AUTOMATIC COMPUTER GENERATION OF SOLUTION PROCEDURES TO LARGE SETS OF NONLINEAR SIMULTANEOUS EQUATIONS VIA GENDER

By

JAMES RICHARD CUNNINGHAM

A DISSERTATION PRESENTED TO THE GRADUATE COUNCIL OF THE UNIVERSITY OF FLORIDA IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

UNIVERSITY OF FLORIDA
1972

COPYRIGHT

JAMES RICHARD CUNNINGHAM

1972

9-2-73

ACKNOWLEDGEMENTS

The author wishes to express his indebtedness to the chairman of his supervisory committee, Dr. A. W. Westerberg, Associate Professor of Chemical Engineering, for suggesting the research topic and for providing the necessary guidance. The author also wishes to thank the members of the supervisory committee: Dr. F. P. May, Professor of Chemical Engineering, Dr. R. G. Selfridge, Professor of Mathematics and Director of the Computing Center, and Dr. F. D. Vickers, Associate Professor of Computer and Information Sciences.

Thanks also are extended to the National Science Foundation, which provided financial support through grant GK-18633. In addition to providing the funds for my research assistantship, this grant defrayed the extensive computer expenses incurred during the debugging of the 135 subprograms constituting the GENDER System.

Without this assistance and support, the development of GENDER would not have been possible.

TABLE OF CONTENTS

		Page
ACKNOWLEDGEMENTS		iii
LIST OF TABLES		vii
LIST OF FIGURES		ix
ABSTRACT		xi
CHAPTERS:		
I.	INTRODUCTION	1
II.	OVERVIEW OF THE CHEMICAL ENGINEERING DESIGN ROUTINES SYSTEM	5
III.	DATA MANIPULATION AND STORAGE	11
III.1.	Data Handling	11
III.1.a.	Lists	11
III.1.b.	Files	14
III.1.c.	Memory Allocation	14
III.2.	Special Data Structures	16
III.2.a	Networks	18
III.2.b	Sparse Incidence Matrices	18
III.3.	Implementation Details	22
*III.3.a.	COAST	22
*III.3.b.	REMOTE	33
*III.3.c.	NETPAC	43
*III.3.d.	SIMPAC	57
IV.	DATA BASE	66
IV.1.	Service Module, SECEDE	66
IV.2.	Solution Procedure, GENDER List	68

TABLE OF CONTENTS (Continued)

		Page
IV.3.	Input/Output Facilities	68
IV.4.	Implementation Details	70
*IV.4.a.	SECEDE	70
*IV.4.b.	SECIAO	85
*IV.4.c.	GENDER List	89
*IV.4.d.	GENIAO	98
V.	ANALYSIS ALGORITHMS	101
V.1.	Hungarian Output Assignment Algorithm	102
V.2.	Speed-Up Precedence Ordering Algorithm	103
V.3.	Barkley - Motard Minimum Tear Algorithm	104
V.4.	Implementation Details	105
*V.4.a.	HASSAL	106
*V.4.b.	SPEDUP	107
*V.4.c.	BEMOAN	112
VI.	INTERPRETATION	114
VI.1.	Link Editor	115
VI.2.	Converter	116
VI.2.a.	SOLVE	119
VI.3.	Interpreter	120
VI.3.a.	MATH	122
VI.3.b.	Subroutines	122
VI.4.	Implementation Details	125
*VI.4.a.	LINKED	125
*VI.4.b.	COVERT	126
*VI.4.c.	GLINT	128

TABLE OF CONTENTS (Continued)

		Page
VII.	USER MANUAL	130
	VII.1. Input Data Preparation	130
	VII.2. Program Preparation	131
	VII.3. Additional Algorithms	134
	VII.4. GENDER as a Design Tool	134
VIII.	EXAMPLES	137
	VIII.1. A Simple Analysis Strategy	147
	VIII.2. Equilibrium Flash	147
	VIII.3. Binary Distillation	155
IX.	CONCLUSION	165
	IX.1. Convenience Aspects	165
	IX.2. Additional Algorithms	166
	IX.3. A Final Note on GENDER	168
APPENDICES		169
	Appendix A	170
	Appendix B	498
	Appendix C	501
	Appendix D	504
BIBLIOGRAPHY		510
BIOGRAPHICAL SKETCH		511

LIST OF TABLES

Table		Page
1	COAST Verbs	28
2	LEND Components Required by REMOTE	42
3	Auxiliary Data Fields Required by NETPAC	44
4	NETPAC Verbs	55
5	Data Fields Required for Ordinate Entries	59
6	Data Fields Required for SIM Elements	61
7	Information Block Components Pertinent to the SIM	62
8	SIMPAC Verbs	64
9	Data Fields for SECEDE	75
10	File Entries for SECEDE	77
11	SECEDE Files	79
12	Information Block Components Pertinent to SECEDE	86
13	Data Fields for GENDER	90
14	The List Entries of GENDER	93
15	Information Block Components Pertinent to GENDER List	99
16	Data Fields for SPEDUP Working Lists	109
17	List Entries on SPEDUP Working Lists	110
18	Data Fields for the BEMOAN Working Ordinate	113
19	Operator Set	121
20	GLINT Error Codes	123
21	Information Block	132
22	The Code Name/Variable Assignments for a Distillation Tray	140
23	The Constant/Code Name Assignments for a Distillation Tray	141

LIST OF TABLES (Continued)

Table		Page
24	Values for Information Block Components.....	149

LIST OF FIGURES

Figures		Page
1	GENDER Architecture	9
2	Typical List Structures	12
3	Typical Hierarchical File	15
4	A Simple Network	19
5	Sparse Incidence Matrix	20
6	A User Mask	24
7	PUSH	30
8	POPUP	32
9	File Structure	34
10	A Record	38
11	Trace (Start)	47
11a	Trace (Step 1)	48
11b	Trace (Step 2)	49
11c	Trace (Step 3)	50
11d	Trace (Step 4)	51
11e	Trace (Completed)	52
12	Sparse Matrix	58
13	An Equilibrium Stage	138
14	SECIAO Data for a Distillation Tray	142
15	Data for COIN	148
16	Equilibrium Flash	150
17	SECIAO Data for Equilibrium Flash	153

LIST OF FIGURES (Continued)

Figures		Page
18	Data for VARIAO.....	154
19	Simple Distillation Column.....	156
20	SECIAO Data for Simple Distillation Column.....	158

Abstract of Dissertation Presented to the
Graduate Council of the University of Florida in Partial
Fulfillment of the Requirements for the Degree of Doctor of Philosophy

AUTOMATIC COMPUTER GENERATION OF SOLUTION PROCEDURES TO LARGE SETS OF NONLINEAR SIMULTANEOUS EQUATIONS VIA GENDER

By

James Richard Cunningham

December, 1972

Chairman: Arthur W. Westerberg
Major Department: Chemical Engineering

Fundamentally, chemical process plant design is the solution of a large set of nonlinear simultaneous equations. For large and complex equation sets, it is frequently not possible for the design engineer to determine readily how the solution should proceed. Algorithms have been developed which analyze equation sets with respect to solution procedure. The algorithms do not generally attempt to solve the equation set, but merely provide the directions for solving to the design engineer. While computer implementation of certain algorithms analyzing equation sets has been accomplished, little emphasis seems to have been placed on the implementation of a complete analysis system integrating a number of algorithms. This dissertation presents an automatic and machine-independent system permitting the design engineer to direct solution procedure generation and execution via selections from a spectrum of analysis algorithms.

GENDER is the acronym for General Engineering Design Routines. The primary objective of GENDER is the development of a complete solution procedure. To be complete, a solution procedure must reflect (1) decision variable selection, (2) output set assignment, (3) tear variable selection

and (4) precedence ordering. These attributes are provided to a solution procedure by GENDER through a minimum tear analysis strategy. While only the minimum tear strategy has been furnished with the first version of GENDER, the GENDER System has been designed to readily accept additional analysis algorithms, expanding the analysis strategy repertoire.

The design engineer, having generated a solution procedure for an equation set, may perform the translation of the solution procedure into FORTRAN for execution. However, an interpreter capable of direct execution of a solution procedure is provided in GENDER. The interpretation capability, while sparing the engineer from the drudgery of translation, has a somewhat more noble purpose. If the design problem involves constrained optimization, the solution procedure will be subjected to a series of revisions as additions or deletions to the set of binding constraints are required. The optimization-solution procedure generation-execution cycle is potentially faster for interpretation than for FORTRAN execution owing to the elimination of the translation and compilation steps.

GENDER depends extensively upon list processing techniques. In fact, the solution procedure has been implemented as a list. In the list format, the revisions and rearrangements associated with an analysis strategy may be conveniently performed on a solution procedure. The application of list processing techniques is not restricted to the solution procedure alone, but permeates the GENDER System. Linked data structures of note include a paged storage system (REMOTE) and the sparse incidence matrix. The former is notable for compatability with mass memory and the latter for memory conservation (on machines lacking virtual memory).

CHAPTER I

INTRODUCTION

The objective of process design is the determination of equipment specifications and operating conditions which will meet certain production goals developed by management. Process design problems are not particularly amenable to solution. The equations quantitatively describing the equipment characteristics may be highly nonlinear, a situation further confounded by the presence of recycle streams. The recycle stream is necessary to economic operation, but it imparts a cyclic character to the design calculations just as it imparts a cyclic character to the process flow. To these complexities of process design, yet another must be added. Processing plants are frequently complex entities, consisting of many individual processing units connected by a maze of piping. This is particularly true if the process engineer intends to extend the scope of the plant model to include not just the major features of the process, but all unit operations. That is, the mathematical model is to resemble the processing plant as closely as possible.

The availability of large, powerful computers has caused process engineers to revise the design procedure. Prior to his introduction to computers, the engineer was forced to simplify the process model to facilitate the completion of hand calculations in a reasonable amount of time. While short-cut techniques were developed for some unit operations, the "assumption" reigned as the arme supreme. Unlike the human brain,

the digital computer is well suited to complex and repetitious calculations. The engineer could now expect solutions for large and complex process models without the uncertainties of simplification.

The new exactness permitted to the process engineer by the digital computer proved to be something less than a complete blessing. The programs developed by the engineer from the mathematical models all too frequently either failed to converge, giving no answer, or generated unreasonable answers. Research, both industrial and academic, determined that the computational difficulties were characteristic of particular formulations of a model. Further, certain formulations of a model may require more information to initiate cyclic calculations than other formulations. What had developed at this point was a new field of endeavor for process engineers: the study of process models. This study begins where the traditional concept of mathematical modeling ends.

The result of mathematical modeling is the development of an equation set descriptive of, for instance, a unit operation. This set of equations is not unique. That is, the algebraic rearrangement of one model produces another equivalent model. It is this rearrangement and its effects which have captured the attention of the process engineer. The result has been a variety of algorithms for selecting a particular model formulation.

The algorithms are rather specific in character, dealing with only a particular aspect of a process model. Algorithms have been developed for selecting the output variables for a set of equations, for selecting the variables for which value estimates are required to initiate cyclic calculations and for ordering the equations so that the calculation of each variable value will depend only upon variable values already calculated. These operations are respectively known as output set

assignment, tear variable selection and precedence ordering. The process engineer employs selected algorithms sequentially to convert the original model he formulated into an equation set amenable to computer solution. The particular selection of algorithms may be referred to as an analysis strategy. The strategy is largely dictated by the objectives of the design engineer, such as a minimum tear solution or a rapidly converging solution. Unfortunately, the execution of an analysis strategy must be primarily manual. Only a few algorithms have been programmed, all of which are machine dependent and isolated from other analysis capabilities.

The next step in this evolutionary development of process design would seem to be totally automatic process model analysis. The notion of automatic design is not original to this work (Kevorkian and Snoek, 1972; Soylemez, 1971; Mah and Rafal, 1971), but has been attempted previously. It is the limitations of other automatic design systems which have encouraged the present effort, the General Engineering Design Routines (GENDER) System. GENDER does not restrict the design engineer to a single analysis strategy by virtue of a library of analysis algorithms. Further, the inclusion of additional algorithms has been made relatively easy. While it is possible to convert the product of an analysis strategy, which we shall call a solution procedure, into FORTRAN code for execution, the GENDER System provides the capability of directly interpreting the solution procedure to produce a solution to the design problem. As a feature designed to permit compact problem representation, GENDER can accept indexed variables and equations. Finally, the GENDER System affords the design engineer with a problem solving medium permitting the convenient modification of the original design problem during the execution of the analysis strategy and/or interpretation of the solution procedure. This feature is essential to constrained

optimization where fluidity in the inclusion and exclusion of constraints is an absolute necessity. While many of these features have appeared in previous attempts at automatic design packages, they have to date not all appeared in a single automatic design system.

CHAPTER II

OVERVIEW OF THE GENERAL ENGINEERING DESIGN ROUTINES SYSTEM

The heart of the GENDER System is the solution procedure and is essentially the set of instructions for solving a set of simultaneous equations. In order to make possible direct machine interpretation of the solution, the solution procedure must be explicit and complete. The purpose of this chapter is to introduce the concept of the solution procedure, its terminology and its relation to the GENDER System.

In Chapter I and in the preceeding paragraph the term equation was used for a model constituent. To be more precise, the model constituents are actually equations and solved equations. That is, the model is composed of algebraic relations between variables and constants, but having the form

$$f(x,y,z,\ldots) = 0$$

rather than the solved equation form

$$x = F(y,z,\ldots)$$

In the solved equation form, the x is called the output variable. An equation may be transformed into a solved equation by selecting a variable to be the output and performing the algebraic rearrangement necessary to place this variable alone on the left-hand side of the equality.

Certain of the equations defining a processing plant may be constraints. Normally one thinks of constraints as inequalities. However, inequalities are readily converted to equalities via the intro-

duction of slack variables. GENDER, at least in its current version, is prepared to manipulate equality constraints only. Being identical to the equations describing the process, the constraints could be grouped with them without differentiation. However, during constrained optimization the set of active constraints varies in numbers and composition. To permit the free inclusion (exclusion) of constraints into (from) a problem, constraints are regarded as distinct model constituents and are carefully differentiated from all other model components.

Normally an equation is permitted but a single output variable, with one exception. The external routine (ER) is a special model constituent which may possesses more than one output variable. In essence, the ER is a subprogram for determining values to output variables given the values of the input variables. An ER may be encoded in any acceptable programming language.

To facilitate the incorporation of existing subprograms in a plant model, a mechanism for indicating prechosen output variables is essential. The mechanism selected for GENDER is a weighting scheme with an integer scale of 0 to 9. Each variable in each equation and ER is assigned a weight in the 0 to 9 range. This weight is the cost of assignment as an output variable. Thus, for an existing subprogram, the output variables are all weighted at 0 while the input variables are all weighted at 9. This weighting scheme allows the design engineer to indicate to GENDER his preferences with respect to output set assignment. For instance, similarity of a new model with one or more previous models may suggest at least a partial output assignment. The weighting mechanism allows the engineer to guide GENDER in performing the initial output assignment and may reduce execution time depending on the ultimate

degree of similarity between the solution procedures for the models.

In addition to output set selection, analysis of the equation set must also order the equations and, if possible, subdivide the functions into smaller groups. Initially, the plant model is treated as a single large group of equations. The decomposition of the large problem into a multitude of smaller problems is obviously advantageous. Equations which are acyclic in character may be grouped, although the equation grouping normally reflects the presence of a cyclic character. It is permitted by GENDER to have groups as members of groups. Thus, we may have, for example, a cyclic group appearing as a constituent of another cyclic group, or even as a constituent of an acyclic group.

Once tear variable selection has rendered all cyclic groups apparently acyclic (by ignoring all occurrences of the tear variables except in the equations where they are the output), it is possible to order all members of groups and to order all of the groups so that the information flow is strictly forward. That is to say, the value of a variable is never required as the input to a equation before it has been calculated as the output of a previous equation. This then is the concept of precedence ordering.

The solution procedure must reflect the output set assignment, grouping, tear variable selection and precedence ordering. The solution procedure must also indicate decision variable selection, the decision variables being the unassigned variables following output set selection. Numerous strategies exist for acquiring these ingredients to the solution procedure, but for completeness all must explicitly appear.

The solution procedure is called the GENDER list. The GENDER list consists of group, each group possessing a body of equations, ER's or groups. Each equation and ER appears once somewhere on the GENDER list.

Its position on the GENDER list is dictated by precedence ordering. Space is reserved with each equation and ER to indicate the selected output variables. Similarly space is reserved for the recording of the decision and tear variables associated with each group. The completed solution procedure will generally be evolved from a skeletal, random listing of the equations in a single group called the crude GENDER list.

The completed solution procedure contains all of the instructions for solving the equation set, but is as yet not amenable to interpretation. The conversion operation transforms equations into solved equations via algebraic arrangement. In lieu of destroying the original equation set, the converter operates on a copy of the equations. This feature of GENDER preserves the original model should reanalysis of all or a portion of the solution procedure be required. This eventuality might, for instance, be realized if convergence difficulties of a cyclic group are encountered. Once subjected to conversion, the solution procedure is transformed into an ordered and grouped list of solved equations ready for numerical evaluation.

The GENDER System is divided into five major program levels as shown in Figure 1. The program packages COAST (level 1), REMOTE (level 2), SIMPAC and NETPAC (level 3) are discussed in Chapter III. These packages provide the capacity to manipulate certain data structures essential to GENDER. Chapter IV introduces the data base SECEDE and provides greater detail on the GENDER list. The algorithms available for problem analysis are discussed in Chapter V. Level 4 of GENDER is devoted entirely to the analysis algorithms. The highest current GENDER level is level 5. On this level is found the input/output

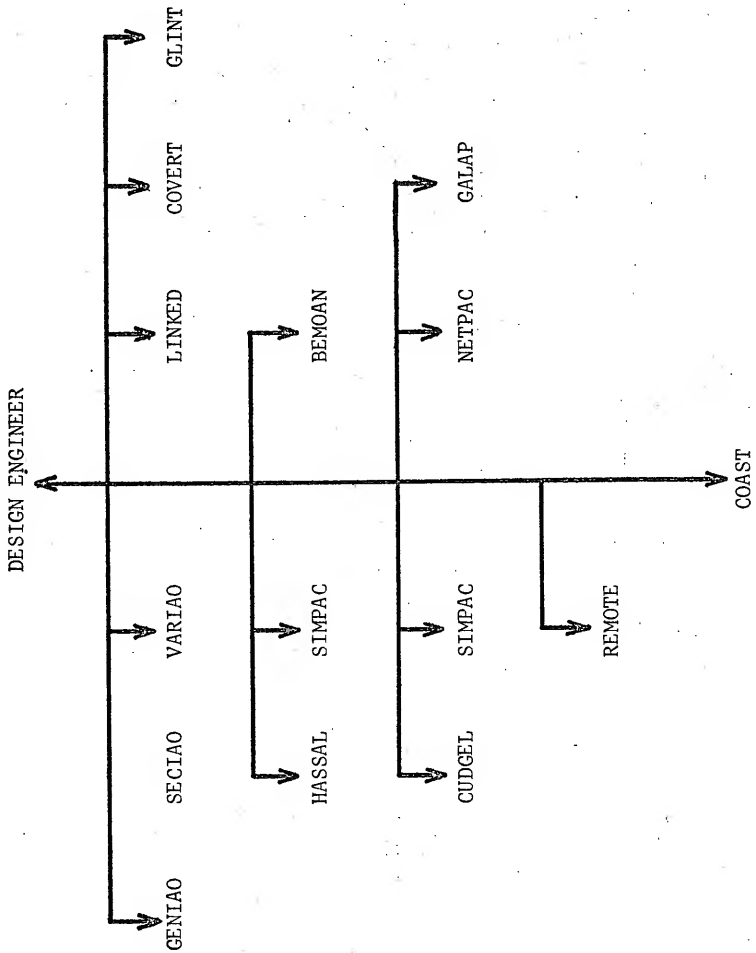

Figure 1. GENDER Architecture.

facilities and the programs relative to interpretation of the GENDER list. The description of the input/output facilities has been incorporated into Chapter IV. The notion of GENDER list interpretation is treated in Chapter VI. Chapter VII is a brief user's guide to the GENDER System. In this chapter the expansion of the repertoire of analysis algorithms is discussed, as well as the employment of GENDER as a design tool. Illustrative problems are presented in Chapter VIII. This work is concluded in Chapter IX with some remarks on the limitations and possible future of GENDER.

Particularly in the next four chapters, some sections are extremely detailed. In addition to being difficult to read, they are likely only of benefit to readers contemplating algorithmic additions or modifications to the GENDER System. For this reason, these sections are identified by an * in the section number and may be omitted by readers not requiring the implementation details of GENDER.

CHAPTER III

DATA MANIPULATION AND STORAGE

III.1. Data Handling

The algorithms for solution procedure development suggest certain data structures. Effective implementation of the algorithms requires the availability of these data structures. Unfortunately, the data structures require vast quantities of on-line memory. A large scale problem would, by virtue of the data structures developed during problem analysis, very quickly reach the bounds of on-line memory. Fortunately, not all of the data structures need exist at one time, making feasible the sharing of memory. A priori, the memory required by each phase of problem solving is unknown. Thus, the simple partitioning of memory is an infeasible sharing policy.

Two basic and complementary data storage mediums have been provided: lists and files. The list permits complex data structures which may be readily rearranged, expanded or contracted. The penalty for this flexibility is high access times. Reaching a particular location may be accomplished only by stepping through all preceding list locations. Conversely, the file allows rapid random access of any location. Files are, however, restricted to a strictly sequential structure. Memory sharing is accomplished by a dynamic memory allocation procedure. For readers not familiar with list processing, Knuth (1968) provides a complete discussion of list processing principles and structures, including memory allocation.

III.1.a. Lists

Figure 2 is an illustration of three possible list structures. As

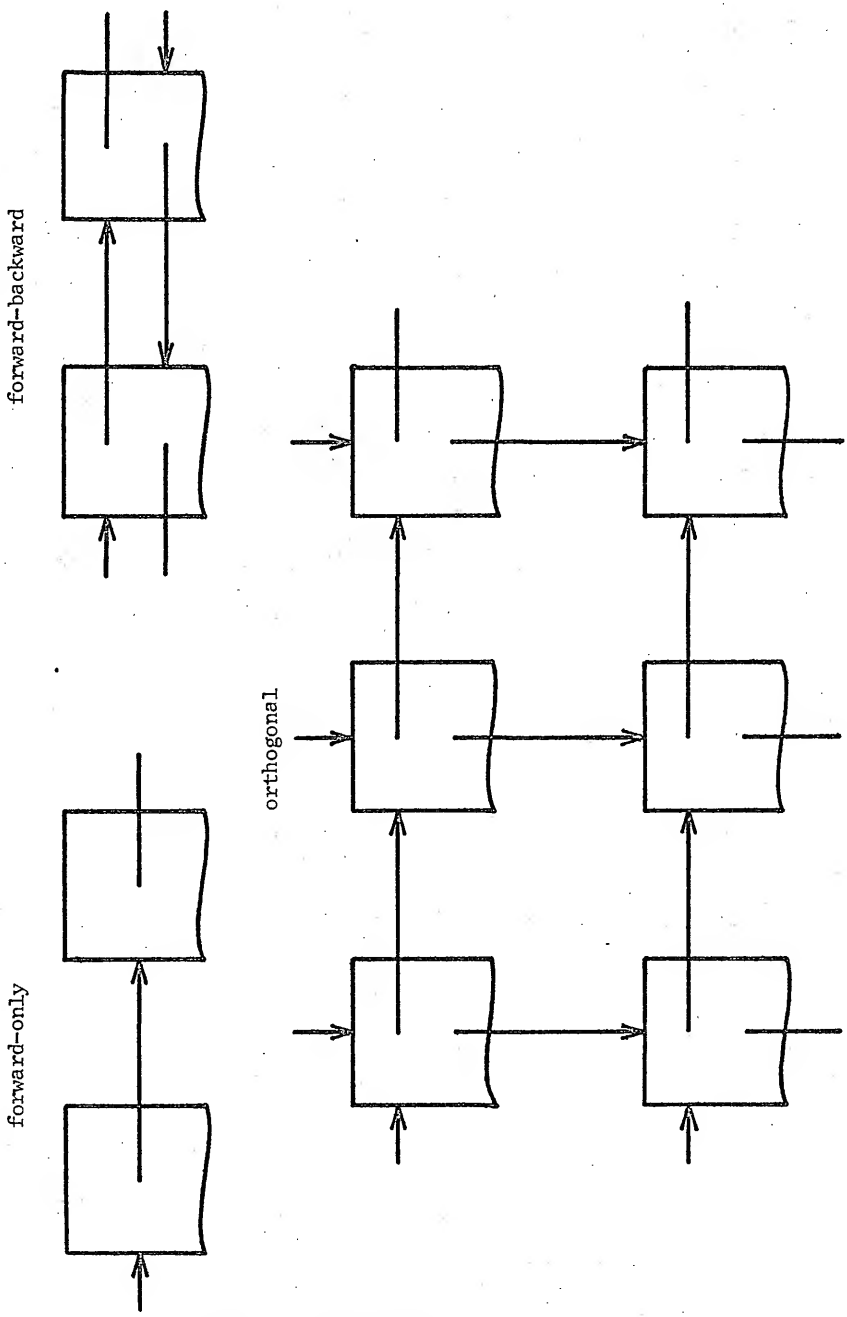

Figure 2. Typical List Structures.

indicated pictorially in Figure 2, the structure of a list is relatively arbitrary. The arrows in Figure 2 represent the connection between list entries and are called links. Thus one data item (at least) in a list entry is a pointer to an adjacent entry on the list. This flexibility has not been restricted to structure alone. Each entry on a list possesses the attributes of size, quantity of data and organization of the data within the list entry. All of these attributes are also arbitrary.

