels or function substitutions depending on the double asterisk symbol. Depending on the decision the routine FSUB1 or FSUB2 on page 189 will be used for function substitutions or the routine LABELS on page 190 will be used for floating statement labels.

The program generation section PROGEN on page 191 makes use of the common routines mentioned above as indicated in the diagram. The generation must invert the order of the program found by the Desired Result Reduction Program. (63) The section also checks to eliminate any identical collections that may have been selected by the Desired Result

Reduction Program. Finally, upon completion of the Program Generation, the Epilogue section is called upon to provide the final cards required by the program to transfer control back to the Prologue for testing and completion of the simulation. (47)

COMMENTS ON THE STEPWISE REGRESSION FLOW DIAGRAMS

The flow diagrams for the Stepwise Regression may be more easily followed through the combined use of the comments presented here and the references given to the main text. Where page numbers are parenthesized the page refers to associated explanation in the main text of the paper.

The Starter program shown on page 198 is charged with the responsibility of entering and storing all of the control parameters and data and accumulated learning for each problem. The initial section may also save the accumulated learning and terms from an earlier problem depending upon the test of NOEXIT. A non-zero NOEXIT corresponds to an unsuccessful execution of the previous problem so it is then necessary to retain these arrays for later restarting of the problem. The subroutine PARAM brings all of the control parameters into storage. DATA IN reads in and saves on magnetic tape all of the raw data supplied for the problem. Next, depending upon the parameter IFTRWT, the accumulated learning is either read in from an earlier trial through RDTRWT or initialized to equal probability by CMTRWT. The various possible types of analysis (101) are initialized by the program following the test of the parameter IFCNST. The parameter NOINT controls the input of suggested terms from the supplied data deck. Finally READ 1 calls in and saves the desired name for the subroutine to be generated upon completion of a successful analysis. If enough terms have been given in the data, the program transfers control directly to the EDITOR core. Otherwise, the second section of the starter program is selected to generate enough terms to fill the allowed regression matrix.

The second section of the Starter program conducts a selection of terms to fill the regression matrix by using the accumulated learning as it has been initialized by the first section. The routine PK TRM selects each new term by generating three term selection parameters (122). TERM IN then inserts the term in the set of trial terms or returns for additional attempts by PK TRM if the term selected should happen to be identical to any previously selected and entered term.

The routines PARAM, DATA IN, CMTRWT, RDTRWT and the routine TERM IN shown on pages 199 through 201 execute their tasks in the straightforward manner shown. TER MIN is a "skeleton" routine (a routine that consists of a sequence of calls upon other routines) calling first upon TRM CHK to verify the uniqueness of the term under consideration and then calling upon ENTRM for the entry of the term if it is unique. TRM CHK on page 202 checks the uniqueness of the term by searching the list of previously entered terms as it has been built up on the magnetic drum.

The selection mechanism for the program selected terms in contained in the diagrams for PK TRM, PICKV, PICKE and PICKS. The skeleton for the selection process is the routine PK TRM which must choose the interaction order using PICKV, the variables to be used in the interaction using PICKS and the function of the chosen variable using PICKE within PICKS. The probabilistic selection mechanism (60,61) is the basis of all three of the routines. Since PICKS must select the number of variables specified by PICKV and since this total could include most or all of allowed variables, the mechanism is modified in this case to reduce the set of possible variables after each selection. In this way every trial will produce a unique new variable.

The Student program (so termed because the learning mechanism is located in this section) on pages 205 and 206 carries out the simple learning process. After arranging to carry forward all successful terms from the previous trial, the student program grades the previously used selection mechanism using the routine GRADER. The one exception to the grading occurs when the previous trial failed due to an impending over or under flow of the floating point data. In this case, only the faulty term is graded. When the grading is completed, the selection mechanism is normalized so that the mean probability is unity. Finally, the terms for the next trial are chosen in the same manner as was done in the second section of the starter program. If the selection of the student program followed the successful completion of an analysis, the student program also retains the accumulated learning for future use.

The routine GRADER on page 206 utilizes the "half-life" concept in rewarding or penalizing the selection mechanism. (105,122) This method preserves the positive probability of a term even after repeated failure. This, in turn, insures that every possible term receives some consideration. The routines NRML and PRINT 4 are shown on page 207. These routines carry out the normalization of the accumulated learning matrix and the printing of the status of the matrix.

The EDITOR program on page 208 processes the raw observation data to form the terms chosen by the student program or given by the user. If the user should wish to supply a function that is not one of the set allowed by the "standard" ZFNCT and PFNCT routines, the following modifications should be made:

1) ZFNCT controls the interaction of the functions of the variables. If anything other than a cross-product interaction is desired, the change must be incorporated in the B loop.

2) Ordinarily, the only changes desired will be made in PFNCT. This is an interpretive routine that selects the desired function for the variable by interpreting the function number given in the term matrix. If special terms were desired for some value of J, the test for this value could be inserted after the entry and the appropriate action taken. The user should also take care to cause the proper printing for the special function to occur for use in interpreting the results of the regression analysis. An appropriate change should also be made in subroutine generation program on page 224. This change would be made in the general term generator GEN TRM.

The stepwise regression program is structured as shown on page 211. Each of the major sections are written as a subroutine to allow for increased flexibility in keeping the analysis abreast of the best current methods. SUMSQ on page 212 loads the regression matrix with the processed data from the Editor program. In the loading process, the sums of the squares and the cross-products are accumulated. Printing of the result of the loading is available under the control of the parameter IFRAW. If the type of analysis selected by the parameter IFCNST requires the adjustment of the sums of squares and

cross-products about the means, the residual sum routine RSDSUM on page 213 carries out the computation.

The product moment coefficient of correlation for the terms with each other and with the dependent variable is calculated by the partial correlation routine PRCRCN. This routine also makes available the standard deviations for each of the terms and the dependent variable, if desired.

The regression analysis itself displays the interesting structure shown on page 215. It should be noted that only the degree of freedom routine DGFRDM and the matrix transformation routine MATRAN have a single logical connection to the switch N. The switch N directs the analysis through successive steps of sorting the terms in VARSRT (78), checking the variance contribution in VARCHK and transforming the matrix. After DGFRDM, VARSRT or VARCHK the analysis may be terminated depending upon the results through the routine RGRTRM. The degree of freedom routine DGFRDM on page 216 insures that the number of degrees of freedom are continually revised as the analysis progresses. Whenever the variance of the dependent variable is non-positive due to roundoff error or machine error or whenever there are no more degrees of freedom remaining the analysis is terminated.

VARSRT on pages 217 and 218 sorts the variables into the sets $X_{i,1}$ and $X_{i,2}$ (78). Depending upon the result of the sorting, the selected term will be checked by the F level test and the analysis will proceed through VARCHK or the analysis will terminate if no more terms are available.

The routine VARCHK compares the F level of the selected term to insure that the risk of committing an insertion or deletion error is not exceeded (81-83). If the requirements are satisfied, the regression matrix is transformed by MATRAN (31, 120) using the relations:

$$A_{i,j} = A_{i,j} - A_{i,k} * A_{k,j}/A_{k,k} \quad \begin{array}{l} i = 1,\ldots,n \\ j = 1,\ldots,n \\ i \neq k \\ j \neq k \\ n = \text{number of terms} \end{array}$$

$$\left.\begin{array}{l} A_{i,k} = -A_{i,k}/A_{k,k} \\ A_{k,i} = A_{k,i}/A_{k,k} \end{array}\right\} \quad i \neq k$$

$$A_{k,k} = 1./A_{k,k}$$

The regression analysis is terminated by RGRTRM. The result of the final step is printed and the predictions of the data using the regression equation is printed, on command, by PREDCT.

Upon completion of the regression analysis, WINDUP checks the postulated criteria given by the user against the properties of the generated relation. If further analysis is indicated and allowed by the number of trials, the program returns to the student program. Otherwise, the subroutine for the equation generated by the last trial is produced by the statement generating program on page 224 after which the program returns to the starter program.

ILLUSTRATIVE EXAMPLE

The Stepwise Regression Program with Simple Learning may be best appreciated by presenting the program with a set of data for which a predicting equation is desired. As a substitute for that experience, the following example is presented. This problem was one of many presented to the program during its development for the purpose of verifying the validity of the procedure. Since data arising from experimental sources always contains some random error components, the example was constructed using a normally distributed random number subroutine to add to a defined function the effect of random error.

The function used in this illustration was the following:

$$Y = 4.0 * X^2 - 16.0 * X + 15.0 + (EPSILON)$$

where EPSILON is a random normally distributed error with a mean value of zero and a standard deviation of 0.25.

In order to show the action of the Simple Learning mechanism, the control parameters are so chosen as to allow 20 of the "standard" functions of X but only allow the regression analysis to have access to 4 of these functions at any one time. Thus on a fairly simple scale, the behavior of the Simple Learning mechanism may be observed. A complete discussion of the data deck and control cards may be found in the Communication of the Problem to the Program beginning on page 98. Obviously, the capacity of the Stepwise Regression Program would allow the solution of this problem in a single trial if so desired. Random restarting of the problem should be expected to produce variations in the sequence and number of trials from those given here but the final result will be the same.

The data was supplied without any accumulated learning deck or suggested terms. On the first trial, the terms X^{-6}, X^2, X^5 and X^{-4} were chosen randomly from among the terms allowed by the 20 functions specified on the control card. Of these, X^2 and X^5 were sufficiently correlated with the data to be included in a predicting equation. The postulated standard error and coefficient of determination criteria were not satisfied, however, so the learning mechanism was called into the computation to assist the selection of new terms to be tested.

On the second trial, neither of the new terms X^4, $X^{1/4}$ were better choices than the terms X^2, X^5 found previously. The third trial suggested $X^{1/5}$, X^3 as new candidates and retained X^3 in addition to the earlier X^2 and X^5.

On the fourth attempt, only one new term was suggested for trial, X. Since X^2 and X were together far superior to any other combination of the trial terms for this attempt, the resulting equation:

$$Y = 3.99999857 * X^2 - 15.999832 * X + 15.0000304$$

satisfied both criteria and the analysis was terminated for the problem.

This example, while too simple to be of much practical value, is a fairly reasonable illustration of the technique. The value is even more apparent when the technique is applied to more complicated situations. The execution time for this problem was approximately five minutes, with about half of that time going to the loading of the magnetic program tape on the tape drive and initiating the problem. More representative figures for more complicated problems run on the order of fifteen to twenty minutes, with the actual time depending heavily upon the number of data sets.

```
* DATA
1     EXAMPLE PROBLEM FOR RELATION Y =4.*X**2 -16.X +15.+(EPSILON)
    1     .0001        .05        .05    1   20    4                    1
        .999         .5   10
1111111 13
    3           0.3E01           0.3E01          0.15E01
    1    3    4    5 3 1 2 2     4 1
     23295      8185         13                                          69
     FORMAT(F5.0,2E15.8)
 1.-0.77635685E 01 0.38030785E 03                                  DATA0001
 2. 0.72477829E 01 0.10915799E 03                                  DATA0002
 3. 0.18261584E 01-0.87888591E 00                                  DATA0003
 4. 0.68130090E 01 0.91661219E 02                                  DATA0004
 5.-0.31857231E 01 0.10656648E 03                                  DATA0005
 6. 0.13964993E 01 0.45702802E-00                                  DATA0006
 7. 0.29685392E 01 0.27526532E 01                                  DATA0007
 8.-0.93314382E 01 0.51260412E 03                                  DATA0008
 9.-0.24912679E 01 0.79685632E 02                                  DATA0009
10. 0.32323989E 01 0.50756451E 01                                  DATA0010
11.-0.31791103E 01 0.10629232E 03                                  DATA0011
12.-0.86843699E 01 0.45562153E 03                                  DATA0012
13.-0.80777898E 01 0.40524609E 03                                  DATA0013
14.-0.65720271E 01 0.29291764E 03                                  DATA0014
15. 0.86774063E 01 0.17735252E 03                                  DATA0015
16.-0.82241484E 01 0.41713149E 03                                  DATA0016
17.-0.37972297E 01 0.13343099E 03                                  DATA0017
18. 0.61368134E 01 0.67453766E 02                                  DATA0018
19. 0.70016510E 01 0.99067087E 02                                  DATA0019
20. 0.85431173E 01 0.17025098E 03                                  DATA0020
21. 0.30293479E 01 0.32386171E 01                                  DATA0021
22.-0.71202764E 01 0.33171670E 03                                  DATA0022
23.-0.50384158E 01 0.19715650E 03                                  DATA0023
24. 0.47421381E 01 0.29077920E 02                                  DATA0024
25. 0.43137813E 01 0.20414905E 02                                  DATA0025
26.-0.10008156E 01 0.35019451E 02                                  DATA0026
27.-0.83967283E 01 0.43136642E 03                                  DATA0027
28. 0.98397510E 01 0.24484918E 03                                  DATA0028
29.-0.46616156E 01 0.17650787E 03                                  DATA0029
30.-0.99740575E 01 0.57251016E 03                                  DATA0030
31. 0.81320423E 01 0.14940762E 03                                  DATA0031
32.-0.29972048E 01 0.98891438E 02                                  DATA0032
33. 0.45478162E 01 0.24966075E 02                                  DATA0033
34. 0.28013960E 01 0.15693002E 01                                  DATA0034
35.-0.77303118E 01 0.37771855E 03                                  DATA0035
36. 0.66956154E 01 0.87198816E 02                                  DATA0036
37. 0.71400023E 00 0.56152710E 01                                  DATA0037
38.-0.63002388E 01 0.27457628E 03                                  DATA0038
39. 0.11567351E 01 0.18445273E 01                                  DATA0039
40. 0.26133195E 01 0.50497638E 00                                  DATA0040
41.-0.33071546E 01 0.11166356E 03                                  DATA0041
42.-0.62067232E 01 0.26840033E 03                                  DATA0042
43.-0.92314617E 01 0.50358116E 03                                  DATA0043
44. 0.35399178E 01 0.84858461E 01                                  DATA0044
45.-0.35085416E 01 0.12037566E 03                                  DATA0045
46.-0.16950303E 01 0.53612783E 02                                  DATA0046
47.-0.26867548E 01 0.86862338E 02                                  DATA0047
48.-0.77692993E 01 0.38075562E 03                                  DATA0048

          EXAMPL
```