The structure of a list, while arbitrary, is specified by the programs which generate and manipulate the list. That is, the degrees of freedom permitted in the selection of a list structure are consumed by the programmer of a list processing application. Since one of the objectives was machine independence, and since word size and memory size vary from machine to machine, it seemed necessary to segregate the organization of data within list entries from the tailoring of list structures to meet strictly programming requirements. This has been accomplished by designing the list processor to accept the data organization parameters as data prepared and supplied by the user. These parameters are further discussed in Chapter VII.

The utility of a list processor lies mainly in the usefulness of its operations. We shall call the subprograms providing these operations "verbs." Verbs have been provided for obtaining and releasing list entries and for making and breaking the links between list entries. These verbs form the nucleus of the structure manipulation power of the list processor. Another set of verbs provides the capability for transferring information to or from a list entry. Of great utility is the ability to search a list for the occurrence of a particular set of data values. A verb providing this service is also included in the list processor.

III.1.b. Files

Figure 3 is an illustration of a typical hierarchical file. The arbitrariness of structure present in lists has, for files, been reduced to the specification of the number of levels in the hierarchy. Rapid access to randomly selected entries at any level is the incentive for the sacrifice of arbitrariness in structure. Owing to the sequential storage of file entries in on-line memory, large contiguous blocks of memory allocation may be required. It is, therefore, vital that the file system utilize core effectively. However, the sequential nature of the file entries makes possible the transfer of file data to mass memory devices. This enables the release of on-line memory normally occupied by files during problem solution phases not relying on the file data. Further, even when file data is required, it is possible to retrieve only portions of a file.

A file entry is essentially the same as a list entry, except that the explicit specification of links, as in the list entry, is absent. The organization of data within a file entry is accomplished identically as for a list entry. Thus, the file entry shares the same arbitrariness of data organization provided for the list entry.

Many fewer verbs need be provided for file manipulations than for lists since link manipulations are not required. In fact, three verbs are sufficient. One verb is required to provide a file entry and transfer data into it. A second verb provides the capability for retrieval of data from a file entry. Finally, the third verb permits the searching of a file for a particular set of data values.

III.1.c. Memory Allocation

The particular memory allocation strategy adopted for GENDER continues the theme of flexibility and arbitrariness. Each phase of

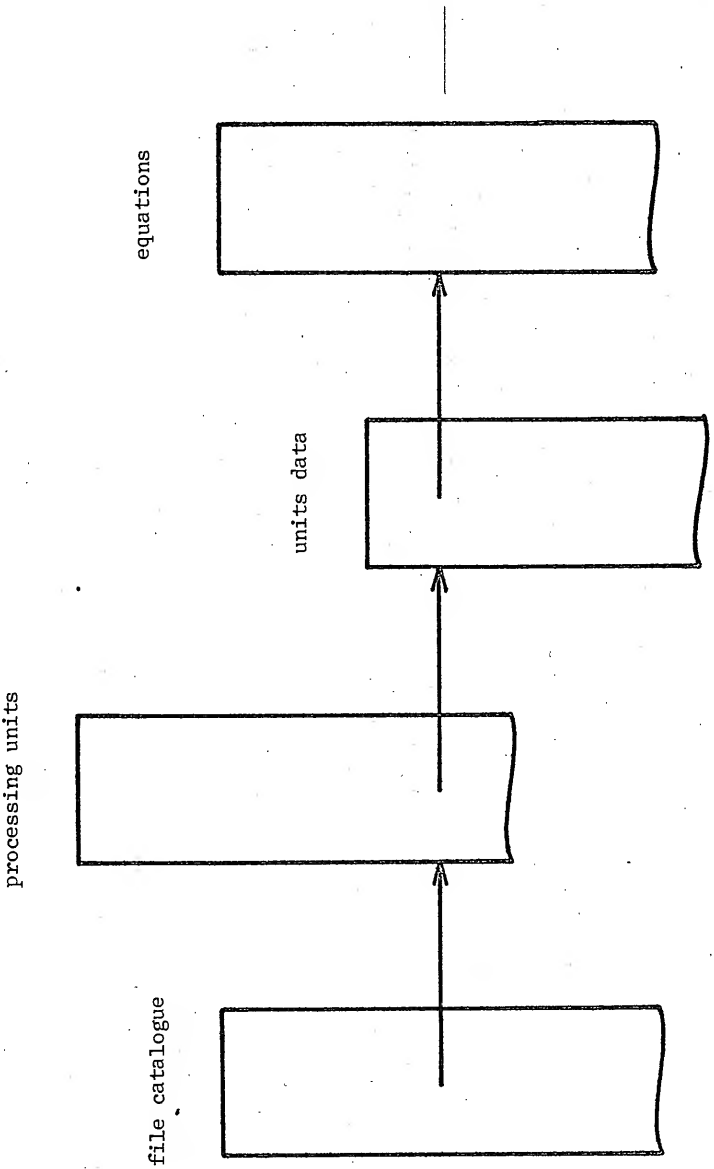

Figure 3. Typical Hierarchical File.

problem solving is allocated memory only as it is required. An accounting mechanism keeps track of the memory assigned to each phase. This mechanism permits the release of memory assigned to a particular phase once its task is complete. The released memory may then be reallocated to another phase of problem solving. The allocation, release and re-allocation capabilities give memory the fluid character essential to providing a time varying memory distribution in response to the time varying demands of the solution phases.

The dynamic allocation of memory provides an opportunity for fracturing memory into small isolated segments. This eventually would prevent the servicing of a request for a large continuous allocation of memory. Though procedures such as garbage collection have been proposed for reclaiming the isolated memory, these procedures have not been used for GENDER. The need for large blocks of memory has been avoided by designing the file system so as to utilize only reasonably sized continuous blocks of memory.

III.2. Special Data Structures

Networks - Since the algorithms for solution procedure development in many instances specify rearrangements of the solution procedure, a list structure was adopted. Hence, the solution procedure became known as the GENDER list. During the course of evaluation of the entries on the GENDER list via the interpreter, an anomaly may be encountered necessitating reanalysis of a portion of the GENDER list. In fact, all of the GENDER list from the anomaly to the list end is subject to reanalysis. Consider the following set of equations.

$$\begin{aligned} \underline{Y}(\underline{y},\underline{u}) &= \underline{0} \\ \underline{Z}(\underline{z},\underline{v}) &= \underline{0} \end{aligned} \tag{1}$$

If reanalysis of $\underline{Y}$ is required, $\underline{Z}$ would be subject to reanalysis,

although reanalysis of Z would result in no change. Instead of the list structure, suppose a structure is substituted in which each entry precedes only those entries that it influences. Then, Y and Z would appear on parallel paths since they are independent. This structure is a network, and was adopted for the solution procedure (which is still called the GENDER list).

Sparse Incidence Matrices (SIMs) - Incidence matrices provide a convenient and lucid medium to express the problem to which an algorithm is to be applied. In fact, many algorithms for solution procedure generation are most easily explained in terms of their effect on incidence matrices. This is to suggest that incidence matrices may be a natural and effective tool in the implementation of these algorithms. Unfortunately, matrices are relatively expensive in terms of memory requirements. This is particularly damaging in light of the intention to construct GENDER so as to be applicable to large scale problems. In chemical engineering, and perhaps in other disciplines as well, most equations contain only a small number of variables compared to the total number of variables in the problem. The average incidence for several chemical engineering applications examined was approximately three to four variables per equation. The majority of the incidence matrix elements are null. By considering the matrix elements to be list entries, and by disregarding all null elements, the use of incidence matrices is made possible without the disadvantage of excessive memory requirements. The elements form the orthogonal list structure shown previously in Figure 2. For example, if an equation contained only two variables, then the row would be composed of only two list entries. The columns represent variables so that each of the two row list entries are also members of the orthogonal column lists as well.

III.2.a. Networks

A network is an extension of the forward-backward list structure to permit each list entry more than a single forward and a single backward link. The capacity for parallelism is furnished by auxiliary lists called divergers. All list entries other than diverger entries are referred to as network nodes. Figure 4 depicts a node followed by three parallel nodes.

Networks have virtually all of the attributes associated with a forward-backward list structure. The arbitrariness of data organization within the nodes is somewhat restricted by the requirement that certain data entries be present as an integral part of the structure. Highly parallel structures may suffer with respect to access times as a result of large divergers.

The network verbs implemented are essentially concerned with the peculiarities of network linkages. The list processing verbs for data transfer serve equally well for nodes as for list entries. Subprograms are provided for the connection and for the separation of nodes. A complementary pair of function subprograms permits the extraction of linkage data, one for the forward link and one for the backward link. The policy of providing the capability for searching list structures is continued here for networks. Finally, a rather special verb is provided enabling an orderly, predictable tracing from node to node through a network. This verb, a function called NEXT, has proven to be of considerable utility.

III.2.b. Sparse Incidence Matrices

Sparse incidence matrices are actually a hybrid of the orthogonal list and of the file data structures. Figure 5 illustrates a sparse

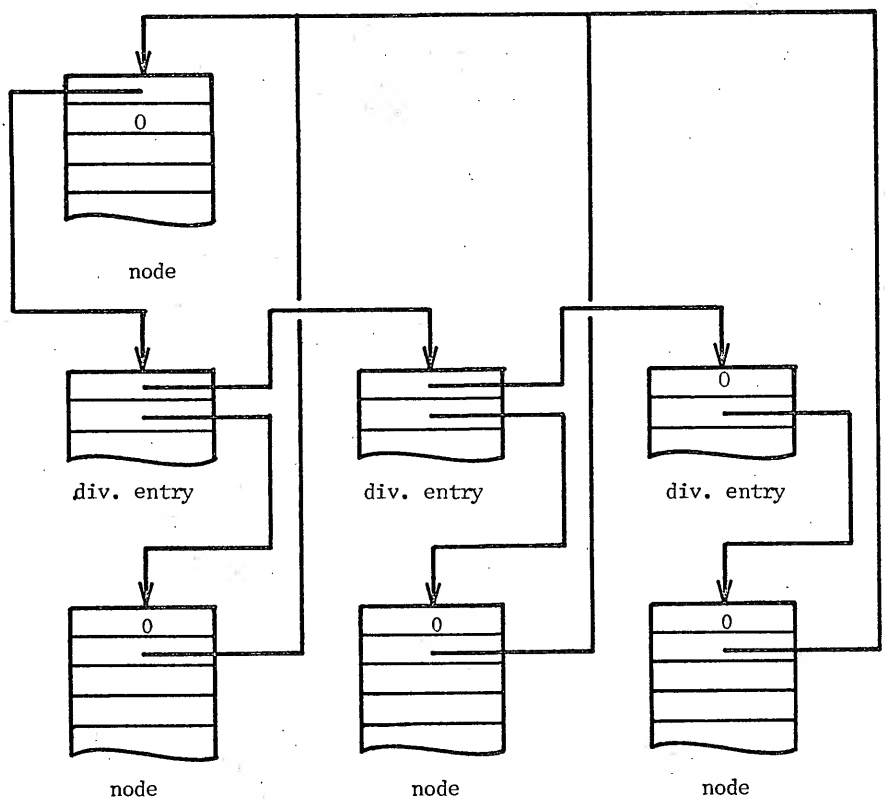

Figure 4. A Simple Network.

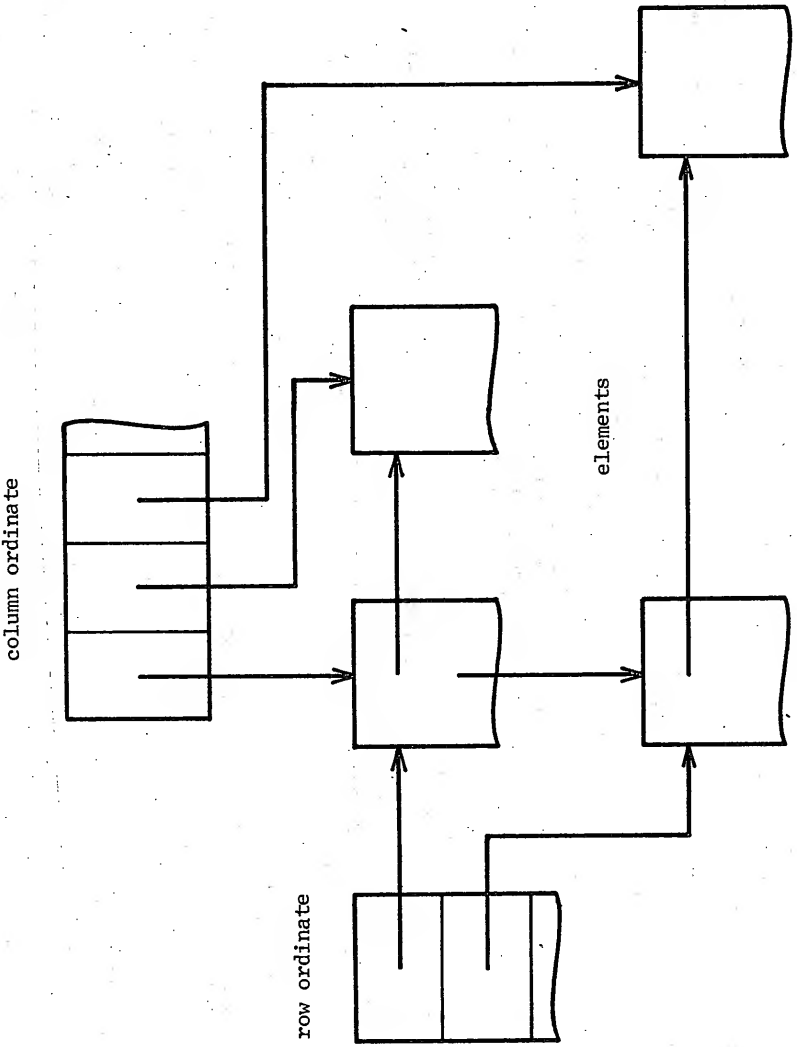

Figure 5. Sparse Incidence Matrix.

incidence matrix consisting of two equations and three variables. The files required for the sparse incidence matrix are termed ordinates. Each file entry contains a pointer to a row or column of the matrix. For simplicity, when reference is made in general to a row or column the term vector will be employed.

Even with the inclusion of the two ordinates, the memory conservation remains substantial. Consider a system of 1,000 equation in which 1,000 variables are incident. The resulting incidence matrix must contain no fewer than 1,000,000 words of memory if stored as a standard two dimension array. Allowing three words per non-null element (for the horizontal link, vertical link and data item), the elements of a typical sparse incidence matrix would consume only about 9,000 words. The ordinates and supporting data structures would require at most approximately 30,000 words. Thus, the memory savings afforded by the sparse incidence matrix exceeds 960,000 words over the simple two dimensional array. If the ability to pack information into list and file entries is utilized to its fullest extent, the size of the sparse incidence matrix may be reduced by as much as 50% depending upon the word length. The penalty for memory conservation is increased access time. A matrix vector may be accessed randomly and relatively quickly by virtue of the ordinates. Reaching a particular element list entry, however, can be accomplished only by tracing the selected vector from element to element until the desired element is accessed.

The nature of the subprograms provided for the manipulation of sparse incidence matrices are rather specific in character. Each subroutine is designed to fulfill the requirement by one or more of the algorithms for a certain sparse incidence matrix operation.

III.3. Implementation Details

The implementation of the data handling facilities discussed in Sections III.1 and III.2 has been partitioned into four program packages. COAST is the name given to the program set providing the core allocation and list processing capabilities. The subroutines furnishing the file manipulations constitute the REMOTE system. NETPAC and SIMPAC are respectively the network and the sparse incidence matrix program sets. Each of these packages are discussed in detail in the subsections to follow.

The remainder of Chapter III may be omitted by any reader not requiring a knowledge of the programming details. A knowledge of these details is not expected to be generally advantageous except to those readers planning additions to the algorithm library, alterations to an existing algorithm, or alterations to the GENDER facilities.

* III.3.a. COAST

To achieve machine independence, FORTRAN was selected as the programming medium. To provide the space required for the development of list structures, a COMMON declaration was employed to reserve a vector named ALLOC. The length of ALLOC is a user specified parameter. The position of a word of memory within ALLOC is referred to as the absolute address, or simply address, of that word. For example, the fifth ALLOC word has an address of five.

The memory allocation strategy adopted for GENDER is the buddy system. To reduce the administrative effort required in supervising the allocation of ALLOC to user programs, ALLOC is partitioned into multi-word segments. Segment size is also an adjustable parameter. To permit the allocation of space in sizes other than a single segment, consecutive segments may be combined to form buddies. In fact, two contiguous blocks of memory

of equal size may be combined to form a buddy. Requests for space are satisfied by supplying the smallest buddy satisfying the request. Should only a larger than necessary buddy be available, a request for a lesser amount of space may be satisfied by first fracturing (splitting into two parts) the larger buddy. All of the available buddies of each size are linked together forming an available space list for each buddy size. The operations necessary for memory management are provided by the subprograms C8COMB and C8GIVE.

In order to facilitate the release of ALLOC segments by user programs, an accounting system is necessary. The accounting device adopted is a mask. Each bit of the mask represents an ALLOC segment. A set bit in a user's mask indicates that the corresponding segment is assigned to that user. Each user program is assigned a user mask on its initial request for space. Figure 6 is an illustration of a user mask. A similar mask, established before allocation commences, is provided to record segments released by user programs. Whenever the buddy available space lists become sufficiently depleted that a request for core cannot be honored, then the space represented by the return mask is re-buddied and returned to the available space lists. This strategy eliminates execution of the costly re-buddying process unless it is absolutely necessary. CORN is the program furnished to permit the release of ALLOC segments by a user. The program LIST accomplishes the update of a user mask as memory segments are made available to a user program.

A user program requests space via a five word vector called a space utilization record (SPUR). The first word of the vector contains the address of the user mask. For the initial space request, this word must contain the integer zero. The second word will contain the pointer to

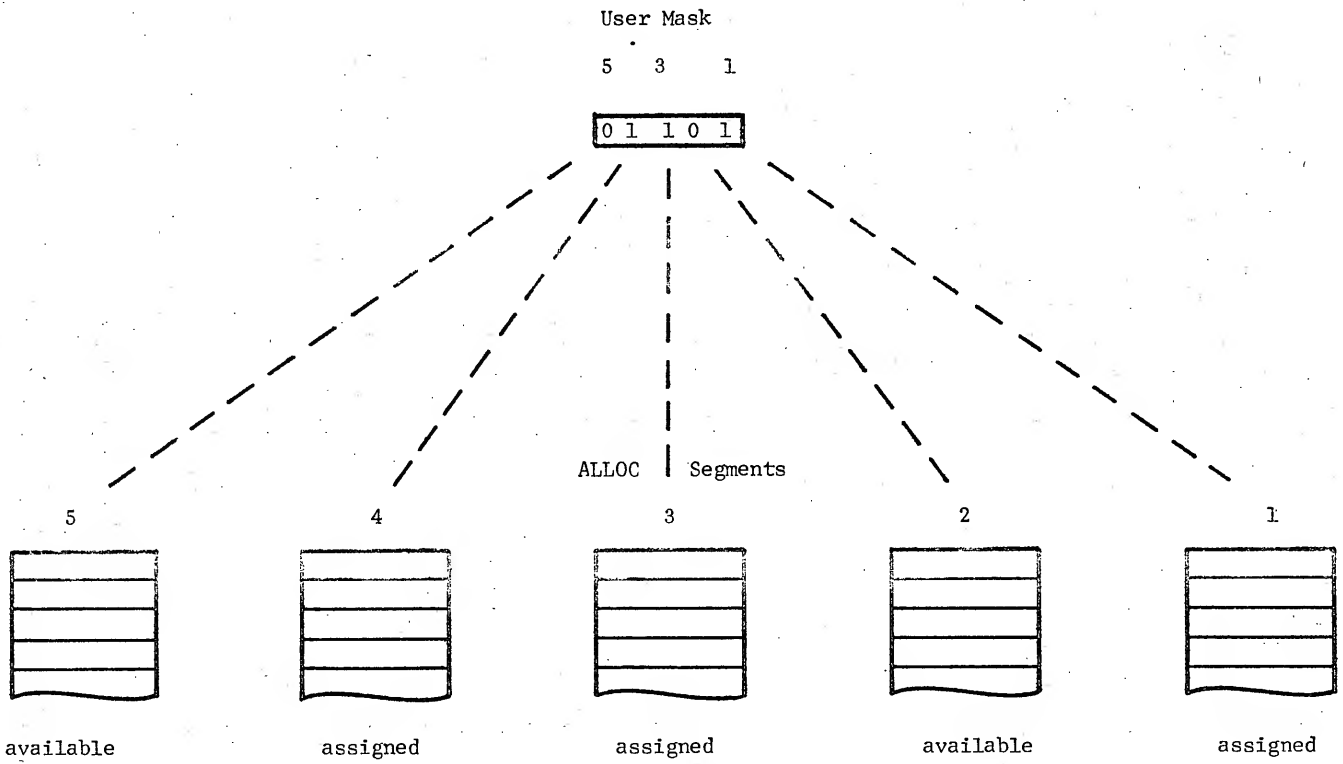

Figure 6. A User Mask.

the space provided during allocation. The third and fourth vector components are respectively the quantity and the size in words of the requested list entries. The list entries are manufactured from a buddy of the appropriate size and provided to the user program as an available space list. The last component of the SPUR indicates the list type, a subject to be considered later in this section. Several programs requiring ALLOC space may share a SPUR vector, thereby sharing the allocated core.

Information is contained within list entries in data fields. A data field consists of either all or a portion of a list entry word. Data fields consisting of only a few bits are usually referred to as flags. A data field appearing in one or more consecutive list entry words constitutes a data type. The collection of all data types associated with the entries of a particular list structure constitutes a list type. The quantitative specification of a data type requires four parameters. The first two delimit the bounds of the data type within a list entry. For example, if these parameters are valued at 1 and at 4, the data fields of this data type appear in words 1 through 4 of the list entry inclusively. The second pair of parameters delimits the bounds of the data field within a list entry word. The third word, called a shift, is the integer by which a list entry must be divided to place the first bit of the data field in bit position 1 of the dividend. For instance, if the data field begins in bit position 3, dividing by a shift of 4 will translate the field the required two positions. The last parameter is a field mask comprising set bits indicating the bit pattern of the data field. The field mask should be right justified to the first bit position of its single word. The collection of all sets of these four parameter vectors constitutes

the list entry definition (LEND) specifying the organization of data fields within the list entries of a list type. Thus, a LEND is required for each data type. COAST utilizes the subroutine COIN to accept the LEND's from a user as data. COIN is the only COAST routine requiring data.