EXAMPLE PROBLEM FOR RELATION Y =4.*X**2 -16.X +15.+EPSILON

STARTER PROGRAM

PROBLEM NO. 1

RAW DATA

OBSERVATION NO. 1. WEIGHT = 1.00000
X(1) = -0.7763568E 01 X(2) = 0.3803078E 03 X(

OBSERVATION NO. 2. WEIGHT = 1.00000
X(1) = 0.7247783E 01 X(2) = 0.1091580E 03 X(

OBSERVATION NO. 3. WEIGHT = 1.00000
X(1) = 0.1826158E 01 X(2) = -0.8788859E 00 X(

OBSERVATION NO. 4. WEIGHT = 1.00000
X(1) = 0.6813009E 01 X(2) = 0.9166122E 02 X(

OBSERVATION NO. 5. WEIGHT = 1.00000
X(1) = -0.3185723E 01 X(2) = 0.1065665E 03 X(

OBSERVATION NO. 6. WEIGHT = 1.00000
X(1) = 0.1396499E 01 X(2) = 0.4570280E-00 X(

OBSERVATION NO. 7. WEIGHT = 1.00000
X(1) = 0.2968539E 01 X(2) = 0.2752653E 01 X(

OBSERVATION NO. 8. WEIGHT = 1.00000
X(1) = -0.9331438E 01 X(2) = 0.5126041E 03 X(

OBSERVATION NO. 9. WEIGHT = 1.00000
X(1) = -0.2491268E 01 X(2) = 0.7968563E 02 X(

OBSERVATION NO. 10. WEIGHT = 1.00000
X(1) = 0.3232399E 01 X(2) = 0.5075645E 01 X(

OBSERVATION NO. 11. WEIGHT = 1.00000
X(1) = -0.3179110E 01 X(2) = 0.1062923E 03 X(

OBSERVATION NO. 12. WEIGHT = 1.00000
X(1) = -0.8684370E 01 X(2) = 0.4556215E 03 X(

OBSERVATION NO. 13. WEIGHT = 1.00000
X(1) = -0.8077790E 01 X(2) = 0.4052461E 03 X(

OBSERVATION NO. 14. WEIGHT = 1.00000
X(1) = -0.6572027E 01 X(2) = 0.2929176E 03 X(

OBSERVATION NO. 15. WEIGHT = 1.00000
X(1) = 0.8677406E 01 X(2) = 0.1773525E 03 X(

OBSERVATION NO. 16. WEIGHT = 1.00000
X(1) = -0.8224148E 01 X(2) = 0.4171315E 03 X(

OBSERVATION NO. 17. WEIGHT = 1.00000
X(1) = -0.3797230E 01 X(2) = 0.1334310E 03 X(

OBSERVATION NO. 18. WEIGHT = 1.00000
X(1) = 0.6136813E 01 X(2) = 0.6745377E 02 X(

OBSERVATION NO. 19. WEIGHT = 1.00000

XC 1) = 0.7001651E 01 XC 2) = 0.9906709E 02 XC

OBSERVATION NO. 20. WEIGHT = 1.00000
XC 1) = 0.8543117E 01 XC 2) = 0.1702510E 03 XC

OBSERVATION NO. 21. WEIGHT = 1.00000
XC 1) = 0.3029348E 01 XC 2) = 0.3238617E 01 XC

OBSERVATION NO. 22. WEIGHT = 1.00000
XC 1) = -0.7120276E 01 XC 2) = 0.3317167E 03 XC

OBSERVATION NO. 23. WEIGHT = 1.00000
XC 1) = -0.5038416E 01 XC 2) = 0.1971565E 03 XC

OBSERVATION NO. 24. WEIGHT = 1.00000
XC 1) = 0.4742138E 01 XC 2) = 0.2907792E 02 XC

OBSERVATION NO. 25. WEIGHT = 1.00000
XC 1) = 0.4313781E 01 XC 2) = 0.2041490E 02 XC

OBSERVATION NO. 26. WEIGHT = 1.00000
XC 1) = -0.1000816E 01 XC 2) = 0.3501945E 02 XC

OBSERVATION NO. 27. WEIGHT = 1.00000
XC 1) = -0.8396728E 01 XC 2) = 0.4313664E 03 XC

OBSERVATION NO. 28. WEIGHT = 1.00000
XC 1) = 0.9839751E 01 XC 2) = 0.2448492E 03 XC

OBSERVATION NO. 29. WEIGHT = 1.00000
XC 1) = -0.4661615E 01 XC 2) = 0.1765079E 03 XC

OBSERVATION NO. 30. WEIGHT = 1.00000
XC 1) = -0.9974067E 01 XC 2) = 0.5725102E 03 XC

OBSERVATION NO. 31. WEIGHT = 1.00000
XC 1) = 0.8132012E 01 XC 2) = 0.1494076E 03 XC

OBSERVATION NO. 32. WEIGHT = 1.00000
XC 1) = -0.2997295E 01 XC 2) = 0.9889144E 02 XC

OBSERVATION NO. 33. WEIGHT = 1.00000
XC 1) = 0.4547816E 01 XC 2) = 0.2496607E 02 XC

OBSERVATION NO. 34. WEIGHT = 1.00000
XC 1) = 0.2801396E 01 XC 2) = 0.1569300E 01 XC

OBSERVATION NO. 35. WEIGHT = 1.00000
XC 1) = -0.7730362E 01 XC 2) = 0.3777185E 03 XC

OBSERVATION NO. 36. WEIGHT = 1.00000
XC 1) = 0.6695685E 01 XC 2) = 0.6719801E 02 XC

OBSERVATION NO. 37. WEIGHT = 1.00000
XC 1) = 0.7140002E 00 XC 2) = 0.5615271E 01 XC

OBSERVATION NO. 38. WEIGHT = 1.00000
XC 1) = -0.6300259E 01 XC 2) = 0.2745763E 03 XC

OBSERVATION NO. 39. WEIGHT = 1.00000
XC 1) = 0.1156735E 01 XC 2) = 0.1044527E 01 XC

```
OBSERVATION NO.   40.   WEIGHT =    1.00000
      X( 1) =  0.2613319E 01        X( 2) =  0.5049764E 00        X(

OBSERVATION NO.   41.   WEIGHT =    1.00000
      X( 1) = -0.3307165E 01        X( 2) =  0.1116636E 03        X(

OBSERVATION NO.   42.   WEIGHT =    1.00000
      X( 1) = -0.6206723E 01        X( 2) =  0.2684003E 03        X(

OBSERVATION NO.   43.   WEIGHT =    1.00000
      X( 1) = -0.9231462E 01        X( 2) =  0.5035812E 03        X(

OBSERVATION NO.   44.   WEIGHT =    1.00000
      X( 1) =  0.3539918E 01        X( 2) =  0.8485846E 01        X(

OBSERVATION NO.   45.   WEIGHT =    1.00000
      X( 1) = -0.3508542E 01        X( 2) =  0.1203757E 03        X(

OBSERVATION NO.   46.   WEIGHT =    1.00000
      X( 1) = -0.1695030E 01        X( 2) =  0.5361278E 02        X(

OBSERVATION NO.   47.   WEIGHT =    1.00000
      X( 1) = -0.2686755E 01        X( 2) =  0.8686234E 02        X(

OBSERVATION NO.   48.   WEIGHT =    1.00000
      X( 1) = -0.7769299E 01        X( 2) =  0.3807556E 03        X(

EDITOR PROGRAM


PROBLEM NO.     1
SOLUTION PASS NO.     1
NO. OF INDEPENDENT VARIABLES =     1
NO. OF TRIAL TERMS =     4

TRIAL TERM DEFINITIONS FOR PASS NO.     1

TERM( 1) = INTERACTION OF ORDER  1, WHERE THE COMPONENTS ARE DEFINED TO BE --
          COMPONENT( 1) = X( 1) .P. -  6

TERM( 2) = INTERACTION OF ORDER  1, WHERE THE COMPONENTS ARE DEFINED TO BE --
          COMPONENT( 1) = X( 1) .P.    2

TERM( 3) = INTERACTION OF ORDER  1, WHERE THE COMPONENTS ARE DEFINED TO BE --
          COMPONENT( 1) = X( 1) .P.    5

TERM( 4) = INTERACTION OF ORDER  1, WHERE THE COMPONENTS ARE DEFINED TO BE --
          COMPONENT( 1) = X( 1) .P. -  4

TERM( 5) = X( 2), DEPENDENT VARIABLE.
```

STEPWISE REGRESSION

PROBLEM NO. 1

NO. OF DATA SETS = 48

NO. OF TERM CHOICES = 4

PROBABILITY OF
1) ERROR IN ENTERING TERM = 5.0000 0/0
2) ERROR IN DELETING TERM = 5.0000 0/0

WEIGHTED DEGREES OF FREEDOM = 48.00

STANDARD ERROR OF V = 0.166283123E 03

STEP NO. 2
TERM ENTERED 3
F LEVEL = 0.647523925E-01
STANDARD ERROR OF V = 0.5517846B7E 02
COEFF OF DETERMINATION = 0.894571610E 00
MULTIPLE CORLTN COEFF = 0.945817955E 00

CONSTANT TERM = 0.168174595E 02

TERM NO.	COEFFICIENT	STD ERR OF COEFF
TERM- 2	0.399137430E 01	0.278628640E-00
TERM- 3	-0.258267805E-02	0.273117624E-03

REGRESSION TERMINATED AFTER 2 STEPS.

DIAGONAL ELEMENTS

VAR. NO.	VALUE
1	0.9560751O2E 00
2	0.106134199E 01
3	0.106134199E 01
4	0.931943156E 00

PREDICTED RESULTS VERSUS DATA POINTS

OBS. NO.	PREDICTIONS V ± SIGMA	PREDICTIONS V	V + SIGMA	DATA POINTS	DEVIATIONS (DATA - V)	DEVIATIONS PERCENT
1.	0.27505229E 03	0.33023076E 03	0.385409226 03	0.33030785E 03	0.50077091E 02	13.168
2.	0.11965423E 03	0.17483270E 03	0.23001117E 03	0.10915799E 03	-0.65674715E 02	-60.165
3.	-0.25102806E 02	0.30075660E 02	0.85254129E 02	-0.97888591E 00	-0.309545466 02	3522.021
4.	0.10899598E 03	0.16417445E 03	0.219352926 03	0.91661219E 02	-0.72513238E 02	-79.110
5.	0.29942183E 01	0.58172687E 02	0.113351156 03	0.10656647E 03	0.48393792E 02	45.412
6.	-0.30590707E 02	0.24587761E 02	0.797662306 02	0.45702802E-00	-0.24130733E 02	-5279.924
7.	-0.37834883E 01	0.51394980E 02	0.106573456 03	0.27526531E 01	-0.48642328E 02	-1767.107
8.	0.49192224E 03	0.54710072E 03	0.602279196 03	0.51260412E 03	-0.34496597E 02	-6.730
9.	-0.13341041E 02	0.41837428E 02	0.970158966 02	0.79685631E 02	0.37848204E 02	47.497
10.	0.24311051E 01	0.57609574E 02	0.112788046 03	0.50756451E 01	-0.52533929E 02	-1035.020
11.	0.28174648E 01	0.57995933E 02	0.113174406 03	0.10629231E 03	0.48296385E 02	45.437
12.	0.39023546E 03	0.44541392E 03	0.500592406 03	0.45562153E 03	0.10207599E 02	2.240
13.	0.31090345E 03	0.36608192E 03	0.421260396 03	0.40524609E 03	0.39164169E 02	9.664
14.	0.16569667E 03	0.22087514E 03	0.276053616 03	0.29291763E 03	0.72042488E 02	24.595
15.	0.13511577E 03	0.19029424E 03	0.245472716 03	0.17735252E 03	-0.12941723E 02	-7.297

16.	0.32877041E 03	0.38394888E 03	0.43912735E 03	0.41713148E 03	0.33182602E 02	7.955
17.	0.21229369E 02	0.76407839E 02	0.13158631E 03	0.13343099E 03	0.57023149E 02	42.736
18.	0.89476648E 02	0.14465512E 03	0.19983358E 03	0.67453766E 02	-0.77201351E 02	-114.451
19.	0.11385031E 03	0.16902878E 03	0.22420725E 03	0.99067086E 02	-0.69961697E 02	-70.621
20.	0.13541794E 03	0.19059641E 03	0.24577487E 03	0.17025097E 03	-0.20345432E 02	-11.950
21.	-0.23912673E 01	0.52787202E 02	0.10796567E 03	0.32386170E 01	-0.49548584E 02	-1529.930
22.	0.21126164E 03	0.26644011E 03	0.32161857E 03	0.33171669E 03	0.65276588E 02	19.678
23.	0.71348273E 02	0.12652674E 03	0.18170521E 03	0.19715650E 03	0.70629758E 02	35.824
24.	0.45202938E 02	0.10038140E 03	0.15555988E 03	0.29077920E 02	-0.71303488E 02	-245.215
25.	0.32055324E 02	0.87233793E 02	0.14241226E 03	0.20414905E 02	-0.66818888E 02	-327.304
26.	-0.34360529E 02	0.20817940E 02	0.75996409E 02	0.35019451E 02	0.14201510E 02	40.553
27.	0.35085150E 03	0.40602997E 03	0.46120843E 03	0.43136641E 03	0.25336444E 02	5.874
28.	0.10985973E 03	0.16503821E 03	0.22021668E 03	0.24484918E 03	0.79810970E 02	32.596
29.	0.54059473E 02	0.10923794E 03	0.16441641E 03	0.17650787E 03	0.67269927E 02	38.112
30.	0.61364532E 03	0.66882379E 03	0.72400226E 03	0.57251015E 03	-0.96313637E 02	-16.823
31.	0.13374104E 03	0.18891950E 03	0.24409797E 03	0.14940762E 03	-0.39511886E 02	-26.446
32.	-0.18786311E 01	0.53299838E 02	0.10847830E 03	0.98891437E 02	0.45591599E 02	46.103
33.	0.39166708E 02	0.94345177E 02	0.14952364E 03	0.24966075E 02	-0.69379102E 02	-277.894
34.	-0.74830217E 01	0.47695447E 02	0.10287391E 03	0.15693001E 01	-0.46126147E 02	-2939.281
35.	0.27145419E 03	0.32663265E 03	0.38181112E 03	0.37771855E 03	0.51085892E 02	13.525
36.	0.10582384E 03	0.16100230E 03	0.21618077E 03	0.87198815E 02	-0.73803490E 02	-84.638
37.	-0.36326700E 02	0.18851768E 02	0.74030237E 02	0.56152710E 01	-0.13236497E 02	-235.723
38.	0.14570635E 03	0.20088482E 03	0.25606328E 03	0.27457628E 03	0.73691458E 02	26.838
39.	-0.33025755E 02	0.22152714E 02	0.77331182E 02	0.18445273E 01	-0.20308186E 02	-1100.997
40.	-0.11416961E 02	0.43761507E 02	0.98939976E 02	0.50497638E 00	-0.43256531E 02	-8566.050
41.	0.63157591E 01	0.61494228E 02	0.11667269E 03	0.11166356E 03	0.50169331E 02	44.929
42.	0.13918967E 03	0.19436814E 03	0.24954661E 03	0.26840033E 03	0.74032188E 02	27.583
43.	0.47493349E 03	0.53011196E 03	0.58529042E 03	0.50358115E 03	-0.26530803E 02	-5.268
44.	0.10219363E 02	0.65397833E 02	0.12057630E 03	0.84858460E 01	-0.56911986E 02	-670.670
45.	0.12145368E 02	0.67323837E 02	0.12250230E 03	0.12037566E 03	0.53051923E 02	44.072
46.	-0.26857144E 02	0.28321325E 02	0.83499794E 02	0.53612783E 02	0.25291458E 02	47.174
47.	-0.91870856E 01	0.45991383E 02	0.10116985E 03	0.86862337E 02	0.40870953E 02	47.053
48.	0.27567682E 03	0.33085530E 03	0.38603376E 03	0.38075562E 03	0.49900322E 02	13.106

MAXIMUM ABSOLUTE DEVIATION = 0.9631364E 02,(SEE OBS. NO. 30., LINE NO. 30)

MAXIMUM ABSOLUTE PERCENT DEVIATION = 8566.050,(SEE OBS. NO. 40., LINE NO. 40)

POSTULATED CRITERIA

STANDARD ERROR OF Y = 0.5000000E 00
COEFF OF DETERMINATION = 0.9990000E 00

FITTED CURVE PROPERTIES

STANDARD ERROR OF Y = 0.5517847E 02
COEFF OF DETERMINATION = 0.3945716E 00

FITTED CURVE MEETS NEITHER CRITERIA.

PASS NUMBER 2 BEGUN FOR PROBLEM NO. 1
10 TOTAL PASSES ALLOWED.

EDITOR PROGRAM

PROBLEM NO. 1
SOLUTION PASS NO. 2
NO. OF INDEPENDENT VARIABLES = 1
NO. OF TRIAL TERMS = 4

TRIAL TERM DEFINITIONS FOR PASS NO. 2

TERM(1) = INTERACTION OF ORDER 1, WHERE THE COMPONENTS ARE DEFINED TO BE --
COMPONENT(1) = X(1) ,P. 2

TERM(2) = INTERACTION OF ORDER 1, WHERE THE COMPONENTS ARE DEFINED TO BE --
COMPONENT(1) = X(1) ,P. 5

TERM(3) = INTERACTION OF ORDER 1, WHERE THE COMPONENTS ARE DEFINED TO BE --
COMPONENT(1) = X(1) ,P. 4

TERM(4) = INTERACTION OF ORDER 1, WHERE THE COMPONENTS ARE DEFINED TO BE --
COMPONENT(1) = X(1) ,P. 1/ 4

TERM(5) = X(2), DEPENDENT VARIABLE.