In order to accomplish the transfer of information to or from the data fields within a list entry, the data fields involved in the transfer operation must be uniquely identified. It is to be permitted that a single transfer operation may involve many data fields from many different data types. Thus, the transfer operation requires two vectors, one reserved for the values of the data fields and a second to identify the data fields. The latter vector consists of three words per data type involved in the transfer, plus an additional word specifying the number of data types. Each three word set comprises a data type, the occurrence of the first field for transfer and the occurrence of the last field for transfer. For example, a three word set (4, 2, 7) indicates that the data type is 4, the first field transferred will be the second and the last field transferred will be the seventh. The transfer for data type 4 will consume six words of the value vector. All of the COAST verbs performing data transfer operations employ the vectors described here, except COPY. COPY performs the direct list entry to list entry transfer of information, and does not require a vector for value storage.

The first occurrence in a list entry of data type 1 was selected for the forward link. Similarly, the first occurrence of data type 2 is reserved for the backward link when backward links are required. Technically, links are simply data fields and may be manipulated as data fields using the data transfer subprograms. The convention adopted

regarding data types for links permits the creation of faster, more convenient verbs specific to link manipulations.

It is the nature of the forward-only list to prohibit the access to the predecessor of a list entry. The absence of the backward link makes necessary special handling of operations involving the removal of entries from a forward-only list. In particular, if the last entry is to be removed, it is not possible to set the forward link in its predecessor to zero since the address of the predecessor is unknown. In this situation, the last list entry becomes a private termination cell, PTC, (Cooper and Whitfield, 1962/3) and is fitted with a forward link of 1. Thus, a unity forward link is the identifying characteristic of a PTC. PTC's remain linked into the list but are not otherwise an active part of the list. If the list entry is to be removed from a forward-only list, the entry itself cannot be physically removed as its precursor in the list (which points to it) is not known. However, by copying the entire contents of the list entry following the one to be removed into the list entry to be removed, one has created the effect of removing the chosen entry. The list entry following is then deleted from the list and added to the user's available space list.

Table 1 presents a list of the COAST verbs which are the primary list processing subroutines. Omitted from Table 1 are those COAST programs which are not of direct utility to the programmer of a list processing application. Three verbs included in the table require further attention.

LOCATE, as its name implies, searches a list for the occurrence of a specified pattern of data field values. The data specification vector employed in data transfer operations provides LOCATE with the

TABLE 1

COAST Verbs

Nature	Name	Function
Initialization	COIN	Read in LEND's
Core management	CORN	Release core assigned to a user mask
	NEWCEL	Provide list entry from user available space list
	RETURN	Return list entry to user available space list
Link manipulation	BSTLNK	Set backward link
	FSTLNK	Set forward link
	LNKBWD	Get value of backward link
	LNKFWD	Get value of forward link
Data transfer	COPY	Transfer data from list entry to list entry
	FRMCEL	Extract a copy of data from list entry
	TOCELL	Store data in list entry
Link manipulation and data transfer	LOCATE	Search list for certain data
	POPUP	Perform FRMCEL and remove list entry(s)
	PUSH	Install new list entry and perform TOCELL

identification of the data fields to be inspected. LOCATE permits two modes of operation which are referred to by the descriptors "AND" and "OR". In the "AND" mode, LOCATE is directed to accept as a match the first list entry found to contain the specified data pattern in its entirety. The "OR" mode relaxes this requirement for matching somewhat. In the "OR" mode, the data pattern is partitioned according to data type. The partitions are compared individually to their respective data fields of a list entry. The search is successful when a list entry is encountered possessing the fields of a data type which match the corresponding data value partition. In either mode, the search may be specified to proceed in the forward direction, in the backward direction or specified to encompass the entire list (accessible from the specified starting point).

Like LOCATE, the verbs PUSH and POPUP involve both links and data, but unlike LOCATE result in alterations to the list structure. The PUSH operation is pictorially presented in Figures 7. Although the illustrations are based on a forward-backward list, PUSH operates essentially the same on forward-only lists with two exceptions. Owing to the absence of the backward link, the push-up mode and the insert-before mode cannot be performed per se. If the position of the data on the list is the important consideration and not the address of the particular list entry containing the data, then the insert-after mode and the push-down mode may be taken as the equivalent of the push up and insert before modes respectively. The verb POPUP is essentially the inverse of the verb PUSH. Figure 8 shows the POPUP operation in the forward direction. In the case of forward-backward lists, POPUP is symmetrical and will operate equally well in the backward

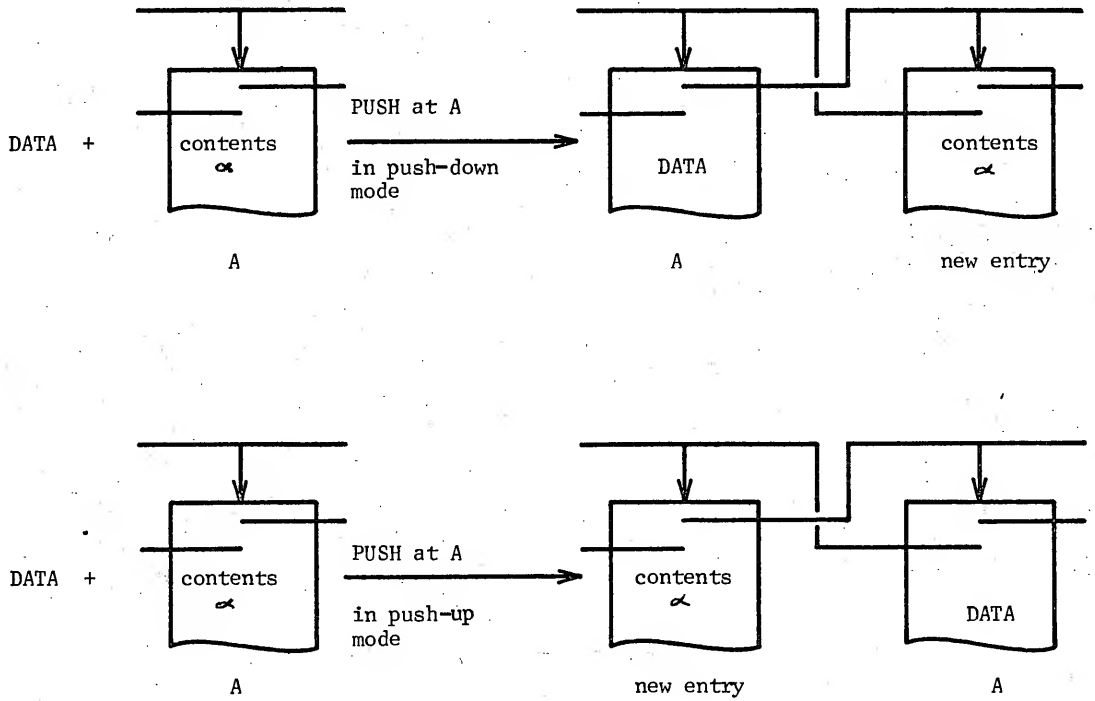

Figure 7. PUSH.

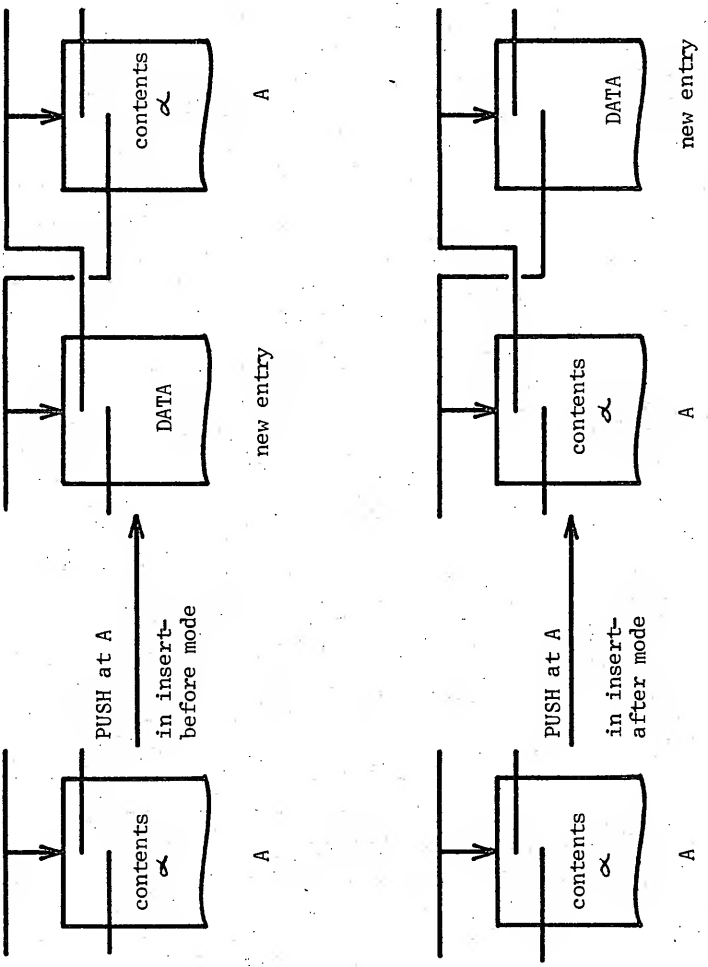

Figure 7.. PUSH (continued).

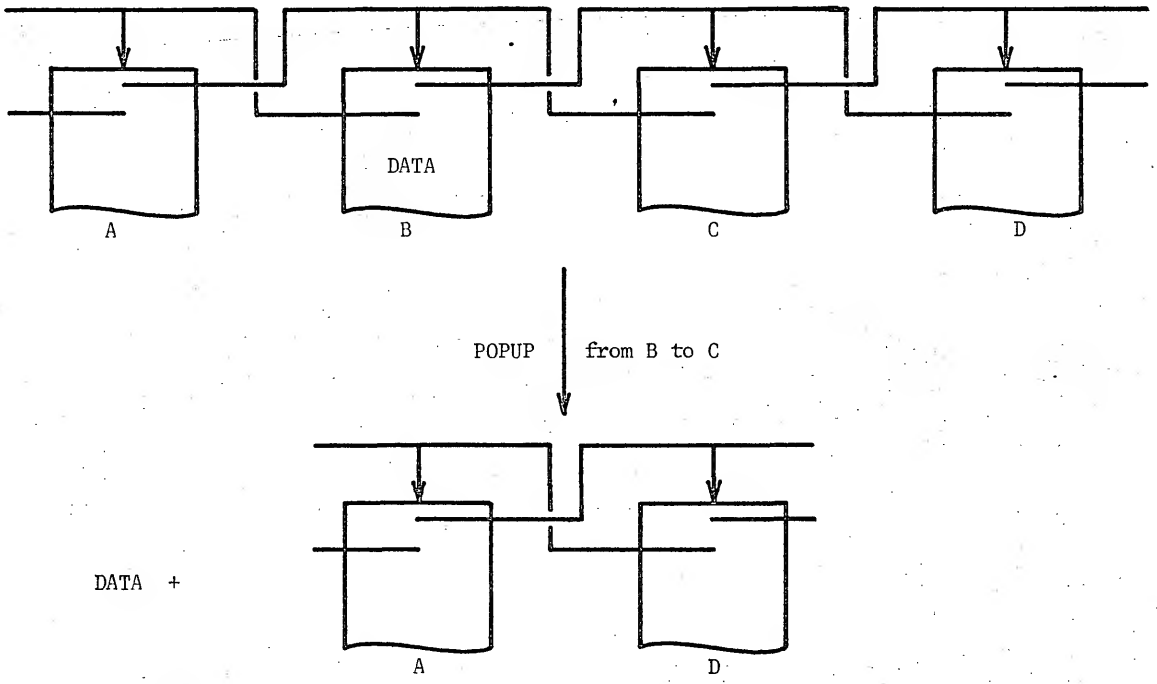

Figure 8. POPUP.

direction. Thus, if POPUP from C to B (according to the backward link) is specified, the entries from B through C in the forward sense will be transferred to the user available space list. The data source will be list entry C. In the example of Figure 8, if the operand C were replaced by 0, POPUP would remove from the list all entries from B to the end of the list. For a forward only list, B would of course become a PTC. Again, the symmetry of POPUP allows this mode of operation for forward-backward linked lists.

As stated at the beginning of this section, machine independence was a goal of the GENDER system design. Unfortunately, the logical operations "AND", "OR" and "exclusive OR" are required for the mask manipulation operations. These logical operations cannot be conveniently provided directly in FORTRAN, necessitating recourse to assembly language. These operations are basic machine instructions and required little effort to prepare for the IBM-360 or 370. This is expected to be true for all other machines as well.

* III.3.b. REMOTE

Avoidance of dependency of the hierarchical file storage system, REMOTE, on large continuous partitions of the ALLOC vector was a paramount consideration in the design of REMOTE. The design is to require file storage in smaller ALLOC partitions, which are somehow connected together to give the illusion of being contiguous. As a further constraint, the linkage mechanism must not seriously interfere with the intention to permit file storage and retrieval form mass memory.

Figure 9 shows the structure adopted for use by REMOTE. The catalogue consists of a forward-backward linked list of ALLOC segments (which could contain, say, 32 or 64 words each). The forward and

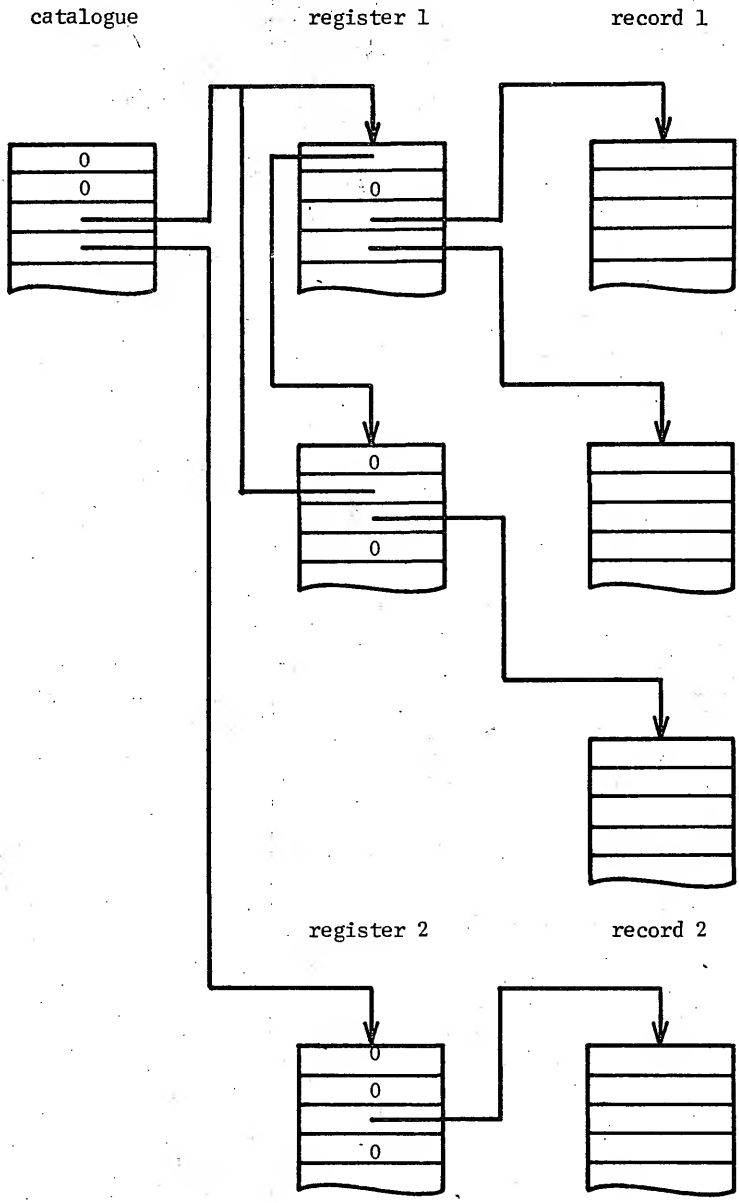

Figure 9. File Structure.

backward links are respectively the first and second words of each catalogue segment. All other catalogue segment words may contain the address of a register. A register is identical in structure to the catalogue. Each register segment word (excluding the first and second) may contain the address of a record segment. The set of all record segments referenced by a single register constitutes a record. It is customary, since a one-to-one correspondence exists between registers and records, to say that the catalogue refers to records.

It is in the record segments that the files are stored. Record segments may be any desired multiple of the ALLOC segment size. The size is specified via the SPUR vector introduced in the preceding section. The continuity of a record (or should we say apparent continuity) is achieved by virtue of its register. The absolute addresses of all record segments participating in file storage are kept in the register segments. Zero contents of a register segment word indicate that the corresponding record segment has not yet been requisitioned from ALLOC. A negative integer stored in a register segment word indicates that the contents of the corresponding record segment are currently resident on mass memory.

Let us briefly review the objectives and constraints noted in the first paragraph of this section before pursuing the features permitting the hierarchical character of file storage. The use of record segments certainly satisfies the stipulation of eliminating any dependence of the file system on large partitions of ALLOC. The registers provide the linkage mechanism for the record segments which is thus segregated from the records and interference with mass memory operations is therefore minimal. Further, the linkage mechanism seems to satisfy the requirement to provide rapid random access and gives the record

apparent continuity to the user. It is important to note, however, that these objectives were met without creating a support structure that would require a large amount of memory. For example, consider an ALLOC segment to be 32 words and a record segment to be 64 words. A catalogue segment and a register segment, a total of 64 words, can hold the addresses for up to 30 record segments or a total of 1920 record words. Thus, the support structure (catalogue and registers) only consumes slightly over 3% of the total memory allocated to a file storage application.

The responsibility for the hierarchical character of the file storage system belongs to the directory concept. Consider, if you will, a file consisting of single word file entries. Into each file entry place the address of a file. This then is the notion of a directory. It is a file which refers to subsequent files. One or more of these subsequent files may also be a directory giving rise to the hierarchical property. However, care must be exercised with respect to the addresses employed in the directory entries. If absolute addresses are used, the file system would become dependent upon the particular ALLOC buddies serving as record segments. This would seriously impair the use of mass memory. An addressing scheme independent of the absolute addresses of the record segments is required; a relative addressing scheme is used.

The relative address of a particular word in a record is simply the integer identifying its sequential position within the record. That is, the ninth word of a record has for its relative address the integer 9. (We shall adopt the convention that the word "address" appearing without the modifier relative will be taken to mean "absolute address.")

The absolute address is readily calculated from the relative address by using the register. If a record segment has 64 words in it then relative address 132 is in the third record segment, word 4. The absolute address of the third record segment is kept as the third entry in the register.

When preparing to add a new file to a record already containing one or more files, it is necessary to know where the last file ends. To this end, the first word of each record (the word in relative address 1) is reserved for the relative address of the first empty word in the record. It is also advantageous, when manipulating a particular file within a record, to know how many file entries constitute the file and how many words each file entry occupies. A special file entry appears as the first entry of each file containing these two parameters. We shall call this file entry the parameter entry of a file.

Figure 10 is an example of the contents of a record. In this example, 16 words are occupied by files, so that the first record word contains 17. The first file commences in word 2 of the record with its parameter entry; this special entry is illustrated as occupying two words. It indicates that the first file contains two single word entries, whose contents here are 6 and 11. In particular, this first file is a directory and its entries contain the relative addresses of two other files. For instance, in word 11 we find the parameter entry of a file composed of two file entries of two words each.

Quite naturally, the question arises as to how access to a particular file entry may be achieved. Access to a particular file entry entails selecting the appropriate record (i.e., the address of its register) from the catalogue, and then making selections from each directory to reach the desired file. Once having accessed the file, by using the

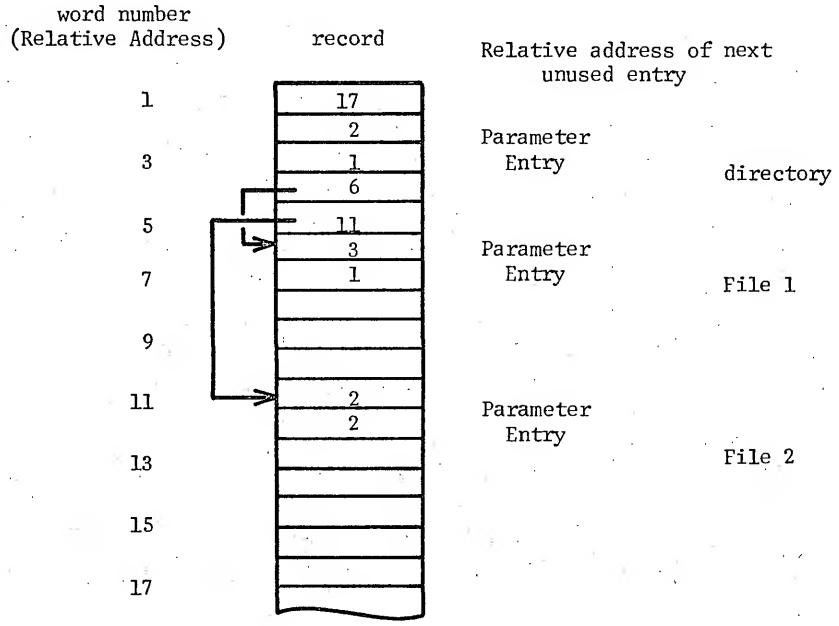

Figure 10. A Record.

number of words per file entry, reaching any particular file entry has been reduced to a problem primarily in counting.

The required selections to be made from the catalogue and directories are specified by a vector, LISTER. The first word of LISTER is its dimension, or in other words one plus the number of choices. The record to be selected from the catalogue is indicated in the second LISTER word. If the dimension of LISTER is N, then the next (N-3) components of LISTER specify selections from the directory hierarchy. (There is no requirement that directories need be employed if a record is to contain only a single file.) Finally, the last LISTER component, the Nth, identifies the particular entry from the file selected. For example, if the record shown in Figure 10 is record number 10, then the LISTER vector (4, 10, 2, 1) will result in the access of the file entry commencing at relative address 13. (Word 1 indicates the length of the LISTER vector which is 4. The 10 says the address of the register for this record is in the tenth entry in the catalogue. The 2 says we are to go to entry 2 of the initial directory which points us to file 2. The 1 indicates we want the first word of the file which is word 13.)

The LISTER vector provides sufficient information for accessing an entry in an existing file, as in the transfer of data from a file entry. When data is to be stored in a file which until this point has not existed, supplemental information is required about the size of any new directories, the size of the new file and the number of words per file entry. This supplemental information is required to insure that sufficient space is reserved (since later entries may be added) for each new directory and file added to a record. This information is provided by a second vector supplied by the user, the length of which

is 2+ the number of directories. If LISTER(1) is N, then the number of components in the auxiliary vector is (N-1). Note that it is necessary only to provide the number of entries in each directory and not also the directory entry size since by convention the directory entry size is permanently fixed at unity. Directory sizes account for the first (N-3) components of the auxiliary vector. The remaining two components are respectively the number of file entries and the file entry size. The supplemental information is of utility only for those directories which have not previously been established. Components corresponding to existing directories are ignored.

Consider a file processing scheme involving the access of many entries on a particular file. The catalogue and directory selections, if repeated for each file entry, would constitute needless repetition since access of the same file would be the result each time the selection sequence is executed. Thus, it is desirable to provide the means of reaching another entry in a file from a file entry that has already been accessed. The move relative to a given file entry to reach another is specified by providing the distance (i.e. number of file entries) of the move and the direction of the move. The LISTER vector is the vector for this information. LISTER(1) is set to 1, indicating the move-relative mode. LISTER(2) contains the displacement as a signed integer, with positive and negative values indicating forward and backward displacements respectively. The LISTER vector (1, 1) would access word 14 of the previous example. Repeating with (1, 1) would then access word 15, and so forth.