STEPWISE REGRESSION

PROBLEM NO. 1

NO. OF DATA SETS = 48

NO. OF TERM CHOICES = 4

PROBABILITY OF
1) ERROR IN ENTERING TERM = 5.0000 0/0
2) ERROR IN DELETING TERM = 5.0000 0/0

WEIGHTED DEGREES OF FREEDOM = 48.00

STANDARD ERROR OF Y = 0.166283123E 03

STEP NO. 2
TERM ENTERED 2
F LEVEL = 0.503597260E 00
STANDARD ERROR OF Y = 0.551784687E 02
COEFF OF DETERMINATION = 0.894571610E 00
MULTIPLE CORLTN COEFF = 0.945817955E 00

CONSTANT TERM = 0.168174595E 02

TERM NO.	COEFFICIENT	STD ERR OF COEFF
TERM- 1	0.3991374 30E 01	0.278628640E-00
TERM- 2	-0.258267805E-02	0.273117624E-03

REGRESSION TERMINATED AFTER 2 STEPS.

DIAGONAL ELEMENTS

VAR. NO.	VALUE
1	0.106134199E 01
2	0.106134199E 01
3	0.744117185E-01
4	0.163850956E-00

PREDICTED RESULTS VERSUS DATA POINTS

OBS. NO.	PREDICTIONS Y - SIGMA	PREDICTIONS Y	PREDICTIONS Y + SIGMA	DATA POINTS	DEVIATIONS (DATA - Y)	DEVIATIONS PERCENT
1.	0.27505229E 03	0.33023076E 03	0.38540922E 03	0.38030785E 03	0.50077091E 02	13.168
2.	0.11965423E 03	0.17483270E 03	0.23001117E 03	0.10915799E 03	-0.65674715E 02	-60.165
3.	-0.25102808E 02	0.30075660E 02	0.85254129E 02	-0.87898591E 00	-0.30954546E 02	3522.021
4.	0.10899598E 03	0.16417445E 03	0.21935292E 03	0.91661219E 02	-0.72513238E 02	-79.110
5.	0.29942183E 01	0.58172687E 02	0.11335115E 03	0.10656647E 03	0.48393792E 02	45.412
6.	-0.30590707E 02	0.24587761E 02	0.79766230E 02	0.45702802E-00	-0.24130733E 02	-5279.324
7.	-0.37834883E 01	0.51394980E 02	0.10657345E 03	0.27526531E 01	-0.48642328E 02	-1767.107
8.	0.49192224E 03	0.54710072E 03	0.60227918E 03	0.51260412E 03	-0.34496597E 02	-6.730
9.	-0.13341041E 02	0.41837428E 02	0.97015896E 02	0.79685631E 02	0.37848204E 02	47.497
10.	0.24311051E 01	0.57609574E 02	0.11278804E 03	0.53756451E 01	-0.52533929E 02	-1035.042
11.	0.28174648E 01	0.57995933E 02	0.11317440E 03	0.10629231E 03	0.48296385E 02	45.437
12.	0.39023546E 03	0.44541392E 03	0.50059240E 03	0.45562153E 03	0.10207539E 02	2.240
13.	0.31090345E 03	0.36608192E 03	0.42126039E 03	0.40524609E 03	0.39164169E 02	9.664
14.	0.16569667E 03	0.22067514E 03	0.27605361E 03	0.29291763E 03	0.72042488E 02	24.595
15.	0.13511577E 03	0.19029424E 03	0.24547271E 03	0.17735252E 03	-0.12941723E 02	-7.297

16.	0.32877041E 03	0.38394888E 03	0.43912735E 03	0.41713148E 03	0.33182602E 02	7.955
17.	0.21229369E 02	0.76407839E 02	0.13158631E 03	0.13343099E 03	0.57023149E 02	42.736
18.	0.89476648E 02	0.14465512E 03	0.19983358E 03	0.67453766E 02	-0.77201351E 02	-114.451
19.	0.11385031E 03	0.16902878E 03	0.22420725E 03	0.99067086E 02	-0.69961697E 02	-70.621
20.	0.13541794E 03	0.19059641E 03	0.24577487E 03	0.17025097E 03	-0.20345432E 02	-11.950
21.	-0.23912673E 01	0.52787202E 02	0.10796567E 03	0.32386170E 01	-0.49548584E 02	-1529.930
22.	0.21126164E 03	0.26644011E 03	0.32161857E 03	0.33171669E 03	0.65276588E 02	19.678
23.	0.71348273E 02	0.12652674E 03	0.18170521E 03	0.19715650E 03	0.70629758E 02	35.824
24.	0.45202938E 02	0.10038140E 03	0.15555988E 03	0.29077920E 02	-0.71303488E 02	-245.215
25.	0.32055324E 02	0.87233793E 02	0.14241226E 03	0.20414905E 02	-0.66818888E 02	-327.304
26.	-0.34360529E 02	0.20817940E 02	0.75996409E 02	0.35019451E 02	0.14201510E 02	40.553
27.	0.35085150E 03	0.40602997E 03	0.46120843E 03	0.43136641E 03	0.25336444E 02	5.874
28.	0.10985973E 03	0.16503821E 03	0.22021668E 03	0.24484918E 03	0.79810970E 02	32.596
29.	0.54059473E 02	0.10923794E 03	0.16441641E 03	0.17650787E 03	0.67269927E 02	38.112
30.	0.61364532E 03	0.66882379E 03	0.72400226E 03	0.57251015E 03	-0.96313637E 02	-16.823
31.	0.13374104E 03	0.18891950E 03	0.24409797E 03	0.14940762E 03	-0.39511886E 02	-26.446
32.	-0.18786311E 01	0.53299338E 02	0.10847830E 03	0.98891437E 02	0.45591599E 02	46.103
33.	0.39166708E 02	0.94345177E 02	0.14952364E 03	0.24966075E 02	-0.69379102E 02	-277.894
34.	-0.74830217E 01	0.47695447E 02	0.10287391E 03	0.15693031E 01	-0.46126147E 02	-2939.281
35.	0.27145419E 03	0.32663265E 03	0.38181112E 03	0.37771855E 03	0.51085892E 02	13.525
36.	0.10582384E 03	0.16100230E 03	0.21618077E 03	0.87198815E 02	-0.73803400E 02	-84.638
37.	-0.36326700E 02	0.18851768E 02	0.74030237E 02	0.56152710E 01	-0.13236497E 02	-235.723
38.	0.14570635E 03	0.20088482E 03	0.25606328E 03	0.27457628E 03	0.73691458E 02	26.838
39.	-0.33025755E 02	0.22152714E 02	0.77331132E 02	0.18445273E 01	-0.20308186E 02	-1100.997
40.	-0.11416961E 02	0.43761507E 02	0.98939976E 02	0.50497638E 00	-0.43256531E 02	-8566.050
41.	0.63157591E 01	0.61494228E 02	0.11667269E 03	0.11166356E 03	0.50169331E 02	44.929
42.	0.13918967E 03	0.19436814E 03	0.24954661E 03	0.26840033E 03	0.74032188E 02	27.583
43.	0.47493349E 03	0.53011196E 03	0.58529042E 03	0.50358115E 03	-0.26530803E 02	-5.268
44.	0.10219363E 02	0.65397833E 02	0.12057630E 03	0.84858460E 01	-0.56911986E 02	-670.670
45.	0.12145368E 02	0.67323837E 02	0.12250230E 03	0.12037566E 03	0.53051823E 02	44.072
46.	-0.26857144E 02	0.28321325E 02	0.83499794E 02	0.53612783E 02	0.25291458E 02	47.174
47.	-0.91870856E 01	0.45991383E 02	0.10116985E 03	0.86862337E 02	0.40870953E 02	47.053
48.	0.27567682E 03	0.33085530E 03	0.38603376E 03	0.38075562E 03	0.49900322E 02	13.106

MAXIMUM ABSOLUTE DEVIATION = 0.9631364E 02,(SEE OBS. NO. 30., LINE NO. 30)

MAXIMUM ABSOLUTE PERCENT DEVIATION = 8566.050,(SEE OBS. NO. 40., LINE NO. 40)

POSTULATED CRITERIA

STANDARD ERROR OF Y = 0.5000000E 00
COEFF OF DETERMINATION = 0.9990000E 00

FITTED CURVE PROPERTIES

STANDARD ERROR OF Y = 0.5517847E 02
COEFF OF DETERMINATION = 0.8945716E 00

FITTED CURVE MEETS NEITHER CRITERIA.