If movement relative to a particular file entry is to be accomplished, one must know exactly the location of that file entry. Unfortunately, knowing the location of a file entry involves more than simply knowing

its absolute or relative addresses, though these addresses are certainly essential. It is also necessary to have available the address of the record segment, the sequence number in the record of the record segment, the register segment address and the particular record segment word referencing the record segment. This is by no means a complete list, for many other parameters have been found to be of utility in expressing positions within files or of utility in executing the relative move itself. To provide a storage medium for these parameters, a vector, JCLOAD, has been created. This vector consists of 30 components, each of which is described by the comments to program C4ADRS in Appendix A. For most users it will be sufficient to know that the vector exists and is provided by the user, and that user changes to the vector are prohibited if the relative movement option is to be exercised.

As a result of the sequential appearance of file entries within the files, no verbs for link manipulations are required. In fact, the user is provided with only three verbs for performing manipulations of file storage. These are STORE to store data in a file entry, FETCH to retrieve a copy of data in a file entry, and FIND to locate a file entry containing a particular data value pattern. The specification of the data fields to participate in the data transfer operations is accomplished identically as in the description in the preceding section for COAST. REMOTE does require certain restrictions with respect to data types. Table 2 gives the data types, with their LEND components, which are necessary. Beyond these four data types, the user is free to organize the LEND in any convenient fashion. The V in Table 2 indicates locations where the user may exercise choice.

One eventuality of file storage has until now been neglected. It is possible that a file entry occupying several ALLOC words may not

TABLE 2

LEND Components Required by REMOTE

Data Type	LEND[a]	Use
1	1,0,1,0	Catalogue and register forward link
2	2,0,1,0	Catalogue and register backward link
3	1,V,V,V	First field is number of entries in the file
4	V,V,V,V	Last field is the number of words per file entry

[a] First word of data type, last word for data type - if zero last word is end of list entry, shift, mask.

exactly fit at the end of a record segment. Two policies are possible. One, the file entries are required to be continuous, risking the waste of a few words per record segment. Two, the partitioning of file entries between record segments is to be allowed. If the file entries are not small with respect to record segment size, the latter policy is clearly more desirable with respect to efficient use of memory. Since the sizes of file entries and of record segments are choices permitted to the user, and since the restriction of the user's freedom to organize a file structure was to be minimized, the partitioning of file entries is allowed by REMOTE. As with the JCLOAD vector, the user need not be aware of file entry partitioning to successfully employ REMOTE. The user may prevent partitioning by a judicious choice of the record segment size or of the number of words required by the parameter file entry initiating the file. This latter play is implemented by the word index selection in the LEND component for data type 4. The parameter entry (See Figure 10) is structured with data type 4 concluding the entry. Thus, by adjusting the number of fields subtended by data type 4, it is possible to regulate the size of the parameter entry without otherwise disturbing its contents.

* III.3.c. NETPAC

The basic network structure has been discussed in Section II.2.a. As noted there, certain data fields are essential to the network structure. That is, the links alone are insufficient to adequately define a network structure.

Table 3 is a list of the data fields augmenting the link fields. Of these additional fields, only data type 3 and the first two fields of data type 5 are of an essential nature. In order to differentiate nodes from diverger entries, some unambiguous feature must be provided

TABLE 3

Auxiliary Data Fields Required by NETPAC

Data Type	LEND[a]	Data Field	Use
3	1,V,V,1+	1	Identification flag (Q diverger entry, > 0 node)
4	V,V,V,V	1	Trace key
		2	Sub-nodal network trace key
5	V,V,V,V	1	Number of forward paths leaving node
		2	Number of backward paths leaving node
		3	Counter employed during trace

[a] First word in list entry for data type, last word in list entry for data type - if zero last word is end of list entry, shift, mask.

to each. This feature is a flag, which is zero for a diverger entry. The first field of data type 3 serves as the flag. The + symbol in the LEND for data type 3 indicates that the mask may encompass more than a single bit. The first two fields of data type 5 contain the number of paths leaving a node in each direction. These fields are absent from diverger entries.

The remaining data fields listed in Table 3 are not essential to the network structure, but are required if the network is to be traced using the verb NEXT.

The trace strategy employed is somewhat involved. Each trace is assigned a unique trace key. We shall call this key the master trace key. The purpose of the key is to flag the nodes which have been encountered by the trace so they may be distinquished from those which have not yet been reached. This is necessary since several paths may lead to the same node. Upon the first encounter of a node, which is indicated by its trace key being unequal to the master trace key, the trace key is set equal to the master key and the counter in field 3 of data type 5 is set to unity. Successive encounters with this node will be indicated by the trace key matching the master key. The counter is incremented by one at each encounter of the node. When the counter matches the first field of data type 5 for a backward trace or the second field of data type 5 for a forward trace, a node has truly been "reached." Further, it is only upon meeting this criterion that the links leading from a node in the direction of trace may be examined.

The trace strategy was dictated primarily by the needs of the interpreter in evaluating the components of a solution procedure. No component may be evaluated until all preceding components have been evaluated. That is, no node is considered as reached until all paths

leading into the node have been followed. Figures 11 through 11e illustrate the steps in tracing a simple network. The data fields appear in the nodes shown in the order of their occurrence in Table 3. For simplicity, each is assumed to occupy an entire word. An * appears beside each node as it is reached.

Let us consider Figure 11c. The count does not match the number of backward paths. This causes the trace procedure to backtrack and consider the path parallel to B. For more complex networks, it is necessary to distinquish the parallel paths yet to be considered from those which have already been inspected. This distinction between paths is facilitated by an auxiliary push-down list. Each entry on this list points to a diverger entry. These diverger entries are the first on their respective divergers which have not yet participated in the trace. The top member of this list always points to the last parallel path encountered and the first to trace when the current path terminates.

It may be noted that the sub-nodal network trace key played no part in the trace illustration. This trace key is the master key for networks existing as auxiliaries to network nodes. The import of the network within a network will be pursued in Chapter IV. When the sub-nodal networks are present in a network structure, the trace procedure accounts for their presence by entering each auxiliary network just prior to advancing to the next node on the main list. For example, if node A of the trace example presented in Figures 11 through 11e possesses a sub-nodal network, node B is not considered to directly follow A in the network trace. Rather, the first node of the sub-nodal network follows A in the trace. A unity identification flag identifies nodes possessing sub-nodal networks. All nodes possessing flags greater than unity are not permitted sub-nodal networks.

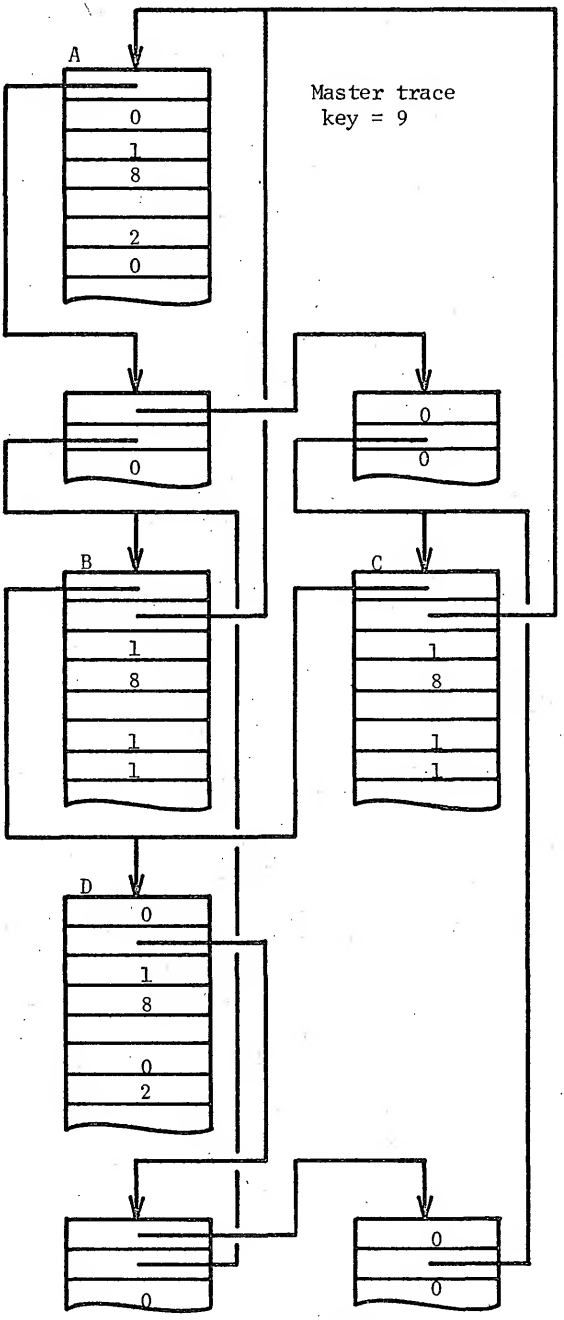

Figure 11. Trace (Start).

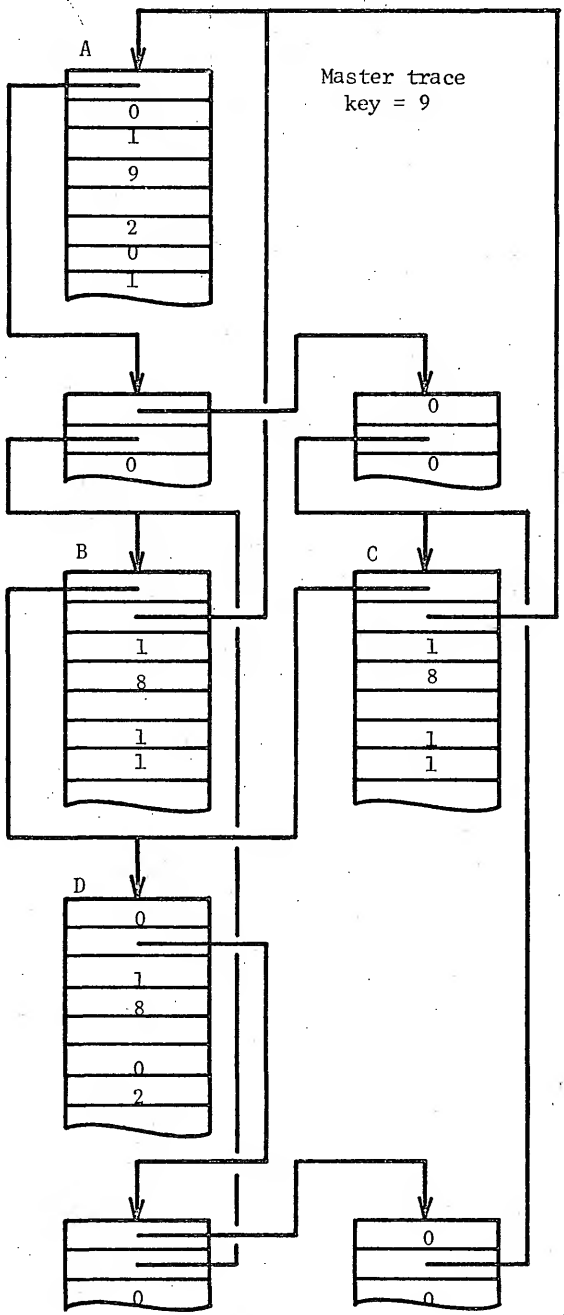

Figure 11a. Trace (Step 1).

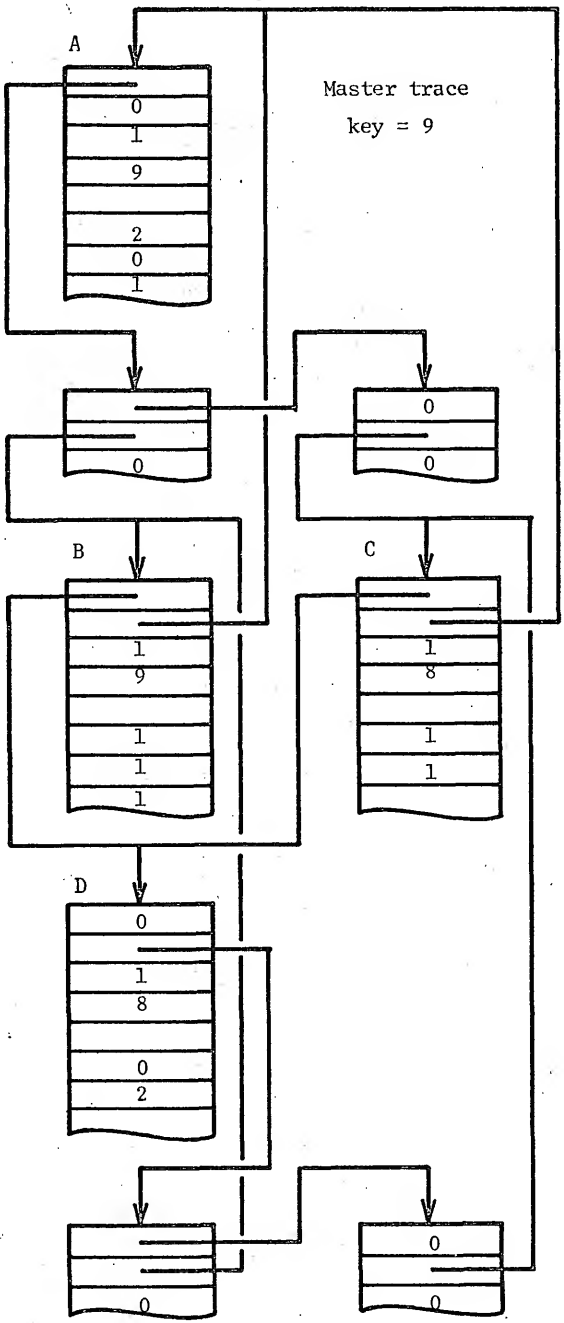

Figure 11b. Trace (Step 2).

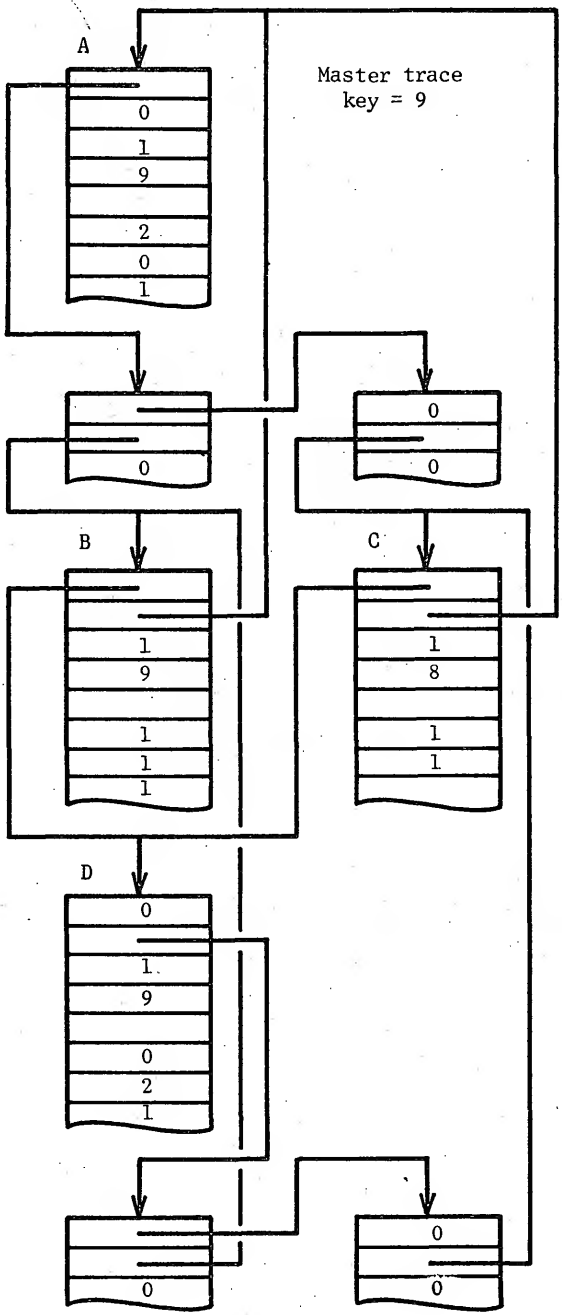

Figure 11c Trace (Step 3 - no node selected).

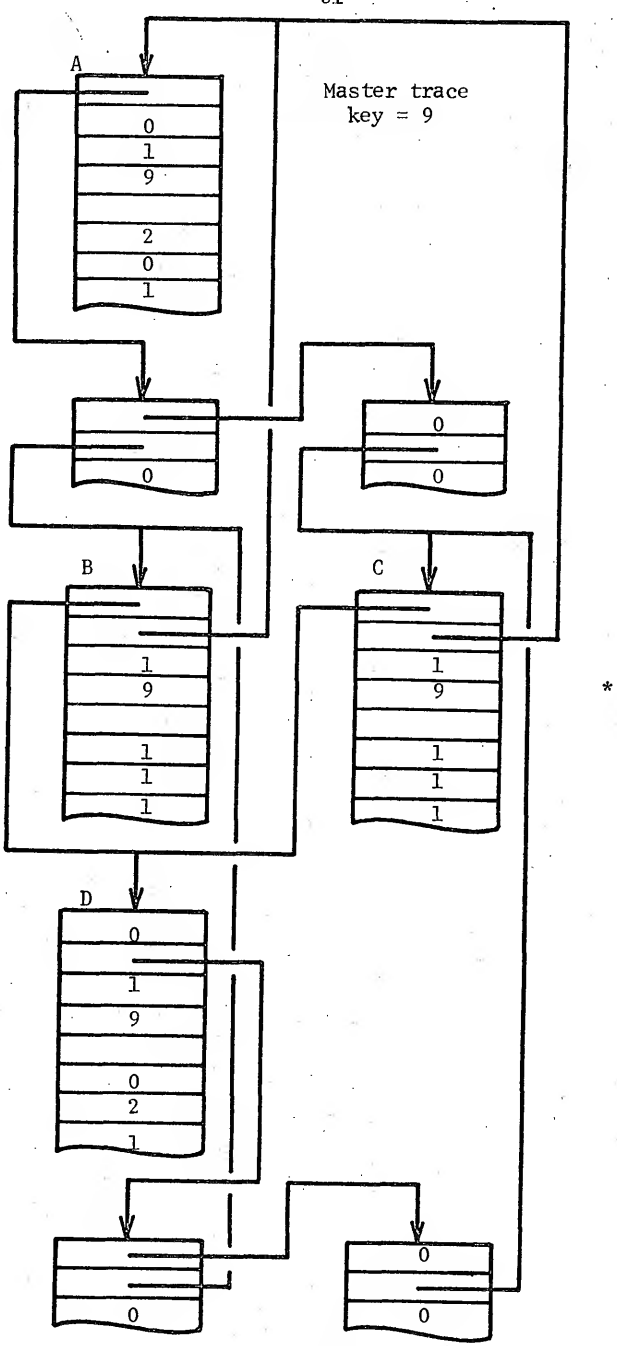

Figure 11d. Trace (Step 4).

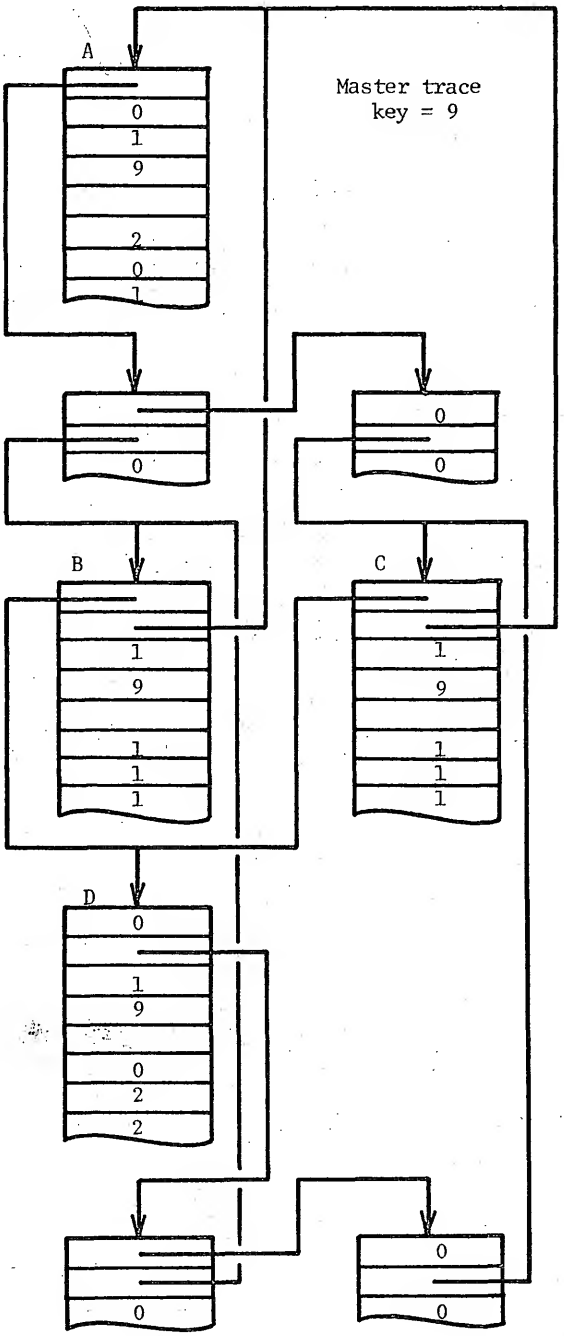

Figure 11e. Trace (Completed).

As a result of networks within networks, the auxiliary list employed by the trace procedure must be complicated somewhat. The paths yet to be inspected at each level in the network must be segregated from those of all other levels. This segregation is accomplished by a list composed of one entry per network level. Each entry points to a list indicating the remaining parallel paths on a particular level. The manipulation of this auxiliary list structure is not required of NETPAC users.

The subroutine NEXT is the NETPAC verb providing the trace capability. NEXT requires only that the user indicate the address of a current node and the value of the master trace key. NEXT performs all of the network and auxiliary list manipulations necessary to provide the user with the address of the next node in the trace sequence.

The COAST verbs require SPUR vectors to indicate the size of list entries and the list type. In order for the NETPAC verbs to employ the COAST verbs, it is necessary to provide a battery of SPUR vectors, since many different sizes of list entries appear in the networks used in GENDER. One SPUR is required for each list entry size. To satisfy this requirement, a sequence of SPUR's graduated in sizes from a single word list entry up to the largest possible list entry size are stored in a continuous ALLOC partition. It is expected that this SPUR block will typically be less than the ALLOC segment size.

Having the SPUR block only solves half the problem. To be able to select the proper SPUR, one must know the sizes of the various list entries constituting the network. A vector, which we shall call the information block, contains this data. The values for each information block component are provided as data by the user. Only three pair of

the 28 information block components are required by NETPAC. These pair are components 8-9, 20-21 and 22-23, and are associated with networks employing list types 2, 6 and 5 respectively. Networks have been restricted to list types 2, 6 and 5 primarily as a programming convenience since the use of NETPAC external to GENDER is doubtful. Information block components 8, 20 and 22 indicate diverger entry sizes, while components 9, 21 and 23 are node sizes. The identification flag distinguishes nodes from diverger entries. Although more than one type and size of node may appear in a network, NETPAC requires access only to data types 1 through 5. Since every node contains these data fields, the SPUR for any node will suffice NETPAC. For the user's convenience, information block components are reserved for the sizes of the other nodes permitted to list types 2 and 5. Discussion of the remaining information block components is deferred to the particular chapters to which they are relevant.

The information block introduced in the preceding paragraph resides in ALLOC. The address of this block is recorded in the seventeenth ALLOC word. Consequently, access to information block data is relatively simple and the data is available to all GENDER subroutines sharing ALLOC. It would seem advisable to make similar provisions for the SPUR block. In fact, a vector, NIMBL, has been invented to contain the addresses of the network and the SPUR block. NIMBL also contains the value of the master trace key. The address of the NIMBL, thus, furnishes sufficient data to permit the access and manipulation of a network.

Table 4 is a list of the NETPAC verbs. BREAK and COUPLE appropriately adjust the path counters, as well as performing the necessary link manipulations. One node to be separated from a second via BREAK

TABLE 4

NETPAC Verbs

Name	Function
BREAK	Breaks the links between two nodes
COUPLE	Connects two nodes
LNKBNT	Gets a copy of the backward link
LNKFNT	Gets a copy of the forward link
NEXT	Performs trace
SEARCH	Employs NEXT to search a network for a specified pattern of data values

must be specified by its absolute address. Although the second node may also be specified by its absolute address, it may be designated by reference to a path leading from the first node. Referring to Figure 11, node A may be separated from node C by providing BREAK with the addresses of both A and C, or with the address of A and the integer 2 indicating the second diverger entry. The path specification mode is available to both forward and backward directions. That is, A may either precede or follow C. BREAK is designed to provide the calling program with the path position if the addresses of both nodes are specified, or with the address of the second node otherwise. This feature has proven to be of use in the algebraic package, GALAP.

COUPLE requires the specification of the addresses of both nodes to be connected and the specification of the path positions. While the order of diverger entries is of little consequence to solution procedure network, the arrangement of parallel paths is of considerable importance in GALAP. Consequently, COUPLE has been designed to permit selective path placement.

The verb SEARCH provides the capability to search a network for a specified pattern of data values. The data values and the corresponding data fields are provided to SEARCH identically as for the COAST verb LOCATE. SEARCH employs NEXT in performing the network trace. Networks possessing a single starting node and a single terminal node may be readily searched in either direction. However, a network having multiple terminal nodes cannot be completely searched in the backward direction. The selection of a terminal node to serve as the initial trace node excludes from the trace, at the very least, the other terminal nodes.

* III.3.d. SIMPAC

Section III.2.b provided a glimpse at the sparse incidence matrix structure. This section will further detail this most important data structure.

The ordinates of an SIM (sparse incidence matrix) are applications of REMOTE. However, this need not have been the case. The ordinates might simply have been continuous partitions of ALLOC. An ordinate of 1,000 entries would require a maximum of approximately 9,000 words. This would constitute a demand for a buddy unlikely to be available, except when specially provided. The implementation of the ordinates as REMOTE records eliminates the requirement for special supervision of the ALLOC allocation mechanism. Further, the use of a record as an ordinate entails no further sophistication of the REMOTE facilities.

Figure 12 shows a sparse incidence matrix with the ordinates detailed as REMOTE records. This matrix is the same as the one in Figure 5, but shown here in more detail. List type 3 has been selected for the ordinates. A number of data fields are required for each ordinate entry. For convenience, these fields are tabulated in Table 5. A few of these fields require further clarification. The status flag is a relatively small field reserved for use by the solution procedure generation algorithms discussed in Chapter V. When equal to one, the dimension flag indicates the substitution of a pointer to an auxiliary data block for the name in data type 9. The auxiliary data block contains the name, dimensionality and index values as whole word entries each. The name is more properly called a code name. The code name uniquely identifies the occurrence of a function, ER, constraint or variable in the data base SECEDE. SECEDE and the rationale of code names is discussed in Chapter IV. Data type 8 has no significance for

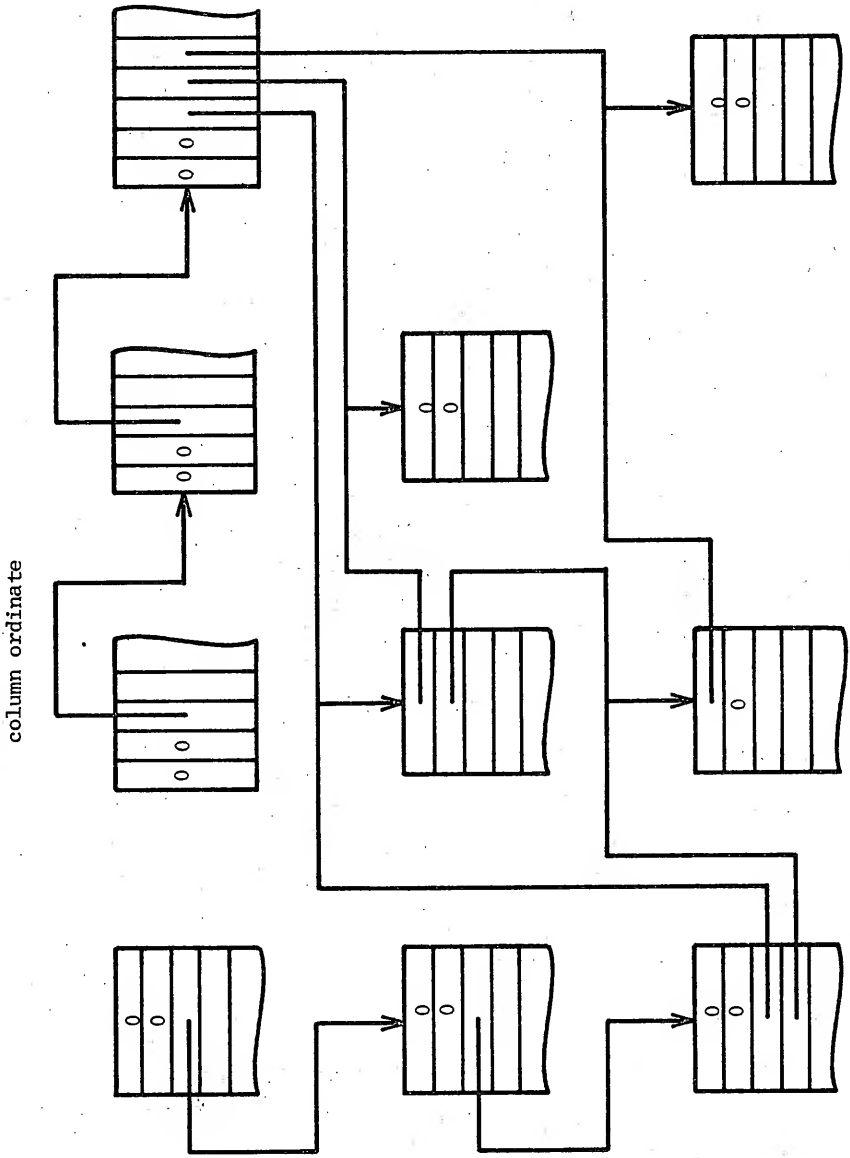

Figure 12. Sparse Matrix.

TABLE 5

Data Fields Required for Ordinate Entries

Data Type	LEND[a]	Use
5	V,V,V,3+	Row/column status flag
6	V,V,V,V	Row/column pointer
7	V,V,V,1+	Dimension flag
8	V,V,V,3+	Identification flag (OER, 1 function, 2 constraint, 3 group)
9	V,V,V,V	Function name
10	V,V,V,V	Number of elements per row/column
11	V,V,V,V	Number of unassigned output variables
12	V,V,V,1+	Multiple output flag
13	V,V,V,V	Output assignment

[a] First word in list entry for data type, last word in list entry for data type - if zero last word is end of list entry, shift, mask.

the column ordinate entries, since all columns represent only variables. The group mentioned in Table 5 for data type 8 will be pursued subsequently in Chapter IV and is more properly called a protected group. For our purposes in this section, it is sufficient to regard a protected group and the ER (for external routine) simply as functions. That is, they are functional relationships involving two or more variables. If the multiple output flag is set to one, the output assignment in data type 13 is replaced by a pointer to a list of output assignments. Two words are required for each list entry. The first contains the link and the second contains the output assignment. An output assignment is simply a row or column number depending on whether the ordinate pertains to columns or to rows respectively.

Each element of a sparse incidence matrix must also carry some information in addition to the links. In particular, it is necessary to know to which column and row an element belongs. This information is provided by fields reserved for the row and the column indices. Other fields are required, but are of a more specialized nature. Table 6 summarizes the data fields appearing in each element. The field for the sensitivity may be omitted at the user's option. The status flag and output selection cost are required for the output assignment and other algorithms. List type 4 is reserved for the SIM elements.

The 28 component information block introduced in the preceding section contains components pertinent to the SIM. Table 7 indicates the information block partition dealing with the SIM.

As in the case of networks, a SPUR block is essential in constructing and performing manipulations on an SIM. The SPUR's required correspond to LOAD vector size (30 words), the element size, ordinate

TABLE 6

Data Fields Required for SIM Elements

Data Type	LEND[a]	Use
1	V,V,V,V	Horizontal link
2	V,V,V,V	Vertical link
3	V,V,V,V	Row index
4	V,V,V,V	Column index
5	V,V,V,3+	Element status flag
6	V,V,V,15+	Output selection cost
7	V,V,1,0	Value of sensitivity

[a] First word in list entry for data type, last word in list entry for data type, shift, mask - If zero entire word is used.

TABLE 7

Information Block Components Pertinent to the SIM

Component	Use
16	Ordinate segment size
17	Number of works per horizontal and vertical link pair
18	Length of an element list entry
19	Length of an ordinate file entry

segment size and list entry sizes from a single word up to the largest required for an auxiliary data block. This largest list entry size is determined as (2 + maximum dimensionality). The maximum dimensionality is contained in the fifteenth information block component. The individual SPUR's appear in the SPUR block in the order of their enumeration above. The point of reference in the SPUR block was selected as the sixteenth word of the block. That is, the address used to access the SPUR block is the first word of the SPUR for a single word list entry. Decrementing this address by five accesses the SPUR for an ordinate segment.

Not only has the SPUR block been adapted from NETPAC, the SIMBL is not unlike the NIMBL. The first two words point respectively to the column catalogue and the row catalogue. The third SIMBL word points to the SPUR block. The description of the SIM structure in detail is completed by stating that the address of the LOAD vectors are contained in the catalogues. That is, the address of the LOAD vector for the row ordinate appears in the second catalogue word.

It is permissible for an SIM to be equipped with more than a single row and a single column ordinate. Working or temporary ordinates are frequently utilized by the solution procedure generation algorithms. Because of the inclusion of the LOAD vector addresses in the catalogues, all ordinates are odd numbered REMOTE records.

The majority of the SIM verbs are specialized in nature, dealing with a single, generally repetitive SIM operation. The SIM verbs are listed in Table 8. The SIM generator, SIMGEN, performs a great deal more than a simple SIM manipulation. SIMGEN uses the list of functions, ER's, etc. belonging to a partition of the solution procedure (called

TABLE 8

SIMPAC Verbs

Name	Function
SIMCIN	Adjust the row and column incidence counts to reflect the removal or reinstatement of a row or column
SIMCPY	Transfer selected data fields from each entry of one ordinate to other selected fields of a second parallel ordinate
SIMDUP	Duplicate the contents of each entry of one ordinate in a second parallel ordinate
SIMEIN	Set all row, column and element status flags to zero
SIMFLG	Set the rodinate and the element status flags of a specified row or column to a selected value
SIMGEN	Generate an SIM
SIMINT	Interchange two parallel ordinates
SIMOPT	Record or erase a row-column output assignment
SIMRTV	Set the status flags of all elements of tear variables to 2 except where they are the output of a row

a group) for which further analysis is required as the basis for SIM generation. Each functional relationship is assigned a row in the order of occurrence in the group. SECEDE furnishes SIMGEN with the incidence and output assignment cost data. SECEDE is described in Chapter IV. In addition to structuring the SIM, SIMGEN also initializes the row, column and element status flags at zero. The utility of the SIMPAC verbs will become more obvious in Chapter IV.

CHAPTER IV

DATA BASE

The information constituting a system of equations and the associated solution procedure must be readily available. One approach to the data base would have been the inclusion of all information relative to an equation set within the solution procedure. As noted in the preceeding chapter, the network structure was adopted for the solution procedure for reasons of flexibility and ease of manipulation. The concept of the unified data base would, during certain operations, encumber the solution procedure with much more data than required. An alternative approach involves providing the general information apart from the specific information pertinent to the solution procedure.

IV.1. Service Module, SECEDE

The general information phase of the data base must at times augment the information within the solution procedure. For instance, during generation of a sparse incidence matrix the variables appearing in an equation must be known. Since this information is not contained within the solution procedure (as SIM generation is one of the few instances when it is required), it must be extracted from the general information. Further, since the equations on the solution procedure are subject to re-ordering, it seems appropriate to require rapid random access to the general information.

The file system, REMOTE, described in Chapter III not only fulfills the requirement for rapid random access, but also permits the release

of ALLOC space by the transfer of data not currently required to mass memory. The application of REMOTE to the storage of information relative to an equation set is termed the service module (SECEDE).

The organization of data within SECEDE required careful consideration. First, the construction of a library of data relative to various processing units is to be permitted. Second, the organization must permit the deletion or inclusion of units from SECEDE. Finally, SECEDE should permit the reorganization of the flowsheet (i.e., connections between units). All of these considerations are essential if the employment of GENDER during the synthesis phase of design is to be feasible. The first two requirements are satisfied by assigning each unit to a separate record. The third requirement is satisfied by restricting all of the information on stream connections to a single record called the zeroth unit. Thus, reorganization of a flowsheet would involve changes only to the zeroth unit unless additional units or unit substitutions are involved as well.

It is, of course, not known a priori what position a unit, which may be from a library, might occupy in the service module. This necessitates a code name assignment scheme for the variables and equations which will produce unique codes. The variables and equations within each unit are sequentially assigned code numbers commencing with unity. The unique code is developed by adding to the code number, as assigned within the unit, the unit number multiplied by an arbitrarily selected scale factor. For instance, variable 5 of unit 9 would be known by the code 905 if the scale happens to be 100. The code 905 differentiates this particular variable from the fifth variable of all other units.

IV.2. Solution Procedure, GENDER List

For the reasons noted in Chapter III, the solution procedure is a network application. It is expected that many solution procedures will involve cyclic calculations. The GENDER list must reflect the existence of such cycles. Special nodes are provided, called groups, which represent the occurrence of a special set of functional relationships. These may be cyclic, or may be grouped because of other common characteristics.

The group node appears in the network as a representation of the group members. Membership within a group is indicated by the nodes of a network called the group body. Each group contains the address of its group body. Groups as well as equations, ER's and constraints may belong to a group body. The group bodies give the GENDER list a dimension of depth - i.e., networks within networks.

The GENDER list must be a complete specification of the solution procedure it represents. That is, the GENDER list must reflect decision variable selection, output set assignment, precedence ordering, tear variable selection, and selection of the method for resolving cyclic calculations. Data fields and/or auxiliary lists are provided to store all of the foregoing information with the exception of the precedence order. The precedence order is reflected by the arrangement of the nodes on the GENDER list.

IV.3. Input/Output Facilities

The initial point for a GENDER list is generally expected to be the crude GENDER list generated by a program called CUDGEL. Each equation, ER and constraint is represented by an entry on the crude GENDER list, but missing are the details and auxiliary lists necessary for a complete solution procedure specification. Though this may be

the usual starting point, it will not always be most convenient. For instance, a solution procedure might be completely developed, but interpretation prohibited by the lack of a value for an algorithm selected decision variable. For operation in a batch machine environment, this eventuality would cause the effort of analysis to be wasted unless recovery facilities are provided. While one obvious solution is to require estimates for all variables prior to initiating analysis to prevent the development of this particular situation, an inordinate amount of effort would be required from the user. This is but one instance of when an analysis might be wasted. Others exist, and enumerating precautions against all is unlikely. The recovery mechanism has been incorporated into the input/output facilities.

While the recovery provisions are not automatic, they do posses the virtue of simplicity. When the development or interpretation of a solution procedure must be abandoned, the service module and the solution procedure may be saved for use at a later time in one of two ways. Both SECEDE and the GENDER list reside in the ALLOC vector. Hence, a copy of the contents of ALLOC will save SECEDE and the GENDER list. The second approach involves the output of only SECEDE and the GENDER list. This latter strategy will greatly reduce the output volume, but this is not the primary advantage. The format of output may be adjusted to produce a much more manageable medium. Particular significance will be attached to each output record and the output records formatted according to content.

The medium selected for storage of service modules and solution procedures for later use is punched cards. Thus, when the solution of a problem must be aborted, the programs SECIAO and GENIAO may be directed to produce respectively the contents of SECEDE and of the GENDER list

in card form. These cards may later be read by SECIAO and GENIAO to resume solving the problem. Observe that a solution procedure, complete in every detail, may be stored in card form as readily as a partially complete solution procedure. In fact, a library of solution procedures may be amassed to eliminate repetitive analysis of problems which are frequently solved.

The format of each card generated by SECIAO and by GENIAO is presented in Appendix A in terms of data formats. This is permissible since the input and output formats are identical. For the convenience of the user, a printed output will always be provided with card output. In fact, a printed only output mode may be selected for both SECIAO and GENIAO should the punched card output not be desired.

IV.4. Implementation Details

The remainder of this chapter delves into the implementation details for SECIAO and for GENIAO. These details will be of utility only to those users wishing to prepare programs which directly access either SECEDE or a GENDER list. Most users should find the program comments for SECIAO adequate to encode properly the statement of the problem as SECIAO data. The printed output of both SECIAO and GENIAO is preceded by a table indicating the composition of each type of output line. A third subprogram, VARIAO, completes the input/output facility package. VARIAO supplies a user with a list of variables for which values are required. VARIAO is provided primarily for use by the interpreter, GLINT, to indicate variables encountered for which values have not been provided.

* IV.4.a SECEDE

Basically, the service module, SECEDE, is an application of REMOTE.

The REMOTE system is founded on the premise that all entries of a file are of the same size. Unfortunately, exceptions to this fundamental premise may arise. For example, SECEDE is required to store functions in terms of the variables, operators and constants of which they are composed. All three entry types require data fields for an identification flag and a code name. However, the variable also requires fields for the dimensionality and, if applicable, the indices. While it would certainly be possible to insist that all file entries representing a function be equal in size, the opportunity for wasting considerable ALLOC space would just as certainly be present. This eventuality would be realized for all functions containing only a few dimensioned variables with respect to the total number of entries.

The apparent inconsistency between storing functions as REMOTE files and efficient memory utilization can be reconciled, albeit artificially. To each file entry add another data field containing the length of the entry. Moving from one file entry to the next would involve adding the file entry length to the current position in the file. Except for the variations in file entry length, this procedure is decidedly reminiscent of the relative move capability already present in REMOTE. In fact, if REMOTE were led to believe that the file entry length is unity, and if the actual entry length is used as the number of file entries to advance, then movement can be accomplished via the REMOTE relative move mechanism.

Before proceeding, we should clearly differentiate the two types of entries discussed in the preceding paragraph. When establishing a file, the JCFINE vector specifies the file entry length. This is a fixed number and is the length employed in specifying the length of a file. The entry of variable size is not a true file entry. It is rather a sequence of one or more words of a file which we choose to

treat as a single entity and to which we attach special significance. Let us assign the acronym ASE to the adjustable size entries, while reserving the term file entry to mean true file entry in the REMOTE sense.

A file composed of ASE's strongly resembles a forward-only list. That is, the length stored in each ASE serves as the link to the next ASE. Consequently, we shall refer to this scheme as relative linking, and to the stored length as the relative link. The last ASE must contain zero for the relative link just as a terminal list entry must contain zero for the forward link. Since the identification flag stipulates the fields within the last ASE, and since the LEND stipulates field placement within the ASE, the zero length specification presents no hindrance to data transfer.

A hindrance to data transfer does, however, exist. The file entry length must be specified as unity to permit relative linking. Normally, REMOTE will not permit data transfer operations to exceed the bounds of a file entry. The ASE's are expected to be multi-word entries. To circumvent this problem, REMOTE was adjusted to permit the violation of file entry bounds if LOAD(29) is set equal to -1. The user must ensure that LOAD(29) is properly set before attempting access to any file. LOAD(29) should be 0 for all files not employing relative linking.

As we noted earlier, an ASE file resembles a forward-only list. Consequently, it is not possible to randomly access the ASE's. This need not be the case. Suppose the relative links are removed from the ASE's and placed at the beginning of the file. The value of the relative links would, of course, be different since the relative link is the number of words to advance from the current position to the ASE.

What we have developed is the notion of a directory, employing relative links instead of relative addresses, which is internal to a file. The term internal directory shall be employed to describe this placement of the relative links.

Files containing ASE's in the list format will be called sequential access files, SAF's. Files containing unlinked ASE's and an internal directory will be called random access files, RAF's. The RAF does not provide a storage capability which can not be duplicated by an ordinary REMOTE file structure without recourse to ASE's. This could be accomplished by establishing each ASE as a separate file and providing a directory permitting random access to the separate files. However, this structure demands that the relative access be accomplished at the expense of repeating all catalogue and directory hierarchy selections for each access. The RAF structure permits the access computations to commence with the internal directory, since this directory and the ASE's are actually all members of a single file making the REMOTE relative address capability applicable. That is to say, the RAF access time would be generally shorter.

SECEDE consists of at least five records, each record containing from one to five files. These files include all three types; ordinary REMOTE files of equal sized entries, SAF's and RAF's. Each record will be given individual attention. In the discussion to follow, the notation (t i f j) will be used. The letters i and j represent integers indicating data type and data field respectively. The letters t and f should be translated as type (i.e., data type) and field. If we were to speak of a data field (t5f9), we will be considering the ninth field of data type 5. On occasion it may be convenient to expand our field designation shorthand to include a range of fields for a single data

type. For example, (t5f7-9) is translated as fields 7 through 9 of data type 5.

Armed with the above bit of cryptography we are prepared to inspect the purpose and contents of record one. Tables 9 through 11 may be of assistance in this discussion. The first record consists of a directory and two files. The files contain respectively decision and tear variables and are of the SAF type. Each of the ASE's contain the data fields (t1f1-3), plus an additional two words for each degree of dimensionality. The two word allocation for each index is required to permit the specification of minimum and maximum of an index range. A single ASE may represent, for a dimensioned variable, a single component or many components. Each word reserved for an index contains the fields (t9f1, t10f1, t11f1, t12f1, t13f1, t14f1, t15f1), which correspond to the seven constituents permitted in an index calculation as discussed in Appendix B. These constituents, in the order of the fields, are mapping flag, mapping index, operator flag (0 for multiplication by scale and 1 for division by scale), sign flag for scale (0 for positive and 1 for negative), scale, sign flag for offset (0 for positive and 1 for negative) and the offset. The mapping flag may be either 0 for no map, or may be 3 indicating mapping to another index of the same variable. If the mapping flag is 0, data types 10-14 are ignored and the offset is taken to be the index value. As a matter of convention, all data structures requiring the seven part indices will employ data types 9-15 exactly as described here.