PASS NUMBER 3 BEGUN FOR PROBLEM NO. 1
10 TOTAL PASSES ALLOWED.

EDITOR PROGRAM

PROBLEM NO. 1
SOLUTION PASS NO. 3
NO. OF INDEPENDENT VARIABLES = 1
NO. OF TRIAL TERMS = 4

TRIAL TERM DEFINITIONS FOR PASS NO. 3

TERM(1) = INTERACTION OF ORDER 1, WHERE THE COMPONENTS ARE DEFINED TO BE --
COMPONENT(1) = X(1) ,P, 2

TERM(2) = INTERACTION OF ORDER 1, WHERE THE COMPONENTS ARE DEFINED TO BE --
COMPONENT(1) = X(1) ,P, 5

TERM(3) = INTERACTION OF ORDER 1, WHERE THE COMPONENTS ARE DEFINED TO BE --
COMPONENT(1) = X(1) ,P, 1/ 5

TERM(4) = INTERACTION OF ORDER 1, WHERE THE COMPONENTS ARE DEFINED TO BE --
COMPONENT(1) = X(1) ,P, 3

TERM(5) = X(2), DEPENDENT VARIABLE.

STEPWISE REGRESSION

PROBLEM NO. 1

NO. OF DATA SETS = 48

NO. OF TERM CHOICES = 4

PROBABILITY OF
1) ERROR IN ENTERING TERM = 5.0000 0/0
2) ERROR IN DELETING TERM = 5.0000 0/0

WEIGHTED DEGREES OF FREEDOM = 48.00

STANDARD ERROR OF Y = 0.166283123E 03

STEP NO. 3
TERM ENTERED 2
F LEVEL = 0.133637249E-00
STANDARD ERROR OF Y = 0.244193383E 02
COEFF OF DETERMINATION = 0.979810469E 00
MULTIPLE CORLTN COEFF = 0.989853755E 00

CONSTANT TERM = 0.146384627E 02

TERM NO.	COEFFICIENT	STD ERR OF COEFF
TERM- 1	0.398712836E 01	0.123308040E-00
TERM- 2	0.417513050E-02	0.510339618E-03
TERM- 4	-0.546617232E 00	0.401052549E-01

REGRESSION TERMINATED AFTER 3 STEPS.

DIAGONAL ELEMENTS

VAR. NO.	VALUE
1	0.106134877E 01
2	0.189210802E 02
3	0.175644755E-00
4	0.188650332E 02

PREDICTED RESULTS VERSUS DATA POINTS

OBS. NO.	PREDICTIONS			DATA	DEVIATIONS	
	Y - SIGMA	Y	Y + SIGMA	POINTS	(DATA - Y)	PERCENT
1.	0.36856145E 03	0.392980796 03	0.41740013E 03	0.38030785E 03	-0.12672947E 02	-3.332
2.	0.75053392E 02	0.99472731E 02	0.12389207E 03	0.10915799E 03	0.96852589E 01	8.873
3.	0.27152658E-00	0.24690865E 02	0.49110204E 02	-0.87888591E 00	-0.25569751E 02	2909.337
4.	0.63714263E 02	0.88133602E 02	0.11255293E 03	0.91661219E 02	0.35276165E 01	3.949
5.	0.46986733E 02	0.71406072E 02	0.95825410E 02	0.10656647E 03	0.35160407E 02	32.994
6.	-0.34716561E 01	0.20947683E 02	0.45367021E 02	0.45702802E-00	-0.20490655E 02	-4483.457
7.	0.12017860E 02	0.36437199E 02	0.60856538E 02	0.27526531E 01	-0.33684546E 02	-1223.712
8.	0.48614096E 03	0.51056831E 03	0.53498764E 03	0.51260412E 03	0.20358124E 01	0.397
9.	0.23015954E 02	0.47435293E 02	0.71854631E 02	0.79685631E 02	0.32250339E 02	40.472
10.	0.14890432E 02	0.39309771E 02	0.63729110E 02	0.50756451E 01	-0.34234127E 02	-674.478
11.	0.46723252E 02	0.71142590E 02	0.95561928E 02	0.10629231E 03	0.35149729E 02	33.069
12.	0.44269939E 03	0.46711873E 03	0.49153806E 03	0.45562153E 03	-0.11497200E 02	-2.523
13.	0.39490088E 03	0.41932022B 03	0.44373956E 03	0.40524609E 03	-0.14074135E 02	-3.473
14.	0.26640206E 03	0.29082140E 03	0.31524073E 03	0.29291763E 03	0.20962372E 01	0.716

15.	0.13869653E 03	0.16311587E 03	0.18753520E 03	0.17735252E 03	0.14236648E 02	8.027
16.	0.40687139E 03	0.43129072E 03	0.45571006E 03	0.41713148E 03	-0.14159237E 02	-3.394
17.	0.74341639E 02	0.98760977E 02	0.12318031E 03	0.13343099E 03	0.34670010E 02	25.983
18.	0.50384638E 02	0.74803977E 02	0.99223315E 02	0.67453766E 02	-0.73502111E 01	-10.897
19.	0.68312383E 02	0.92731722E 02	0.11715106E 03	0.99067086E 02	0.63353643E 01	6.395
20.	0.13039259E 03	0.15481193E 03	0.17923126E 03	0.17025097E 03	0.15439051E 02	9.068
21.	0.12677906E 02	0.37097245E 02	0.61516584E 02	0.32386170E 01	-0.33858629E 02	-1045.466
22.	0.31327043E 03	0.33768976E 03	0.36210910E 03	0.33171669E 03	-0.59730682E 01	-1.801
23.	0.14779282E 03	0.17221216E 03	0.19663150E 03	0.19715650E 03	0.24944334E 02	12.652
24.	0.31602037E 02	0.56021376E 02	0.80440714E 02	0.29077920E 02	-0.26943456E 02	-92.660
25.	0.26772121E 02	0.51191460E 02	0.75610799E 02	0.20414905E 02	-0.30776555E 02	-150.755
26.	-0.52434778E 01	0.19175861E 02	0.43595200E 02	0.35019451E 02	0.15843589E 02	45.242
27.	0.42066642E 03	0.44508576E 03	0.46950509E 03	0.43136641E 03	-0.13719345E 02	-3.180
28.	0.24061318E 03	0.26503251E 03	0.28945185E 03	0.24484918E 03	-0.20183337E 02	-8.243
29.	0.12304358E 03	0.14746293E 03	0.17188226E 03	0.17650787E 03	0.29044943E 02	16.455
30.	0.51711493E 03	0.54153427E 03	0.56595360E 03	0.57251015E 03	0.30975883E 02	5.411
31.	0.10841110E 03	0.13283043E 03	0.15724977E 03	0.14940762E 03	0.16577183E 02	11.095
32.	0.39747377E 02	0.64166716E 02	0.88586055E 02	0.98891437E 02	0.34724721E 02	35.114
33.	0.29390577E 02	0.53809916E 02	0.78229254E 02	0.24966075E 02	-0.28843841E 02	-115.532
34.	0.10212438E 02	0.34631777E 02	0.59051115E 02	0.15693001E 01	-0.33062476E 02	-2106.829
35.	0.36573885E 03	0.39015818E 03	0.41457752E 03	0.37771855E 03	-0.12439640E 02	-3.293
36.	0.61074256E 02	0.85493594E 02	0.10991293E 03	0.87198815E 02	0.17052202E 01	1.956
37.	-0.79464436E 01	0.16472895E 02	0.40892234E 02	0.56152710E 01	-0.10857624E 02	-193.359
38.	0.24373405E 03	0.26815339E 03	0.29257273E 03	0.27457628E 03	0.64228859E 01	2.339
39.	-0.52833369E 01	0.19136002E 02	0.43555340E 02	0.18445273E 01	-0.17291474E 02	-937.447
40.	0.82021186E 01	0.32621457E 02	0.57040796E 02	0.50497638E 00	-0.32116481E 02	-6359.997
41.	0.51947930E 02	0.76367269E 02	0.10078660E 03	0.11166356E 03	0.35296290E 02	31.609
42.	0.23605778E 03	0.26047712E 03	0.28489646E 03	0.26840033E 03	0.79232063E 01	2.952
43.	0.48011518E 03	0.50453451E 03	0.52895385E 03	0.50358115E 03	-0.95336151E 00	-0.189
44.	0.18255421E 02	0.42674760E 02	0.67094098E 02	0.84858460E 01	-0.34188914E 02	-402.893
45.	0.60688604E 02	0.85107942E 02	0..09527276 03	0.12037566E 03	0.35267717E 02	29.298
46.	0.42782802E 01	0.28697619E 02	0.53116958E 02	0.53612783E 02	0.24915163E 02	46.472
47.	0.29017781E 02	0.53437120E 02	0.77856458E 02	0.86862337E 02	0.33425217E 02	38.481
48.	0.36904796E 03	0.39346730E 03	0.41788664E 03	0.38075562E 03	-0.12711681E 02	-3.339

MAXIMUM ABSOLUTE DEVIATION = 0.3529629E 02,(SEE OBS. NO. 41., LINE NO. 41)

MAXIMUM ABSOLUTE PERCENT DEVIATION = 6359.997,(SEE OBS. NO. 40., LINE NO. 40)

POSTULATED CRITERIA

STANDARD ERROR OF Y = 0.5000000E 00
COEFF OF DETERMINATION = 0.9990000E 00

FITTED CURVE PROPERTIES

STANDARD ERROR OF Y = 0.2441934E 02
COEFF OF DETERMINATION = 0.9798105E 00

FITTED CURVE MEETS NEITHER CRITERIA.

PASS NUMBER 4 BEGUN FOR PROBLEM NO. 1
10 TOTAL PASSES ALLOWED.

EDITOR PROGRAM

PROBLEM NO. 1
SOLUTION PASS NO. 4
NO. OF INDEPENDENT VARIABLES = 1
NO. OF TRIAL TERMS = 4

TRIAL TERM DEFINITIONS FOR PASS NO. 4

TERM(1) = INTERACTION OF ORDER 1, WHERE THE COMPONENTS ARE DEFINED TO BE --
COMPONENT(1) = X(1) .P. 2

TERM(2) = INTERACTION OF ORDER 1, WHERE THE COMPONENTS ARE DEFINED TO BE --
COMPONENT(1) = X(1) .P. 5

TERM(3) = INTERACTION OF ORDER 1, WHERE THE COMPONENTS ARE DEFINED TO BE --
COMPONENT(1) = X(1) .P. 3

TERM(4) = INTERACTION OF ORDER 1, WHERE THE COMPONENTS ARE DEFINED TO BE --
COMPONENT(1) = X(1) .P. 1

TERM(5) = X(2), DEPENDENT VARIABLE.

STEPWISE REGRESSION

PROBLEM NO. 1

NO. OF DATA SETS = 48

NO. OF TERM CHOICES = 4

PROBABILITY OF
1) ERROR IN ENTERING TERM = 5.0000 0/0
2) ERROR IN DELETING TERM = 5.0000 0/0

WEIGHTED DEGREES OF FREEDOM = 48.00

STANDARD ERROR OF Y = 0.166283123E 03
ARRAY5(Y,YD) = -0.6705523E-07

STEP NO. 2
TERM ENTERED 4
F LEVEL = 0.115484945E-02
STANDARD ERROR OF Y = 0.440055206E-01
COEFF OF DETERMINATION = 0.999999933E 00
MULTIPLE CORLTN COEFF = 0.999999963E 00

CONSTANT TERM = 0.150000304E 02

TERM NO.	COEFFICIENT	STD ERR OF COEFF
TERM- 1	0.399999857E 01	0.219932971E-03
TERM- 4	-0.159998320E 02	0.110057287E-02

REGRESSION TERMINATED AFTER 2 STEPS.

DIAGONAL ELEMENTS

VAR. NO.	VALUE
1	0.103970297E 01
2	0.315417871E-00
3	0.152735107E-00
4	0.103970297E 01

PREDICTED RESULTS VERSUS DATA POINTS

OBS. NO.	PREDICTIONS			DATA	DEVIATIONS	
	Y - SIGMA	Y	Y + SIGMA	POINTS	(DATA - Y)	PERCENT
1.	0.38026371E 03	0.38030771E 03	0.38035171E 03	0.38030785E 03	0.13351440E-03	0.000
2.	0.10911406E 03	0.10915806E 03	0.10920207E 03	0.10915799E 03	-0.78201294E-04	-0.000
3.	-0.92279042E 00	-0.87878489E 00	-0.83477937E 00	-0.87888591E 00	-0.10101497E-03	0.011
4.	0.91617320E 02	0.91661325E 02	0.91705330E 02	0.91661219E 02	-0.10681152E-03	-0.000
5.	0.10652237E 03	0.10656638E 03	0.10661038E 03	0.10656647E 03	0.10204315E-03	0.000
6.	0.41310870E-00	0.45711422E-00	0.50111974E 00	0.45702802E-00	-0.86203218E-04	-0.019
7.	0.27087816E 01	0.27527871E 01	0.27967926E 01	0.27526531E 01	-0.13393164E-03	-0.005
8.	0.51256030E 03	0.51260430E 03	0.51264830E 03	0.51260412E 03	-0.18310547E-03	-0.000
9.	0.79641547E 02	0.79685552E 02	0.79729558E 02	0.79685631E 02	0.79154968E-04	0.000
10.	0.50317791E 01	0.50757846E 01	0.51197901E 01	0.50756451E 01	-0.13959408E-03	-0.003
11.	0.10624821E 03	0.10629221E 03	0.10633622E 03	0.10629231E 03	0.10299683E-03	0.000
12.	0.45557749E 03	0.45562149E 03	0.45566549E 03	0.45562153E 03	0.34332275E-04	0.000
13.	0.40520196E 03	0.40524597E 03	0.40528997E 03	0.40524609E 03	0.12207031E-03	0.000
14.	0.29287345E 03	0.29291745E 03	0.29296146E 03	0.29291763E 03	0.17929077E-03	0.000