The second and third records in SECEDE each contain only a single file. These files are ordinary REMOTE files composed of file entries all equal in size. Each file entry contains alphameric characters forming the name of an operator in the case of record two or the name

TABLE 9

Data Fields for SECEDE

Data Type	LEND	Use Number	Use
1	1,0,1,0	1	Foward link
		2	Relative link
		3	Variable code name
		4	Dimensionality
		5	Dimension range
		6	Operator name (alphameric)
		7	Method name (alphameric)
		8	Constant value
		9	Internal directory pointer
		10	Absolute pointer to a DIVARB or to a scalar variable value
		11	Variable value
		12	Number of outputs
		13	Variable name (alphameric)
		14	ER name (apphameric)
		15	Common variable code name
2	2,0,1,0	1	Backward link
3	1,V,V,V	1	Number of file entries
4	V,V,V,V	1	Number of words per file entry
5	2,V,V,3+	1	Identification flag (0 operator, 1 variable, 2 constant)

TABLE 9

(Continued)

Data Type	LEND	Use Number	Use
6	V,V,V,V	1	Unused
7	V,V,V,15+	1	Output selection cost flag
8	V,V,V,V	1	Variable code name
9	1,0,V,3+	1	Mapping flag (0 no may, 1 unused, 2 map to equation index, 3 map to another index of itself)
10	1,0,V,V		Mapping index
11	1,0,V,1+		Operator flag (0 multiply, 1 divide)
12	1,0,V,1+	1	Operator flag (0 add, 1 subtract)
13	1,0,V,V,	1	Scale
14	1,0,V,1+	1	Operator flag (0 add, 1 subtract)
15	1,0,V,V	1	Offset or index
16	V,V,V,V	1	Operator code name
		2	Constant code name
17	V,V,-,31	1	Key (upper 5 bits)
18	1,1,1,V	1	Relative link to first value of a dimensioned variable
19	1,V,V,V	1	Dimensionality

TABLE 10

File Entries for SECEDE

File Entry	Composition	Use Number	Use
1	t3f1,t4f1	1	Pre-file parameters
2	t1flu2*,t1f2u3,	1	Decision variable
	t1f3u4,(t9-15)x2d	2	Tear variable
3	t1flu6	1	Operator name
4	t1flu7	1	Method name
5	t1flu8	1	Constant value
6	t1flu2,t1f2ul2,	1	Dimension
	t1f3u4,(t9-15)x2d		
7	t1flu2,t5f1,t16flul	1	Operator
8	t1flu2,t5f1,t6f1,	1	Variable
	t7f1,t8f1,t19f1,		
	(t9-15)xd		
9	t1flu2,t5f1,t16flu2	1	Constant
10	t1flu9		Internal pointer
11	t1flu4,t1f2-(d+1)u5	1	Declaration (equation, ER, constraint)
12	t1flul0,t1f2u4,	1	Declaration (common variable)
	t1f3-(d+2)u5		
13	t1flull^{+}	1	Variable value
14	t17f1,t18f1	1	Relative pointer to values of dimensioned variable

TABLE 10
(Continued)

File Entry	Composition	Use Number	Use
15	tlflul3	1	Variable name
16	tlflul4	1	ER name
17	tlflul0,tlf2ul5, tlf3u4,tlf4-(d+3)u5	1	Declaration (variable)

* The u indicates the use number from Table 9. Where it is unambiguous to do so, the u will be omitted.

\+ For double precision, the composition will be (tlfl-2ull).

TABLE 11

SECEDE Files

Catalogue	Directory	Directory	File Composition	Use
1	1	-*	R(e2u1)+	Decision variable
1	2	-	R(e2u2)	Tear variable
2	-	-	R(e3)	Operator name
3	-	-	R(e4)	Method name
4	1	-	R(el0),R(el3)	Common variable declaration
4	2	-	-**	Number of common variables
4	3	-	R(el3 or el4)	Common variable value
4	4	-	R(el5)	Common variable name
5	1	1	R(el0),R(el7)	Variable declaration
5	1	2	-	Number of variables
5	1	3	R(el0),R(ell)	ER declaration
5	1	4	-	Number of ER's
5	1	5	R(el0),R(ell)	Equation declaration
5	1	6	-	Number of equations
5	1	7	R(el0),R(ell)	Constraint declaration
5	1	8	-	Number of constraints
5	2	-	R(e5)	Constant value
5	3	-	R(el3 or el4)	Variable value
5	4	-	R(el6)	ER name
5	5	-	R(el5)	Variable name

TABLE 11

(Continued)

Catalogue	Directory	Directory	File Composition	Use
5	6	.++	e6,R(e8)	ER
5	7	.	e6,R(e7 or e8 or e9)	Equation
5	8	.	e6,R(e7 or e8 or e9)	Constraint
6	(repeat sequence as in 5 for unit 2)			

* No directory.

\+ The e indicates entry numbers from Table 10.

** The quantity described by use is recorded in the last directory referenced. Thus, there is no file entry.

++ The . indicates any selection from this directory.

of a method in the case of record three. A method is a user prepared subroutine for supervising the convergence of a cyclic calculation during interpretation. The notion of a method is further pursued in Chapter VII. The entries of records two and three may be described as (tlfn), where n is the number of words per name.

The processing units are regarded in GENDER to be completely separate entities. To form a processing plant, the units must somehow be connected together to permit the transfer of material between the units. This physical connection of units is represented algebraically by a connection equation. For example, VCN201 = VCN422 is a connection equation relating variable code name (VCN) 201 to variable code name 422. If the scale factor for unit numbers employed in developing the unique code names is 100, then this example indicates the equality of variable 1 of unit 2 with variable 22 of unit 4. It is entirely possible that a single variable may appear in several connection equations. That is, many variables may share a common value. This leads to the notion of a common storage location for the value of all variables sharing a common value as stipulated by connection equations. Further, if the unit variables are connected by the sharing of value storage locations, the need for explicitly stating the connection equations in either SECEDE or the GENDER list is eliminated.

Unit 0 is contained within record four of SECEDE and is reserved for common variables. This record contains three files. The first file is an RAF and contains data relative to the dimensionality of the common variables. The similarity of function between this file and the DIMENSION statement in FORTRAN suggested the name declaration file. Apart from those containing relative pointers, each declaration file ASE will contain the fields (tlf(2+d)), where d is the dimensionality.

Field (tlf2) contains the dimensionality. The d fields reserved for the indices indicate the maximums of each index. The field (tlfl) is the pointer to the value storage location. The field is an absolute pointer which is computed by the program SECIAO during the establishment of SECEDE. In particular, the absolute address corresponds to an entry of the common variable value file. Although this file is only the second on the record, a LISTER of (4, 4, 3, 1) must be employed to permit access. The second directory entry is reserved for the number of common variables. The variable value file consists principally of entries of the form (tlfn), where n is the number of words per floating point value. All values are in floating point format. While a single entry containing (tlfn) is satisfactory for scalar variables, many such entries are required for dimensioned variables. If a dimensioned variable appeared on the file expanded into its components, the access of any variable, appearing later in the file than the dimensioned variable, would require knowledge of the number of dimensioned variable components. That is to say, the random access feature of the file is greatly compromised. As an alternative, let us consider partitioning the variable value file into two divisions. The primary division appears first in the file and contains one entry per variable. For scalar variables, the primary entry is simply the value. For dimensioned variables, the primary entry is of the form (tl7fl, tl8fl), with the fields containing respectively the integer 21 and a relative link to the first of a sequence of entries in the secondary partition. This sequence of entries encompasses the values of every component of the dimensioned variable. The integer 21 is provided as a safety measure but has proven to be of only occasional utility.

The third file of unit 0 is the common variable name file and is referenced by the fourth directory entry. The entries of the variable name file are identical to those of the operator name and method name files of records two and three.

Each record following record four is designated as a unit. The unit records are considerably more complex than any record we have previously considered.

A unit record comprises eight major file divisions, four of which are further subdivided by second level directories. The individual files are all of types we have already considered in the first four records. The first entry of the first level directory references an eight entry second level directory. Entries 1, 3, 5 and 7 of the second level directory reference declaration files for variables, ER's, equations and constraints. Directory entries 2, 4, 6 and 8 are respectively reserved for the numbers of variables, ER's, functions and constraints. The four declaration files are essentially the same as the common variable declaration file of record four.

The variable declaration file requires the addition of a data type 1 field between the value address and the dimensionality, making each entry of the form (tlf3+d). This additional field is reserved for the code name of a common variable. When a unit variable is declared common by a non-zero (tlf2) field, the common variable code name and value supercede those of the unit variable during preparation and interpretation of the GENDER list. That is, the common declaration substitutes for the connection equation. The remaining three declaration files require only the fields (tlf1+d), where the field (tlf1) is the dimensionality.

The second and third first level directory entries refer to the variable and constant value files. The variable value file is identical to the common variable file of record four. The constant value file is quite similar to the variable value file, except that all constants are single entities making the provisions for dimensioned variable components unnecessary for the constant variable file.

The ER and the variable name files correspond to the first level directory entries four and five. The name files are structured identical to the files of the second and third records.

The first level directory entries remaining correspond to the ER, equation and constraint files. Let us consider the equations. Each equation is recorded on a separate SAF. The equation SAF's are recorded on the equation directory, which is in turn recorded on the first level directory. For example, the LISTER vector (5, 5, 7, 1, 1) would be appropriate to accessing the first equation of unit 1. The equation SAF is not unlike the decision and tear variable SAF's. The first ASE pertains to the equation and is of the form (t1f3, 2xd). The field (t1f1) will always be the relative link in an SAF. The fields (t1f2) and (t1f3) are respectively the number of outputs and the dimensionality. The indices are recorded exactly as they were for the decision and tear variables. Each ASE after the first ASE represents a variable, operator or a constant. A field, (t5f1), is employed as an identification flag . Flag values range from 0 to 2 and correspond in order to an operator, a variable or a constant. The operator and constant entries are identical and consist of the fields (t1f1, t5f1, t16f1), with (t16f1) containing the operator or constant code name.

The variable ASE contains the fields (t1f1, t5f1, t7f1, t8f1, t19f1, 1xd). In order of appearance commencing with (t7f1) these fields are

an output selection cost flag, variable code name, dimensionality and the indices. The algorithm employed for output set assignment permits the use of weights to influence the output set selection process. The weights represent a cost of assignment and are scaled from 0 to 10. One word is reserved for each index which are in the seven part format described for the decision and tear variable files. The ER and constraint files are identical to the equation files except that operators and constants do not appear in the ER files.

* IV.4.b. SECIAO

In the previous chapter, it was noted that certain support structures to the sparse incidence matrix were of utility. Much the same situation exists here for SECEDE. The file entries are of varying compositions and sizes, suggesting the use of an information block to serve as a reference table.

The twenty-five word information block, mentioned in Chapter III with respect to networks and SIM's, is again pressed into service. Information block components 1 through 7 are reserved for data pertinent to SECEDE. Table 12 list these components. It should be noted that component 2, the length of a floating point value, is uniform throughout SECEDE. All variables will either all be single precision or all be double precision.

Apparently all of the ALLOC space required for SECEDE is in the form of record segments, making the use of a SPUR block a somewhat dubious feature. This is, however, not necessarily the case. In the previous section, the description of the variable declaration ASE's included a field reserved for the address of a storage location. While this is perfectly satisfactory for scalar variables, a single address may not be sufficient for a dimensioned variable. In particular, if

TABLE 12

Information Block Components Pertinent to SECEDE

Component	Use
1	Length of a record segment
2	Length of a floating point value
3	Length of a subprogram (ER or method) or variable name
4	Length of a variable ASE excluding indices
5	Length of a constant ASE
6	Number of record segments requested by each call to NEWCEL
7	Scale factor applied to unit numbers to develop unique code names for variables, constants, ER's functions and constraints

the value locations reserved for a dimensioned variable extend over several record segments, the single address will be inadequate. This situation is remedied by establishing for each dimensioned variable a dimensioned variable address resolution block, DIVARB. It is the address of the DIVARB which appears in the declaration file for a dimensioned variable. The DIVARB encompasses (2+d+r) words, where d is the dimensionality and r is the number of record segments containing values of the components to a dimensioned variable. The first word of the DIVARB contains the dimensionality d. The next d words contain the ranges of the indices. The (d+2)nd DIVARB word is called the offset. The offset is the integer which when added to the address of the record segment containing the first variable component will give the address of that first variable component. For example, suppose that the values of a dimensioned variable commence with the 2902 nd word of ALLOC. Further, suppose the record segment address is 2861. The offset to be recorded in the DIVARB would be 41. The remaining or DIVARB words are reserved for record segment addresses. The DIVARB permits rapid address resolution for a dimensioned variable component. The variable value file is a linear storage medium necessitating the conversion of multi-index sets into a single index. That is, the set of indices for an array component must be converted into the single equivalent index for the storage of the array as a vector. Except for the length of a record segment and the number of words per value, all of the data necessary for both the index conversion and the address resolution is contained within the DIVARB.

The need for a SPUR block is now some what obvious. The size of the SPUR block (i.e. the number of SPUR's required), however, is not clearly defined. The DIVARB size depends not only on the dimensionality,

but also on the number of components of a dimensioned variable. The dimensionality and number of components will remain unknown to SECIAO until input of the variable declarations commences. Although it would be possible to complete variable declaration input before preparing the SPUR block and establishing the DIVARB's, it is unnecessary to do so and may even result in the wastage of ALLOC space. A SPUR stipulating a list entry size larger than one half the ALLOC segment size will waste a portion of each segment allocated. This cannot be avoided. However, the SPUR itself can be eliminated. We shall reserve a single SPUR for all large DIVARB's, employing it by altering the list entry size contained in the fourth SPUR word. The SPUR block adopted for SECEDE consists of (2+(ALLOC segment size)/2) five word SPUR's. The first SPUR is provided for the requisition of DIVARB's larger than one half of ALLOC segment size. The second SPUR is appropriate to record segment acquisition. The remaining SPUR's are graduated in list entry size from 1 to one half of ALLOC segment size. It is the address of the SPUR for the record segment which is considered to be the address of the SPUR block.

The use of REMOTE requires a LOAD vector. In particular, two LOAD vectors are required by SECIAO for manipulating SECEDE. The LOAD vectors are furnished as a single vector 60 words in length which is called the LOAD block.

The entire service module is bound together by a service module block, SECBL. The first word of SECBL contains the address of the SECEDE catalogue. The remaining SECBL words are pointers to the SPUR block, the LOAD block and to an allocation of space reserved for index calculations.

* IV.4.c. GENDER List

We have already established that a GENDER list is actually a network. However, the GENDER list is far from being a simple network. This complexity includes, but certainly extends beyond, the concept of depth discussed briefly in Section IV.2. To assist in describing the details of the GENDER structure, Tables 13 and 14 summarize the data types and list entries embodying a GENDER list.

The fundamental GENDER list building blocks are nodes representing equations, ER's or constraints. Nodes in this category contain the fields (t1f1, t2f1, t3f1-2, t4f1-3, t5f1-3, t16f1, t17f1, t18f1, t19f1, t20f1, 2xd). The fields (t3f2, t4f3, t16f1, t17f1, t18f1, t19f1,t20f1, 2xd) are additional to fields required of all NETPAC applications. The value of the flag (t3f1) is 2 for these basic nodes. The identification flag (t3f2) indicates the exact character of the node. Its values may range from 0 to 7 and correspond in ascending order to an ER, an evaluated ER, an equation, solved equation, an evaluated equation, a constraint, a solved constraint and an evaluated constraint. A solved equation relates an output variable to a set of one or more input variables. The solved equation is simply an algebraic rearrangement of an equation. The solved constraint has been algebraically solved for a particular variable, making it analogous to an equation. An evaluated equation is best explained by example. Suppose we have an equation of the form

$$F(x,y,z) = 0$$

If we evaluate the equation for a given set of x, y and z values we obtain

$$F(x,y,z) = f$$

where f represents the value of the equation at x, y and z. If we were

TABLE 13

Data Fields for GENDER

Data Type	LEND	Use Number	Use
1	1,V,V,V	1	Forward link
2	V,V,V,V	1	Backward link
		2	Pointer to group
		3	Dimensionality
3	V,V,V,7+	1	Identification flag (0 diverger, 1 group, 2 group body, 3 insert)
		2	Group body protection flag (0 unprotected, 1 protected)
		3	Identification flag (0 ER, 1 evaluated ER, 2 equation, 3 solved equation, 4 evaluated equation, 5 constraint, 6 solved constraint, 7 evaluated constraint)
		4	Output/Decision variable selection flag (0 algorithm, 1 pre-chosen, 2 tear variable used by method)
		5	Identification flag (0 operator, 1 variable, 2 constant, 3 output variable, 4 equation value, 5 local constant)
4	V,V,V,V	1	Trace key
		2	Group body trace key
		3	Linear sequence number (for use by GENIAO)

TABLE 13

(Continued)

Data Type	LEND	Use Number	Use
5	V,V,V,V	1	Number of forward paths
		2	Number of backward paths
		3	Working counter for tracing
6	1,0,1,0	1	Index
		2	Dimensionality
		3	Maximum index
		4	Offset
		5	Pointer to record segment of variable value file
		6	Pointer to equation value
		7	Pointer to variable value
7	V,V,V,V	1	Pointer to group body
		2	Pointer to decision variable list
		3	Pointer to tear variable list
		4	Depth (0 if undimensioned group)
8	V,V,V,V	1	Pointer to SIM
		2	Unused, but reserved
		3	Unused, but reserved
		4	Method identifier

TABLE 13

(Continued)

Data Type	LEND	Use Number	Use
9	1,0,V.3+	1	Mapping flag (0 no map, 1 map to group index, 2 map to equation index, 3 map to another index of itself)
10	1,0,V,V	1	Mapping index
11	1,0,V,1+	1	Operation flag (0 multiplication, 1 division)
12	1,0,V,1+	1	Operation flag (0 addition, 1 subtraction)
13	1,0,V,V	1	Scale
14	1,0,V,1+	1	Operation flag (0 additon, 1 subtraction)
15	1,0,V,V	1	Offset or index
16	V,V,V,V	1	Pointer to link editor list
		2	Pointer to equation value block
17	V,V,V,V	1	Equation, ER or constraint code name
		2	Pointer to tear variable block
		3	Dimensionality
18	V,V,V,V	1	Pointer to output variable list
		2	Number of pointers
19	V,V,V,V	1	Dimensionality
20	V,V,V,V	1	Number of outputs
21	V,V,V,V	1	Variable code name
22	V,V,V,V	1	Dimensionality
23	V,V,1,0	1	Value of local constant

TABLE 14

The List Entries of GENDER

List Entry	Composition	Use Number	Use
1	t1fl,t2flul,t3flul,t3f2u2, t4fl-3,t5fl-3, t7fl-4,t8fl-4, (t9-15)x2	1	Group
2	t1fl,t2flu2,t3flul	1	Diverger
3	t1fl,t2flul,t3flul,t3f2u2, t4fl-3,t5fl-3,t16flul, t17flul,t18flul,t19fl, t20fl,(t9-15)x2d	1	Group body
4	t1fl,t2flu3,t3flu4,t21fl, (t9-15)xd	1	Output variable
5	t1fl,t2flu3,t3flu4,t21fl, (t9-15)x2d	1 2	Decision variable Tear variable
6	t1fl,t2flu4,t3flu5,t21fl, t22fl,(t9-15)xd	1	Variable or equation value on link editor list
7	t1fl,t2flu4,t3flu5,t21fl	1 2	Constant on link editor list Operator on link editor list
8	t1fl,t3flu5,t23fl	1	Local constant on link editor list
9	t6flu2,t6f2-(d+1)u3,t6f(d+2)	1	Dimensioned variable address

TABLE 14
(Continued)

List Entry	Composition	Use Number	Use
	u4,t6f(d+3)-(d+2+n)u5*		resolution block
10	t1f1,t2flul,t3flul,t4fl-3, t5fl-3,t16flu2,t17flu2, t18flu2	1	Group body insert for method
11	R(t6u7)	1	Function value block
12	R(t6u8)	1	Variable value block

If we were to include a "variable" representing f in the original equation F, and if we were to perform whatever algebraic rearrangement might be required to solve for the "variable" f, the result would be an evaluated equation. The evaluated ER and evaluated constraint are analagous to the evaluated equation. Evaluated equations, ER's and constraints appear only in cyclic calculations where they are employed for adjusting the values of tear variables.

In order to consider the algebraic rearrangement of, for instance, a function, it is necessary to have a list of the components to the equation. This list is called the link editor list, the pointer to which is field (t16f1). Further discussion of the link editor list is momentarily deferred until the discussion of the nodes currently under consideration has been completed. The fields (t17f1), (t19f1) and (t20f1) are respectively the code name (for the equation, etc.), the dimensionality and the number of outputs. Two words are served for each index, to record an index range in the seven part format. The pointer to the output variable list is field (t18f1). The output variable list consists of forward-only linked list entries containing the fields (t1f1, t2f1, t3f1, t21f1, d). Field (t21f1) is the output variable code name . The flag (t3f1) will normally be 0, except when an output variable is also a tear variable and the function, ER or constraint is to be evaluated rather than solved for its output. In this case, the flag will be 2. The dimensionality is recorded in field (t2f1). One word only is reserved for each of the seven part indices.

The link editor list of an equation must contain distinct entries for variables, operators and constants. A flag, data field (t3f1), is employed to identify the type of equation component represented by a list entry. The flag values 0, 1 and 2 correspond to operator, variable and

constant list entries.

The operator and constant list entries are identically structured as (t1f1, t2f1, t3f1, t21f1). Data type 1 is the forward link, and data type 3 has already been discussed. The field (t21f1) contains the code name of the operator or constant. For the constant, field (t2f1) is an absolute pointer to the constant value. For an operator, this field specifies the order of the operator.

A variable entry on a link editor list contains the fields indicated for a constant plus the field (t22f1) and one word per degree of dimensionality. The field (t22f1) is the dimensionality. The pointer to the variable value is (t2f1), and the variable code name appears in (t21f1).

So far, we have considered variables, constants and operators with respect to the link editor list. One might be tempted to conclude that these three entries are sufficient for all functions, ER's and constraints. However, this is not the case. Recall the notion of an evaluated equation. For an evaluated equation, a link editor list entry must be provided to represent the equation value. This entry is identical in structure to the variable list entry. In fact, the data fields serve the equation precisely as they do a variable.

A further entry type may be seen by considering the result of an algebraic rearrangement of an equation or a constraint. In solving for a particular variable (i.e. the output variable), it may be possible to combine some of the constants appearing in the equation or constraint. Rather than expand the constant name and value files, a second type of constant list entry has been invented. It contains only the forward link, the flag and an one or two word field reserved for the constant value. We shall differentiate the constants verbally by naming this

second type a local constant. The flag values for the equation value and local constant entries are 4 and 5 respectively. The flag value 3 is reserved to indicate which variable(s) is the output of a function, ER or constraint.

The connection of the basic GENDER building blocks into a network forms what is termed a group body. The group body is the possession of a node called a group. Generally, a group represents a collection of equations, ER's and constraints having a common trait. For instance, the members of a group body might all be participants in a cyclic calculation. A group node is formed of the usual network fields plus (t3f2, t4f3, t7f1-4, t8f1-4), of which it is the field (t7f1) that points to the group body.

The flag (t3f2) differentiates between a protected group (set flag) and an unprotected group (flag unset). The unprotected group may be modified by the analysis algorithms included in GENDER, whereas the protected group may not altered. An example of a protected group might be a completed solution procedure, perhaps retrieved from a solution procedure library, requiring no further analysis. The field (t4f3) appears in every GENDER list node and is provided as a programming convenience for GENIAO. The data fields (t7f2-4) and (t8, f1 and 4) are, in order, the pointer to the decision variable list, the pointer to the tear variable list, the depth, the pointer to the SIM and the method identifier. The pointers (t8f2) and (t8f3) are reserved for use with the dynamic programming algorithms to be added to GENDER at a future date. The entries of the tear and decision variable lists are identical to the entries on the output variable lists for functions, ER's and constraints, except that two words are reserved for each degree

of dimensionality. Each dimension requires two words so that a minimum and maximum to each index may be specified. This feature permits compactness with respect to dimensioned variables assigned as tear or decision variables.

* IV.4.d. GENIAO

The implementation of GENIAO was relatively straightforward, depending heavily on the COAST and NETPAC verbs. As was the case for SECIAO, GENIAO requires the reservation of seven words in the information block for data pertinent to the GENDER list. Table 15 lists these components with their respective contents.

The GIMBL is the GENDER information block, and is the analogy of SECBL for SECIAO. The first word of GIMBL points to the GENDER list. The address of the first GIMBL word is recorded in JCCORE(16). The second GIMBL word points to the SPUR block for list type 2. The SPUR block contains one five word SPUR for each possible list entry size from unity up to and including the largest possible GENDER list entry. The SPUR's are arranged in the block in ascending order. The third GIMBL word is reserved for the trace key applicable to the GENDER list at level 0. The remaining seven words of GIMBL contain the addresses of vectors employed during index calculations.

Each of the seven vectors is equal in length to the maximum possible dimensionality for a function, ER, constraint or variable. This integer is obtainable from the fifteenth word of the information block. Let us label the vectors A through G in the order they are referenced by GIMBL. The vectors A through G correspond exactly to the vectors a through g of Appendix B.

Although the GENDER list is principally a NETPAC implementation, REMOTE is employed by GENIAO. For both the input and output of a

TABLE 15

Information Block Components Pertinent to GENDER List

Component	Use
8	Length of a diverger list entry
9	Length of a group list entry
10	Length of a group body list entry*
11	Length of an output, decision or tear variable list entry*
12	Length of a variable list entry on the link editor list*
13	Reserved
14	Length of a group body insert for method

* Indices excluded

GENDER list, temporary storage is required for linkage data pertinent to the nodal connections. With rapid access being the paramount consideration, a file structure was selected for the temporary storage. Two files are employed, one for the current GENDER list level and one for the previous level. As the input of the current GENDER list level is completed, the files are interchanged so that the current level file now becomes the previous level file. Each file entry contains the address of a node and a pointer to an auxiliary list. The list entries each contain the file entry number of a predecessor node to the node labeled in the file entry.

CHAPTER V

ANALYSIS ALGORITHMS

The current selection of algorithms available with the GENDER System is somewhat limited. Though limited, the algorithms nevertheless do permit the development of a complete solution procedure according to a realistic analysis strategy. The selected strategy involves resolving cyclic calculations via minimum tear considerations.

Additional algorithms can be relatively easily provided. Several such additions are contemplated for GENDER in the near future. These additions are futher discussed in the concluding remarks of Chapter IX.

The rationale for the selection of algorithms for inclusion in GENDER depended upon two principal criteria. First, the algorithm must be of obvious utility with respect to solution procedure development generally. Secondly, the algorithm should be as simple as possible. This latter criterion arises since the algorithm must currently serve as a test and evaluation vehicle for the support and administrative facilities of the GENDER System.

The next three sections of this chapter describe the analysis algorithms selected to perform out put set assignment, precedence ordering and minimum tear selection. Rather than reproduce in this work the algorithms stepwise, reference will be made to the literature source of the algorithm version implemented. To be described in the following sections, then will be the essential characteristics of the algorithms as pertaining to the design engineer and to GENDER. The literature

references may be regarded as * flagged sections.

V.1. Hungarian Output Assignment Algorithm

The Hungarian algorithm for output assignment selected for GENDER is the version presented by Gupta (1972) in his master's thesis. To each variable in each equation, ER and constraint a cost or weight of selecting the variable as the output must be assigned. The selection of weights will, in general, reflect the design strategy.

The design engineer may simply assign equal weights to all variable/equation incidences. By so doing, he has reduced the Hungarian algorithm to an arbitrary output assignment procedure. This strategy minimizes the effort expended by the design engineer in obtaining an output assignment.

A second weight selection criterion, closely allied to the proceeding one, is user preference. The design engineer may wish to influence the Hungarian algorithm to make certain output assignments while avoiding certain other assignments. Thus, he would ascribe low weights to preferred assignments and high weights to undesired assignments. Although the weight range may be as large as desired the range 0-9 is envisioned to provide sufficient resolution to convey user preference.

Although a design engineer may weight each variable/equation incidence according to preference, it is more likely that a specific weighting scheme would be applied. For example, Lee and Ozawa (1971) and Soylemez and Seider (1972) suggest weighting on the basis of the algebraic complexity of solving for a variable in the event of selection as the output. While this increases the effort expended by the engineer, a more efficient solution procedure will likely result.

A final scheme for weight selection involves the determination of the sensitivities of each variable/equation incidence. If the weight is assigned as the negative of the logarithm of the sensitivity, than the Hungarian algorithm, which seeks to minimize the sum of the weights for a complete output assignment, will be performing output set assignment according to the maximum product criterion (Edie and Westerberg, 1971). This criterion has been shown to lead to rapidly convergent solution procedures to cyclic equation sets.

Chapter VII is a brief user's manual and discusses the use of the Hungarian algorithm. The subprogram performing this algorithm is designated HASSAL. As is the case for all of the analysis algorithms, the passed parameters required by HASSAL are minimal, being only the address of a group and a unique call statement identification number. The latter is an essential feature of the debug facilities incorporated into COAST and described via comments in Appendix A.

HASSAL operates on a single group, although this group may have groups as nodes on its group body. The effect of HASSAL on a group is the substitution of a new output assignment for the previous output assignment. The previous output set may, of course, be null. Unassigned variables at the conclusion of output assignment become decision variables and are recorded in the group.

V.2. Speed-Up Precedence Ordering Algorithm

The Speed-Up algorithm for precedence ordering is the product of research by Sargent and Westerberg (1964). Two algorithms are presented in the reference, but only algorithm I has been selected for use with GENDER. The subprogram name is SPEDUP.

SPEDUP performs precedence ordering of a group body on which the output set assignment is recorded. Only the address of a GENDER group and a call statement identification number are required as passed parameters by SPEDUP. The use of SPEDUP in an analysis strategy is discussed in Chapter VII.

The effect of SPEDUP is the reorganization of a group body to reflect precedence ordering. If the group body possesses an unresolved cyclic character, SPEDUP will create a new group containing only the equations which are not acyclic. That is, the cyclic equations are transferred from the original group body to the group body of a new group which becomes a member of the group body to the original group. The original group is, therefore, acyclic at the conclusion of SPEDUP. The sub-group into which the cyclic character has been concentrated may now be subjected to tear variable selection.

V.3. Barkley - Motard Minimum Tear Algorithm

The Barkley and Motard (1972) minimum tear algorithm was presented in a paper at the NATO Advanced Study Institute on Decomposition as a tool for solving large problems held in Cambridge, England. The program executing the Barkley - Motard algorithm is the subroutine BEMOAN.

BEMOAN requires the same passed parameters as HASSAL and SPEDUP. The GENDER group on which BEMOAN is to operate must posses an output set assignment. After selection of the tear variables has been made, the selections are recorded in the group. The deletion from consideration of the tear variables in all functions except where they are the output variables renders the group body acyclic. This strategy and the program SPEDUP are combined to precedence order the group body as the final BEMOAN step.

V.4. Implementation Details

Each of the analysis algorithms depends upon the existence of a sparse incidence matrix for the group under analysis. The concept of the SIM and the SIM manipulation verbs of SIMPAC were developed expressly for the implementation of analysis algorithms. Although the SIM is not always the appropriate medium for the execution of an algorithm, it is the most convenient medium for conveying to an algorithm the pertinent data relative to a group body.

An important feature shared by the analysis algorithms, and indeed by most of the major GENDER programs, is the high degree of subroutinization. While the delegation of tasks among subprograms can be carried to the extreme, a certain degree of task delegation greatly simplifies the debugging operation and enhances the readability of the code. HASSAL, SPEDUP and BEMOAN are hardly more that supervisor programs directing the operation and interaction of their respective subroutine sets.

In addition to the SIM, each of the analysis algorithms expects the existence of certain conditions before execution is attempted. For example, BEMOAN requires an output assignment before it can proceed. To save the design engineer from at least a portion of the details of solution procedure generation; each of the three analysis programs automatically generates missing information before proceeding. A subprogram DFAULT is employed to provide this service and is more fully described in Chapter VII.

Continuing the pattern established in previous chapters, the following section provide the implementation details and are generally only of value to readers contemplating modifications to existing programs or expansion of the analysis algorithm repertoire. Before attempting the

next three sections, it is strongly advised that familiarization with the Hungarian, Speed-Up and Barkley - Motard algorithms be acquired. Only the adaptation of the algorithms to the data structures available with GENDER will be discussed in the ensuing sections.

* V.4.a. HASSAL

HASSAL is by far the largest and most complex of the three analysis algorithms. Gupta (1972) presents the Hungarian algorithm in the form of two interactive algorithms. One of the algorithms resembles a supervisor directing the utilization of the other algorithm. It is essentially after this supervising algorithm that the program HASSAL is patterned. The steps of the algorithm for performing the output set assignment are parcelled among several subprograms.

The major subprograms employed by HASSAL consist of J2ACYC, J2AOSA, J2PATH, J2MARK and J2SWIT. The subprogram J2ACYC is the acyclic output assignment phase of the algorithm (steps 1 through 4 of Cruptas algorithm). Step 5 represents a decision and transfer of control step exterior to the domains of J2ACYC and of the next algorithmic phase, J2AOSA. Such control steps have been incorporated into HASSAL. Steps 6 through 8 constitute the instructions incorporated within J2AOSA. J2AOSA performs an arbitrary output set assignment. The remaining subprograms under the guidance of HASSAL expand the arbitrary output assignment into a complete assignment. A complete assignment is characterized by output variables assigned to every equation. Not all variables need be assigned; those remaining unassigned become decision variables.

J2PATH and J2MARK are both involved in the search for alternative output assignments which will permit the extension of the output set.

J2PATH performs steps 10 through 14 of the assignment algorithm. Alternative assignments are sought via a row and column marking scheme. Marking is also employed by steps 16 through 17, which are embodied in J2PATH. The marking strategy employed by J2PATH is some what more complex than that of J2MARK. Both programs result in the identification of Steward paths. J2SWIT is employed whenever a Steward path is identified to perform the output set adjustments. For paths discovered by J2MARK, J2SWIT must break one output assignment and make two new output assignments. For paths discovered by J2PATH, three output assignments are replaced by four new assignments. In either case, a gain of one assignment in the output set is noted.

HASSAL and its subprograms operate entirely on the sparse incidence matrix. A working ordinate is created for both the rows and columns. The working ordinate file entries are identical to the file entries of the primary ordinates. The principal function of the working ordinates is for the recoding of row and column marks. Marks are stored in data type 13, which is the data type employed by the primary ordinates for storage of output assignments. Thus, as a programming convenience, the SIM routine for output set assignment is employed for marking the working ordinates. No other temporary data structures are required for the implementation of the Hungarian algorithm.

* V.4.b. SPEDUP

The Sargent and Westerberg (1964) Speed-Up I algorithm was by far the easiest and most natural to implement. The algorithm was developed expressly for list processing. The authors propose four list structures for use with the Speed-Up I algorithm. With a sparse incidence matrix available to SPEDUP, two of the four lists need not be generated.

List 1 is an unordered list of all equations, ER's and constraints. The row ordinate of the SIM is a file containing references to the functions, ER's and constraints which SPEDUP is to reorder. Consequently, the row ordinate is substituted for list 1.

The lists of type 2 are described in the paper as providing data on the linkages between the elements to be precedence ordered. A type 2 list must be provided for each element to be ordered. Let us consider an SIM row. Each SIM row contains an output variable (perhaps several outputs for an ER), one or more input variable and perhaps one or more decision variables. The input variables, by virtue of their not being either outputs in a particular row or decision variables, must each be assigned as outputs to other SIM rows. Thus, the SIM rows furnish the linkage data directly and eliminate the need for creation of the type 2 lists. Note the one to one correspondence between row ordinate file entries and SIM rows.

Lists 3 and 4 must be created by SPEDUP. List 4 is employed as temporary storage during execution of the algorithm, with the final precedence ordering appearing on list 3. The algorithm stipulates the transfer of entries from list 4 to list 3 making it necessary to design only a single list entry data format serving both lists 3 and 4. The data format selected was (t1fl, t2fl, t3fl, t21fl, t22fl), with the list being GENDER type 2. Tables 16 and 17 are provided as a guide to the structure and contents of the entries on lists 3 and 4. Data types 1 and 2 serve as the forward and backward links as is usual with forward - backward lists. Data type 3 is a flag serving to differentiate equations (0) from decision variables (1) and groupings of equations (2). The contents of data types 21 and 22 depends upon the flag value.

TABLE 16

Data Fields for SPEDUP Working Lists

Data Type	LEND	Use Number	Use
1	V,V,V,V	1	Forward link
2	V,V,V,V	1	Backward link
3	V,V,V,1+	1	Flag (0 equation, ER or constraint, 1 decision variable, 2 grouping of equations, ER's and constraints)
21	V,V,V,V	1	SIM row number (flag = 0)
		2	SIM column number (flag = 1)
		3	Address of a list of grouping members (flag = 2)
22	V,V,V,V	1	Address of an SIM row element (flag =0)
		2	Address of a grouping member (flag = 2)
		3	Wasted

TABLE 17

List Entries on SPEDUP Working Lists

List Entry	Composition	Use
1	t1fl,t2fl,t3fl,t21flult22flul	Equation, ER or constraint
2	t1fl,t2fl,t3fl,t21flu2t22flu3	Decision variable
3	t1fl,t2fl,t3fl,t21flus,t22flu2	Grouping

If the value of the flag is zero, the list entry represents an equation, ER or constraint. In this situation, data type 21 contains the SIM row number and data type 22 contains the address of a row element. In particular, data type 22 refers to the next input of an equation, ER or constraint to be considered by SPEDUP. Data type 22 will initally point to the first element of the row.

If the value of the flag is two, the list entry represents a grouping of equations, ER's and constraints. Such groupings indicate cyclic information flow patterns among the group members. Data type 21 contains the pointer to a list of grouping members. The list entries on this auxiliary list are each identical to the list 3/4 entry with the flag set to 0. Data type 22 is also a pointer to an entry on the auxiliary list, but not necessarily pointing to the leading entry as is the case for data type 21. Data type 22 points to the first member of the grouping having a non-null input list. This pointer is advanced to the next grouping member when, during SPEDUP execution, the input list of a grouping member is depleted.

A flag value of 1 indicates the list entry to be representative of a decision variable. In this case, only data type 21 contains meaningful data. Data type 21 contains the SIM column number of the decision variable. Data type 22 is ignored.

The implementation of the Speed-Up I algorithm does not exhibit the degree of decentralization observed for HASSAL. This is principally due to the design of the algorithm. The majority of the steps are simple; that is, they do not contain instructions to perform complex or extensive manipulations. Further, the overall manipulations demanded by Speed-Up I are much less complex than those required for HASSAL.

* V.4.c. BEMOAN

BEMOAN is the supervisor program for the Barkley and Motard (1972) minimum tear algorithm. The algorithm employs precursor and interval lists to search initially for self-loops, then for two-way edges in an effort to reduce to null the set of interval lists. Nullity of the interval list set indicates complete elimination of the cyclic character in a group body by virtue of tear variable selections.

Instead of employing separate lists as suggested in the algorithm, the BEMOAN inplementation incorporates the list information into a single column ordinate. This working ordinate is composed of file entries containing the data fields (t3fl, t4fl, tl4fl). The LEND summary for these data fields is given by Table 18. The field (tl4fl) is a flag whose possible settings of 0, 1 and 2 respectively indicate the file entry to represent the head of an interval, an interval member and a tear variable. Note that the head of an interval is also a precursor either to itself or to another interval. The field (t3fl) contains the column number of the interval to which a variable belongs. This field is only of significance for a flag of 1. The field (t4fl) is used variously for counting membership of intervals in two-way edges or occurrences in other intervals. The number of occurrences of an interval in the other intervals is useful in selecting a tear variable when neither a self-loop nor a two-way edge is present. If the flag value is 2, both (t3fl) and (t4fl) contain no significant data.

TABLE 18

Data Fields for the BEMOAN Working Ordinate

Data Type	LEND	Use
3	1,V,V,V	SIM column number (the interval to which a variable belongs)
4	V,V,V,V	A working data field for temporary storage (used primarily as an accumulator)
14	V,V,V,V	Flag (0 variable is head of an interval, 1 variable is an interval member, 2 variable is a tear variable)

CHAPTER VI

INTERPRETATION

The GENDER list contains all of the instructions for solving an equation set. During the course of GENDER list development via the algorithms, no modifications are imposed upon the original equation set. The equation set stored in SECEDE serves only as a source of information. When a design engineer employs GENDER, the ultimate objective is, of course, the solution to the equation set and not merely the instructions for obtaining that solutions.

Two choices are apparent at this point. The engineer may impose the algebraic rearrangements and precedence ordering on the equation set as per the GENDER list instructions. He may then program, for instance in FORTRAN, the evaluation of the output variables, including the resolution of cyclic calculations. It is possible to provide a FORTRAN converter as an integral component of the GENDER system.

While FORTRAN conversion of the GENDER list could have been provided, it was rejected for the present version because of the difficulty in performing modifications to the FORTRAN source module. Changes to the source module might originate from reanalysis of a cyclic group if convergence problems are encountered, or they might originate from alterations in the active constraint set during constrained optimization. Even assuming that the modification of FORTRAN code could be conveniently accomplished, re-compilation would remain a necessity.

The foregoing considerations led to the examination and ultimate

selection of an interpreter system. By interpretation we mean execution directly from the GENDER list. The interpretation of a GENDER list is a three step process; link edit, convert and interpret. The link edit step creates for each equation, ER and constraint a list of components with variable value storage locations explicitly addressed. The conversion step performs any required algebraic manipulations on the link editor generated lists.

The following sections describe the interpretation capability designed for GENDER. Since the advantages of interpretation will not be fully realized until the advent of a constrained optimization supervisor for GENDER, and since the creation of this supervisor is somewhat distant, the interpretation capability has not been fully implemented. To ensure compatability with the currently implemented GENDER components, the essential features of interpretation have been carefully analyzed. The remainder of this chapter reflects the results of this analysis. The inclusion of this information in a description of GENDER has been prompted by two factors. (1) It will form the basis for the implementation of the interpreter, and (2) the interpreter (coupled with the notion of constrained optimization) provides an insight into the justification for the implementation of GENDER as a complex, list processing-based system.

VI.1. Link Editor

At the completion of analysis, the GENDER list contains all of the instructions necessary to solve an equation set. However, the GENDER list contains only the instructions, for the equation set is stored in SECEDE. As files in SECEDE, the equation set is virtually unavailable with respect to performing the modifications recorded on the GENDER list - i.e. algebraic rearrangements. Therefore, it is necessary to generate

a second copy of the equation set in a form amenable to modification.

The link editor, LINKED, provides each group body entry with a linked list of the constituents in the equation, ER or constraint represented by the group body entry. Thus, the second copy of the equation set is distributed over the entire GENDER list. This distribution has the characteristic of automatically conforming to the precedence ordering exhibited in the GENDER list.

Now let us consider a variable appearing on a link editor list (LEL). In terms of interpretation, the really important aspect of the variable is not its code name, but is its value storage location. Since the variable value files are integrated into SECEDE, the use of REMOTE verbs is necessary to determine value storage addresses. While it is of no consequence for acyclic groups, deferring address determinations until interpretation causes the address of a variable to be determined on every iteration during convergence of a cyclic group. Consequently, each LEL entry representing a variable value is provided by the link editor with the address of the variable value storage location.

VI.2. Converter

The converter, COVERT, has three primary duties: (1) solve LEL's algebraically for the output variable, (2) install equation, ER and constraint evaluation list entries into LEL's, and (3) provide a method insert to the group bodies of cyclic groups. The converter is programmed to accomplish all three tasks with only a single trace through the GENDER list.

A method is a subroutine required to direct the successive selection of tear variable values as a means of accelerating the convergence of cyclic groups. Methods are identified in the GENDER system by the method code. This code is recorded in the data field (t8f4) of each group. A

code of unity indicates a group to be acyclic, while a code of zero indicates that a group has not yet been subjected to the analysis algorithms. Thus, the reference by the method code to actual methods begins with the code 2.

All cyclic groups, that is all groups having a method code of 2 or larger, must have a method insert as the last accessible node of the group body. The order of access is, of course, dictated by the verb NEXT. The contents and structure of the method insert has already been presented in Tables 13 and 14 in Chapter V. The purpose of the method is twofold: (1) to indicate to the interpreter that repitition of the group body is required and (2) to provide information required by the method subprogram.

In general, a method subprogram requires a vector of values for evaluated equations, ER's and constraints, and a vector of tear variable values. The evaluated entities and the tear variables will be equal in number. The objective of the method subprogram is the selection of tear variable values which drive the values of the evaluated entities as close to zero as practicable. Since some convergence accelerators, i.e. methods, rely on a history of tear variable and evaluation values over several iterations, this data is provided as an auxiliary to the method insert.

The current or latest values for both variables and evaluated entities are provided as a linked list containing the addresses of storage values. That is, these lists are essentially LEL's. The old values for both tear variables and evaluated equations, ER's and constraints are provided via an application of REMOTE. A single catalogue references two records. The first record contains a single file of tear variable values. Each file entry contains all of the old values for a particular tear variable. Therefore, if the last 5 values (excluding the current

value) are required, then each file entry will accomodate 5 value fields. To avoid unnecessary shifting of the contents of a file entry when a current value becomes the most recent old value, an indexing scheme is proposed. An index is stored in the method insert which indicates which of the old values is the most recent. For example, an index of 2 indicates the values in the second value field of each file entry to be the most recent. By subtracting unity (modulo 5) one determines the value field to receive the current value just prior to the creation of a new set of current values. The twenty-second word of the information block (whose address is JCCORE(17)) contains the number of old values to retain. The second record, identical to the first, is reserved for the values of evaluated equations, ER's and constraints.

The second of the three functions performed by COVERT involves the addition of evaluation list entries to the LEL's of equations, ER's or constraints. In particular, such modification is performed on entities declaring a tear variable as the output. Further, this modification is only performed if the method code stipulated in the group node is greater than five. The method codes two through five are special cases in which the tear variables are calculated as output variables. Thus, for codes two through five, the method subroutine must rely only on the current tear variable values and perhaps a history of old tear variable values to provide an estimate of the tear variable values for the next iteration.

The first of the three COVERT duties enumerated in the first paragraph of this section requires the capability of algebraically manipulating an LEL. All of the LEL manipulative power has been concentrated in a general algebraic package, GALAP. This program package has

been developed expressly for the GENDER system as an independent research effort. The listings for GALAP have been included in Appendix A. User interface with GALAP is via the programs SOLVE and MATH. SOLVE permits the algebraic solution of an equation or constraint for its output, while MATH determines the value of an output variable based on the current value of the input variables to an equation or constraint. As one might suspect, it is SOLVE which is employed by COVERT.

VI.2.a. SOLVE

The program SOLVE is employed to solve algebraically for a particular variable. The variable of interest is indicated to SOLVE by the occurrence of a flag value 3 for an output variable or a flag value 4 for an evaluation entry. The evaluation entry is viewed by SOLVE to be identical to an output variable. SOLVE accomplishes the algebraic rearrangement by initially expanding the LEL into a network structure. The algebraic rearrangement is actually performed on the network, which is collapsed back into an LEL after the algebraic manipulations have been completed.

Algebraically speaking, SOLVE is not always capable of solving an equation or constraint for its incident variable. If purely algebraic manipulations cannot solve an equation for a particular output variable, then SOLVE must fail. For example, algebraic manipulations alone cannot reduce the equation

$$x + y = \ln X$$

to the form

$$x = f(y)$$

SOLVE is, however, empowered to recognize and perform many simplifications. For example, the equation

$$x + y = 2x$$

becomes

$$x = -y$$

if x is designated as the output variable. Further, the equation

$$2(x + 2) - y = 9$$

becomes

$$x = y/2 + 2.5$$

again with x as the output variable. In order to accomplish the combination of constants, the constant entries on the LEL are released and new constants introduced. The new constants are termed local constants. That is, they are constants appearing only in one equation, with each equation having its own set of local constants.

Table 19 indicates the operators which SOLVE is capable of recognizing and manipulating.

VI.3. Interpreter

Put quite simply, the task of the interpreter is the determination of the values for all variables except the decision variables. However, the size and complexity of the interpreter program GLINT indicates the presence of some latent complexity in this simple task.

With respect to acyclic groups, the assignment of values to output variables is relatively simple. Each equation, ER and constraint constituting an acyclic group need be considered once only. The GALAP program MATH is employed by GLINT to effect all output variable evaluations for solved equations and constraints. The presence of an ER on an acyclic group body somewhat complicates the procedure since the calling of the subprogram (i.e. ER) is necessary. The procedure for calling subprograms is the subject of section VI.3.b.