15.	0.17730837E 03	0.17735238E 03	0.17739639E 03	0.17735252E 03	0.13542175E-03	0.000
16.	0.41708738E 03	0.41713139E 03	0.41717539E 03	0.41713148E 03	0.95367432E-04	0.000
17.	0.13338685E 03	0.13343085E 03	0.13347486E 03	0.13343099E 03	0.12779236E-03	0.000
18.	0.67409895E 02	0.67453900E 02	0.67497905E 02	0.67453766E 02	-0.13446808E-03	-0.000
19.	0.99023176E 02	0.99067181E 02	0.99111187E 02	0.99067086E 02	-0.95367432E-04	-0.000
20.	0.17020688E 03	0.17025089E 03	0.17029489E 03	0.17025097E 03	0.87738037E-04	0.000
21.	0.31947478E 01	0.32387533E 01	0.32827588E 01	0.32386170E 01	-0.13622642E-03	-0.004
22.	0.33167252E 03	0.33171652E 03	0.33176053E 03	0.33171669E 03	0.17166138E-03	0.000
23.	0.19711233E 03	0.19715633E 03	0.19720034E 03	0.19715650E 03	0.16403198E-03	0.000
24.	0.29034071E 02	0.29078077E 02	0.29122082E 02	0.29077920E 02	-0.15735626E-03	-0.001
25.	0.20371056E 02	0.20415061E 02	0.20459066E 02	0.20414905E 02	-0.15616417E-03	-0.001
26.	0.34975432E 02	0.35019437E 02	0.35063443E 02	0.35019451E 02	0.12874603E-04	0.000
27.	0.43132234E 03	0.43136635E 03	0.43141035E 03	0.43136641E 03	0.64849854E-04	0.000
28.	0.24480431E 03	0.24484831E 03	0.24489231E 03	0.24484918E 03	0.86593628E-03	0.000
29.	0.17646369E 03	0.17650770E 03	0.17655171E 03	0.17650787E 03	0.16593933E-03	0.000
30.	0.57246737E 03	0.57251137E 03	0.57255537E 03	0.57251015E 03	-0.12207031E-02	-0.000
31.	0.14936358E 03	0.14940759E 03	0.14945159E 03	0.14940762E 03	0.26702881E-04	0.000
32.	0.98847330E 02	0.98891335E 02	0.98935340E 02	0.98891437E 02	0.10204315E-03	0.000
33.	0.24922225E 02	0.24966230E 02	0.25010236E 02	0.24966075E 02	-0.15544891E-03	-0.001
34.	0.15254248E 01	0.15694403E 01	0.16134558E 01	0.15693001E 01	-0.13016164E-03	-0.008
35.	0.37767440E 03	0.37771840E 03	0.37776241E 03	0.37771855E 03	0.14114380E-03	0.000
36.	0.87154925E 02	0.87198930E 02	0.87242936E 02	0.87198815E 02	-0.11539459E-03	-0.000
37.	0.55713254E 01	0.56153309E 01	0.56593364E 01	0.56152710E 01	-0.59962273E-04	-0.001
38.	0.27453209E 03	0.27457609E 03	0.27462009E 03	0.27457628E 03	0.13351347E-03	0.000
39.	0.18005994E 01	0.18446349E 01	0.18886704E 01	0.18445273E 01	-0.77679753E-04	-0.004
40.	0.46109568E-00	0.50510120E 00	0.54910672E 00	0.50497638E 00	-0.12482703E-03	-0.025
41.	0.11161944E 03	0.11166744E 03	0.11170745E 03	0.11166356E 03+	0.11634827E-03	0.000
41.	0.11161944E 03	0.11166744E 03	0.11170745E 03	0.11166356E 03	0.11634827E-03	0.000
42.	0.26835614E 03	0.26840015E 03	0.26844415E 03	0.26840033E 03	0.17547607E-03	0.000
43.	0.50353727E 03	0.50358128E 03	0.50362528E 03	0.50358115E 03	-0.12588501E-03	-0.000
44.	0.84419864E 01	0.84859920E 01	0.85299975E 01	0.84858480E 01	-0.14591217E-03	-0.002
45.	0.12033153E 03	0.12037554E 03	0.12041955E 03	0.12037566E 03	0.11444092E-03	0.000
46.	0.53568732E 02	0.53612737E 02	0.53656743E 02	0.53612783E 02	0.45299530E-04	0.000
47.	0.86818246E 02	0.86862251E 02	0.86906257E 02	0.86862237E 02	0.85830688E-04	0.000
48.	0.38071147E 03	0.38075547E 03	0.38079947E 03	0.38075562E 03	0.14877319E-03	0.000

MAXIMUM ABSOLUTE DEVIATION = 0.1220703E-02,(SEE OBS. NO. 33., LINE NO. 30)

MAXIMUM ABSOLUTE PERCENT DEVIATION = 0.025,(SEE OBS. NO. 40., LINE NO. 40)

POSTULATED CRITERIA

STANDARD ERROR OF Y = 0.5000000E 00
COEFF OF DETERMINATION = 0.9990000E 00

FITTED CURVE PROPERTIES

STANDARD ERROR OF Y = 0.4400552E-01
COEFF OF DETERMINATION = 0.9999999E 00

FITTED CURVE MEETS BOTH CRITERIA.

PROBLEM NO. 1 TERMINATED AFTER 4 PASSES.

```
  MAD EXTERNAL FUNCTION STATEMENTS FOR PREDICTING EQUATION PRODUCED BY LAST REGRESSION STEP

*  COMPILE MAD
*  PUNCH OBJECT
          EXTERNAL FUNCTION (X1)
          ENTRY TO EXAMPL.
          INTEGER I
          DIMENSION T( 3)
          T( 1) =   0.15000030E 02
          T( 2) =   0.39999986E 01
          T( 2) = T( 2) * X 1 .P. (   2)
          T( 3) =  -0.15999832E 02
          T( 3) = T( 3) * X 1
          T(0) = 0.
          THROUGH SUM, FOR I = 1,1,I .G.   3
      SUM T(0) = T(0) + T(I)
          FUNCTION RETURN T(0)
          END OF FUNCTION

  FORTRAN II SUBROUTINE STATEMENTS FOR PREDICTING EQUATION PRODUCED BY LAST REGRESSION STEP

*  COMPILE FORTRAN
*  PRINT SAP
*  PUNCH OBJECT
      FUNCTION EXAMPL(X1)
      DIMENSION T( 3)
      T( 1) =  0.150000304E 02
      T( 2) =  0.399999857E 01
      T( 2) = T( 2) * X 1 ** (   2)
      T( 3) = -0.159998320E 02
      T( 3) = T( 3) * X 1
      EXAMPL = 0.
      DO 1 I =1, 3
    1 EXAMPL = EXAMPL + T(I)
      RETURN

PRESENT STATUS OF SELECTOR ARRAYS.

INTERACTION ARRAY

INTERACTION NO.    WEIGHT      INTERACTION NO.    WEIGHT      INTERACTION NO.    WEIGHT      INTERACTION NO.    WEIGHT
      1          1.00000000E 00

SUM OF WEIGHTS =     1.0000000E 00

VARIABLE ARRAY

   VARIABLE NO.    WEIGHT         VARIABLE NO.    WEIGHT         VARIABLE NO.    WEIGHT         VARIABLE NO.    WEIGHT
      1          1.00000000E 00

SUM OF WEIGHTS =     1.0000000E 00

FUNCTION ARRAY

VARIABLE NO.  1

   FUNCTION NO.    WEIGHT         FUNCTION NO.    WEIGHT         FUNCTION NO.    WEIGHT         FUNCTION NO.    WEIGHT
      1          0.1239835E 01       2          0.7810472E 00       3          0.4959340E 01       4          0.7810472E 00
      5          0.7810472E 00       6          0.7810472E 00       7          0.7810472E 00       8          0.7810472E 00
      9          0.7810472E 00      10          0.7810472E 00      11          0.4920289E-00      12          0.4920289E-00
     13          0.4920289E-00      14          0.7810472E 00      15          0.1968115E 01      16          0.7810472E 00
     17          0.4920289E-00      18          0.7810472E 00      19          0.7810472E 00      20          0.4920289E-00

SUM OF WEIGHTS =     0.2000000E 02
```

BIBLIOGRAPHY

Automatic Programming and Artificial Intelligence

1. Theodoroff, T. J. and Olsztyn, J. T. DYANA: Dynamics Analyzer Programmer. General Motors Research Staff Publication, 1959.

2. Friedberg, Dunham, North, A Learning Machine. Parts I and II, Vol. 3, No. 1 and 3, IBM Journal Research and Development, 1959.

3. Minsky, M. L. Heuristic Aspects of the Artificial Intelligence Problem. Group Report 34-35, MIT Lincoln Laboratory, December, 1956.

4. Minsky, M. L. Artificial Intelligence and Heuristic Programming. Proceedings of the International Conference on the Mechanization of Thought, London, November, 1958.

5. Newell, A., Shaw, J. C. and Simon, H. A. Elements of a Theory of Human Problem Solving. Psych. Rev. 65.

Mathematics, Logic and Statistics

6. Schreirer, O. and Sperner, E. Modern Algebra and Matrix Theory. Chelsea Publishing Co., 1951.

7. Hartree, D. R. Numerical Analysis. Oxford at the Clarendon Press, 1958.

8. Hildebrand, F. B. Introduction to Numerical Analysis. New York: McGraw-Hill Book Co., Inc., 1956.

9. Cramer, H. The Elements of Probability Theory and Some of Its Applications. New York: John Wiley and Sons, 1959.

10. Fisher, R. A. Statistical Methods and Scientific Inference. Oliver and Boyd, 1956.

11. Davies, O. L. Statistical Methods in Research and Production. Oliver and Boyd, 1947.

12. Dallemand, J. E. Stepwise Regression Program on the IBM 704. General Motors Research Staff, GMR 199, 1958.

13. Weyl, H. Philosophy of Mathematics and Natural Science. Princeton University Press, 1949.

14. Tarski, A. Introduction to Logic. Oxford University Press, 1951.

15. Church, A. Introduction to Mathematical Logic. Princeton University Press, 1956.

16. Tintner, G. Econometrics. New York: John Wiley and Sons, 1952.

17. Klein, Textbook of Econometrics. Row, Peterson and Co., 1957.