The necessity of calling method subprograms further complicates

TABLE 19

Operator Set

Unary		Binary		Other	
Symbol	Code	Symbol	Code	Symbol	Code
COMP	1	=	101	+n	201
SIN	2	-	102	*n	202
COS	3	/	103		
TAN	4	**	104		
ARCSIN	5				
ARCCOS	6				
ARCTAN	7				
ALOG	8				
EXP	9				
RECIP	10				
SQRT	11				
SQUARE	12				

the resolution of cyclic groups, which by their cyclic character automatically require the repetitive evaluation of all concomitant output variables. The aspect of repetition requires that data be saved which will permit the reinitiation of the calculations at the beginning of a group body should satisfaction of the convergence criterion fail at the termination of an iteration. The program METHOD, discussed in section VI.3.b, is responsible for evaluating a group with respect to the convergence criterion, unless such a comparison is performed by the method subroutine.

If the interpreter encounters an error condition, the interpreter will return to the calling program the address of the group under consideration and a code indicative of the particular error. Table 20 lists the error codes which the interpreter is designed to recognize. The executive program prepared by the design engineer must be programmed to follow a recovery procedure for each error code.

VI.3.a. MATH

Like SOLVE, MATH is constrained to equations and constraints constructed with the operator set tabulated in Table 19 of section VI.2.a. MATH exists in two versions; one for single precision floating point arithmetic and one for double precision floating point arithmetic. The design engineer must select the MATH version appropriate to the accuracy demanded for the design computations. Since all variables must be either single or double precision, election to employ double precision may cause a significant increase in memory requirements. In view of the numbers of other machine instructions required to support the interpretation process, selection of double precision should not affect execution times.

VI.3.b. Subroutines

The subroutines which the interpreter GLINT must call are ER's and

TABLE 20

GLINT Error Codes

Error Code	Condition
1	Zero method code
2	Missing output assignment
3	Link editor list missing
4	No LEL entry flagged as output
5	LEL syntax error
6	Tear variable list missing for cyclic group
7	Method insert missing for cyclic group
8	Convergence excessively slow

methods. The design engineer must provide the programs EXTRA (for ER's) and METHOD (for methods). Essentially, each of these programs is a library of call statements for the various ER or method subroutines. The ER and method code names are employed to access the appropriate call statement via a computed GO TO statement. Note that for an ER, two computed GO TO's must be employed: one conditioned on the unit number and a second conditioned on the ER code name exlusive of the unit number modifier.

This procedure for calling subroutines during GENDER list interpretation may seem somewhat crude. However, in light of the original decision to remain machine independent to the largest possible extent, the use of the call statement libraries EXTRA and METHOD is inescapable.

In addition to the library functions of EXTRA and METHOD, these programs must also perform transformations on the data to be provided to an ER or method. That is to say, most ER and method subroutines accept values as data rather than the addresses of values so prevalent in GENDER list interpretation. The value address of interest to EXTRA are distributed within the LEL's of the ER's. The ER output variables will be indicated by LEL entry flags equal to 3 just as for the output of an equation or constraint. For METHOD, the value addresses are available from an LEL-type list and an old value file. One list/file pair will be available for both the tear variables and the evaluated entities (equations, ER's and constraints).

Since it is the duty of METHOD to employ a method subroutine for the purpose of returning to GLINT a new set of tear variable values from which new values for the evaluated entities are to be determined, METHOD must update the old value storage files. The nature of the update procedure is suggested in section VI.2. Finally, METHOD must be programmed

to test for convergence when employing any method subroutine into which a convergence test has not been incorporated. Typically, a convergence test would involve the comparison of the latest tear variable value with its predecessor as for example in a sum of the squares of the differences technique. METHOD indicates satisfaction of a convergence test by returning zero to GLINT in substitution for the address of the method insert.

VI.4. Implementation Details

In the subsections to follow, we shall concentrate on the implementation details for LINKED, COVERT and GLINT. As per the pattern established for previous chapters, these sections are marked with an * and may be omitted by all readers not requiring this detailed data.

The information of the ensuing subsections is not necessary to the preparation of the programs EXTRA and METHOD. Sections VI.2 and VI.3.b provides, when used in concert with Tables 13 and 14, sufficient information on the data structures involved. The user must thoroughly understand the LEL, method insert and old value storage structures. The programming tools are provided principally by COAST and REMOTE. In accessing LEL entries for an ER, LNKFWD and FRMCEL should be used to effect the advance and data transfer operations rather than POPUP. Although POPUP does perform the advance and data transfer operations automatically, POPUP also destroys the list as it proceeds. Remember that for cyclic groups or in the event of group reanalysis, the LEL's will be utilized repetitively.

* VI.4.a. LINKED

LINKED is basically a supervisor program for the link editing step. NEXT is employed to access the group body entries of either a single group or an entire GENDER list. LINKED utilizes the group body flag

(t3fl) to identify the particular group body constituents, each of which is treated separately.

A group on a group body causes no action by LINKED save to access the body of the group via NEXT. Equations, ER's and constraints all carry group body flags of 2. All entries with flags of 2 are referred to the IOPAC program JlLEDL for LEL generation. The flag value 3 is reserved for method inserts. The LEL-type list entries, as well as the old value storage files, are released via CORN for later reallocation. The field (tl6fl) is the address of the SPUR employed with CORN.

* VI.4.b. COVERT

COVERT, like LINKED, is primarily a supervisor program. Also like LINKED, COVERT employs NEXT to access the group bodies. For the group body entries representing equations and constraints, COVERT utilizes SOLVE to solve algebraically for the output variable. The output variable list of each group body indicates which of the LEL entries are to recieve flags of 3. Although SOLVE is not generally employed for ER's, the flagging of the output variables on the LEL is performed.

An exception to the exclusion of ER's from algebraic rearrangement is the declaration of an ER to be an evaluated ER. Equations, ER's and constraints become evaluated entities when the method code is greater than 5 and when an output variable is also a tear variable. COVERT makes a direct comparison of each output variable list encountered to the tear variable list for all groups having method codes exceeding 5. An evaluated entity receives an additional LEL entry with a flag of 4. This entry represents the value of the evaluated entity. COVERT requisitions a storage location for the value. The convention for including the new LEL entry is (1) place the new entry at the end of

the LEL and (2) add to the new end of the LEL a list entry representing the equal operator. To understand this procedure, we must first recall that the equations and constraints stored in SECEDE conform to the reverse polish notation with the "= 0" implied. Since the LEL's are developed from SECEDE, it follows that the LEL's will also be in reverse polish without the "= 0" actually appearing. Let us now consider the conversion of an equation

$$f(x,y,z)$$

(written in reverse polish notation) to an evaluated equation. Including the evaluation entry and the subtraction operator, the equation becomes

$$f(x,y,z)F =$$

or in standard notation

$$F = f(x,y,z)$$

COVERT is programmed to release an evaluation entry discovered on LEL prior to the output variable list/tear variable list comparison. the elimination of an evaluation entry is simplified by the use of reverse polish notation for the LEL and by the fact that evaluation entries are encountered only as the algebraic output of the LEL. Thus, an evaluation entry may always be deleted by removing from the LEL the last two list entries; the evaluation entry and the equality operator. The storage location assigned to the evaluation entry is placed on the available space for later reallocation.

For groups possessing method codes greater than 1, COVERT must install a method insert as the last accessible node on the group body. The end of the group body is indicated by a change in level following a call to NEXT. COVERT retains the address of the last node to effect the access to the last group body node once a level decrease is noted.

COUPLE attaches the method insert to the last group body entry. COVERT establishes the value storage lists and files discussed previously in sections VI.2 and VI.3.b. The tear variable list furnishes the data required for establishment of the storage facilities necessary for the tear variables. No such list being automatically available, COVERT develops the LEL-type list for evaluated entities as the evaluation entries and the value storage locations are created. Note that this list will not be created for method codes less than 6. The storage facilities of NEXT are employed to store the address of a partially completed list of evaluation entries whenever a level increase occurs - i.e. whenever a group is encountered on the group body of another group.

* VI.4.c. GLINT

GLINT uses NEXT to trace through either a single group or an entire GENDER list performing numerical evaluations with MATH and EXTRA. At the conclusion of each group possessing a method code exceeding unity, GLINT must call METHOD. If METHOD returns the address of the method insert, iteration on the current group body is necessary. Conversely, if METHOD returns zero for the address of the method insert, the convergence criterion has been satisfied and consideration of the next group may begin.

To facilitate the iteration on group bodies, GLINT uses a temporary list to store the address of the group. If an interior group is encountered, a new entry is pushed onto the temporary list. The trace of a group body is repeated by first incrementing the group trace key (t4f2) by unity. The trace is begun directly with the first group body entry rather than with the group. This procedure is essential to preserve the temporary lists upon which NEXT depends.

A program called NUMBER serves as an interface between GLINT and MATH. NUMBER directs MATH in the computation of an output variable value. NUMBER maintains a push-down list for the storage of operands (variables and partial results) in the fashion of a stack. NUMBER begins with the first LEL entry in computing an output value. Each operand encountered is stored on the push-down list until an operator is encountered. The encounter of an operator causes NUMBER to pop-up from the push-down list (PDL) a number of operands equal to the order of the operator (t2lfl). MATH performs the operation of the operator on the selected operands, and the result is placed on the PDL. The computation is complete when both the LEL and the PDL are simultaneously exhausted.

CHAPTER VII

USER MANUAL

In this chapter we shall not try to present all of instructions for using the GENDER System. Rather, we shall discuss briefly the essential instructions on a somewhat superficial level. References to text and tables in other chapters will furnish the details. The intent is the creation of a guide to the use of GENDER while repeating information already presented to the smallest extent possible.

Two principal areas of activity are required of the design engineer prior to employing GENDER: (1) preparation of input data and (2) program preparation. Each of these very general instructions contain a number of steps and are discussed in the sections to follow.

One instruction which fits in both the data preparation and the program preparation areas is the preparation of the information block. The data required to establish the information block may be input either via a READ statement or via a DATA statement.

VII.1. Input Data Preparation

The input data may be categorized as follows: (1) the core allocation, list and data field information required by COIN, (2) the information block, (3) SECEDE and (4) an initial GENDER list. Category 4 is optional and would be supplied when the continuation of a previous problem analysis is desired. The data in categories 1, 3 and 4 are input via the programs COIN, SECIAO and GENIAO respectively. The data formats for each of these programs are presented in the comment statements for the respective programs in Appendix A.

The data required to initialize the core allocation and list processing package read by COIN furnishes the means by which a design engineer may tailor GENDER to the machine available. If performed with caution and foresight, the preparation of the data for COIN may need be performed only once for a given machine environment. Tables 2, 3, 5, 6, 9 and 13 give the LEND's required for the various list types. The table entries indicated by a V represent variables which the engineer is permitted to choose. Perhaps we should review the definition of a LEND. For example let us consider a data field occupying bit positions 4 through 7 of the second and third words of a list entry. The four LEND components are, in order, the word index of the first field, the word index of the last field, the shift and the mask. For the data type we are considering, the LEND would be (2, 3, 2^3=8, 2^4-1=15). The aforementioned tables must be used in conjunction with Tables 10 and 14 for determining the data organization within the list and file entries. It is the data organization adopted by the engineer which will assign integer values to the V's in the LEND tables.

Table 21 lists all of the components to the information block. In general, numerical values can be assigned to each of these components only after the data for COIN has been prepared. Since the information block is to be established by the user provided main program, the format for the information block data is a matter of user preference.

VII.2. Program Preparation

Program preparation involves the coding of three programs by the design engineer. The programs METHOD and EXTRA are described in Chapter VI. Note that even if no ER's have been included in the problem, a program named EXTRA must still be provided to avoid an unresolved external reference error.

TABLE 21

Information Block

Component	Use	Component	Use
1	Length of a record segment	15	Highest possible dimensionality
2	Length of a floating point value	16	Ordinate segment size
3	Length of an alphameric name	17	Length of horizontal/vertical link pair
4	Length of a scalar variable file entry	18	Length of an element list entry
5	Length of a constant file entry	19	Length of an ordinate file entry
6	Number of segments requested	20	Length of a GALAP diverger
7	Unit scale factor for code names	21	Length of a GALAP node
8	Length of a diverger list entry	22	Reserved
9	Length of a group node	23	Reserved
10	Length of a group body node	24	Reserved
11	Length of an output list entry	25	Reserved
12	Length of a variable LEL entry	26	Number of old values to retain
13	Reserved	27	Maximum number of iterations
14	Length of insert for method	28	Convergence criterion

The third program which must be supplied by the user is a main program which must be supplied by the user is a main program. This program is the executive routine which will supervise the data input operations, problem analysis and solution, and the output of results.

The data input operations might be regarded as an initialization phase prior to attempting to solve the problem. In this phase, the program COIN is called, the information block is established and the program SECIAO is called. If the input of a GENDER list is desired, GENIAO is employed. Note the order of events for the initialization phase, for successful initialization is possible only in this order. Establishment of the information block involves the acquisition, via NEWCEL, of a list entry at least 22 words long. The data may be read directly into this list entry. Finally, the storage of the address of the list entry in JCCORE(17) completes the procedure.

The remainder of the main program is devoted to the analysis and output phases. The analysis strategy is limited to weighted output assignment and then minimum tear for the present. The future expansion of ALPAC (or perhaps additions to ALPAC by the design engineer) will permit alternate strategies. The completed solution procedure (i.e. - GENDER list may be output via GENIAO. SECEDE may also be output using SECIAO. To facilitate the resumption of analysis at a future date, punched output of both the GENDER list and SECEDE from GENIAO and SECIAO is available. If direct interpretation of the GENDER list is elected, either VARIAO or SECIAO may be employed to obtain the numerical results of the calculations. Note that the main program must be coded to react properly to any contingencies which may develop during interpretation. This topic of error recovery has already been discussed in Chapter VI.

In the event a user may wish to employ output facilities other than those afforded by GENIAO, SECIAO or VARIAO, JCCORE (13) must be set equal to 6 prior to the utilization of any GENDER program. JCCORE (13) serves as a printer carriage control character. The integer 6 indicates that a new line is to be begun for the next output. This internal carriage control does not supercede formated carriage control instructions, but applies only to GENDER - generated trace and error messages.

VII.3. Additional Algorithms

It was noted in the preceeding section that the algorithm selection currently available permits only a single analysis strategy. While it is planned to expand the algorithm library in the near future, it is unlikely that it will be possible to satisfy the needs of all potential users. Thus, it is likely that a design engineer will wish to program an analysis algorithm himself.

Before attempting the expansion of the algorithm library, certain of the * marked sections must be read. In particular, the purpose of an analysis algorithm is to modify a GENDER list. Consequently, the * marked section pertaining to the GENDER list must be understood. If the algorithm to be implemented is to employ a sparse incidence matrix, and particularly if the matrix operations are not presently provided in SIMPAC, the implementation details or SIMPAC must be read. The * marked sections for all data structure manipulation packages must be regarded similarly to SIMPAC whenever the manipulations to be imposed on the data structure are not the standard manipulations provided in GENDER.

VII.4. GENDER as a Design Tool

The design of a chemical processing plant is a rather complex procedure. Let us consider the impact of GENDER on the single design phase concerned with the differentiation of processing alternatives.

In response to a proposed raw materials/products set, the design engineer will develop a number of possible processing schemes. By omitting processing elements such as pumps and heat exchangers a simplified flowsheet can be created. The simplified flowsheet should retain only those units essential to the intent of the processing scheme. From the simplified flowsheets, mathematical models maybe developed employing short cut and simplification techniques wherever possible. The equation sets thus developed may be subjected to analysis and interpretation via GENDER. The design engineer has been relieved of the burden of solving the several modes either by hand or by programming the models for computer solution.

Frequently, the resolution of the crude models is insufficient to distinguish the best processing scheme from one or two other proposals. Consequently, it may be necessary to perform a run-off comparison between a subset of the original possible schemes. This second level comparison can afford improved resolution only if the accuracy (and the complexity) of the remaining models is increased. If the re-comparison involves only a few models, the development of models describing the processes to the greatest possible extent may be justified. Again, GENDER is to be employed for solution procedure generation and interpretation.

The GENDER list of the model selected as best should be retained for conversion (by hand) to a FORTRAN program. This program should be found to be quite useful as a comparison to actual plant performance. The model may even be of utility in establishing set joints for automatic control.

It is realized that the use of GENDER as a design tool is limited by the lack of an optimization supervisor to the GENDER system. A

program package performing constrained optimization would lend a higher degree of credibility to the model comparisons, particularly for the higher resolution models. Futher, it would improve the steady state performance predictions required to select optimally controller set points.

CHAPTER VIII

EXAMPLES

In this chapter we shall consider the application of GENDER to two example problems; equilibrium flash and a simple binary distillation. The equation set describing the behavior of a binary equilibrium flash operation is not dissimilar to the equation set pertaining to a distillation equilibrium stage. Consequently, it is possible to propose a single equation set sufficing, with minor modifications, for both examples.

We shall adopt the convention that the tray numbers of a distillation column increase from bottom to top. The letter I will be employed to represent the tray number. Figure 13 illustrates the general distillation tray, for which the following equations have been prepared.

(1) $K_{I,1} X_{I,1} - Y_{I,1} = 0$

(2) $K_{I,2} X_{I,2} - Y_{I,2} = 0$

(3) $\left[\exp\left(\frac{-0.120}{T}(32000) - 2.30(7.6)\right)\right]/P - K_{I,1} = 0$

(4) $\left[\exp\left(\frac{-0.120}{T}(39000) + 2.30(8.3)\right)\right]/P - K_{I,2} = 0$

(5) $X_{I,1}L_I + Y_{I,1}V_I - Z_{I,1}F_I - X_{I+1,1}L_{I+1} - Y_{I-1,1}V_{I-1} = 0$

(6) $X_{I,2}L_I + Y_{I,2}V_I - Z_{I,2}F_I - X_{I+1} - Y_{I-1,2}V_{I-1} = 0$

(7) $L_I + V_I - F_I - L_{I+1} - V_{I-1} = 0$

(8) $Z_{I,1} + Z_{I,2} - 1 = 0$

(9) $X_{I,1} + X_{I,2} - 1 = 0$

(10) $h_{L,I}L_I + H_{V,I}V_I - h_{F,I}F_I - h_{L,I+1}L_{I+1} - H_{V,I-1}V_{I-1} - Q = 0$

(11) $H_{V,I} - h_{L,I} - 2 = 0$

(12) $H_{V,I} - H^{\circ}_{V1}(Y_{I,1}) - H^{\circ}_{V2}(1 - Y_{I,1}) - (\alpha_1 Y_{I,1} + \alpha_2(1-Y_{I,1}))T + (\beta_1 Y_{I,1} + \beta_2(1-Y_{I,1}))T^2 = 0$

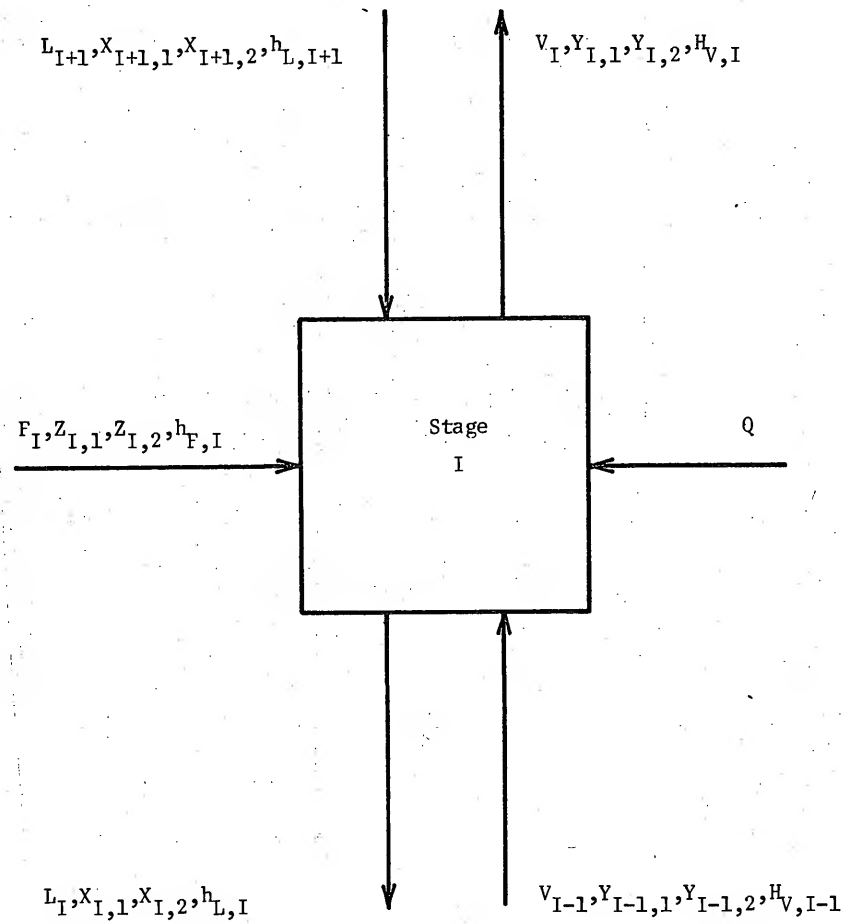

Figure 13 An Equilibrium Stage.

This equation set is adapted from the equations presented by Edie (1970) as an illustration of equilibrium flash. Two basic assumptions are implied by this equation set: (1) Raoult's law holds and (2) the pressure is constant throughout the column.

For both of the examples, the twelve equations will be regarded as a processing unit. The encoding of the equations for the creation of a unit in SECEDE requires the assignment of code names to each of the variables and constants. Table 22 presents the code name/variable assignments, while Table 23 provides the analagous information for constants.

To illustrate the preparation of the data required to establish the twelve equations describing a distillation tray as a unit in SECEDE, Figure 14 is provided. The formats for each of the input cards are given in the prefatory comment statements for the program SECIAO in Appendix A. With respect to the equation files, it must be recalled that the reverse Polish notation is employed. For example, equation 1 written in reverse Polish is

$$K_{I,1}X_{I,1} * Y_{I,1} - 0 =$$

Note that in the reverse Polish notation, all operators appear explicitly. GENDER has been programmed to "assure" the 0 = terms for equations in which an explicit equality is not declared. Thus, equation 1 is encoded for SECEDE as if it were simply

$$K_{I,1}X_{I,1} * Y_{I,1} -$$

In Figure 14, the cards terminating files contain messages indicating their specific function. This is in contradiction with the SECIAO requirement for concluding files with a blank card. The comments were included in Figure 14 for clarity and must not actually appear in the

TABLE 22

The Code Name/Variable Assignments for a Distillation Tray

Code Name	Variable	Code Name	Variable
1	F_I	16	$h_{L,I}$
2	$Z_{I,1}$	17	V_I
3	$Z_{I,2}$	18	$Y_{I,1}$
4	$h_{F,I}$	19	$Y_{I,2}$
5	L_{I+1}	20	$H_{V,I}$
6	$X_{I+1,1}$	21	$K_{I,1}$
7	$X_{I+1,2}$	22	$K_{I,2}$
8	$h_{L,I+1}$	23	T_I
9	V_{I-1}	24	P
10	$Y_{I-1,1}$	25	H°_{V1}
11	$Y_{I-1,2}$	26	H°_{V2}
12	$H_{V,I-1}$	27	$\alpha 1$
13	L_I	28	$\alpha 2$
14	$X_{I,1}$	29	$\beta 1$
15	$X_{I,2}$	30	$\beta 2$
		31	λ
		32	Q

TABLE 23

The Constant/Code Name Assignments for a Distillation Tray

Code Name	Constant
1	0.120
2	32000.
3	2.30
4	7.6
5	39000.
6	8.3
7	1.0
8	2.0

```
1                32                0                12                0
1                 1                0                 0
1                 2                0                 0
1                 3                0                 0
1                 4                0                 0
1                 5                0                 0
1                 6                0                 0
1                 7                0                 0
1                 8                0                 0
1                 9                0                 0
1                10                0                 0
1                11                0                 0
1                12                0                 0
1                13                0                 0
1                14                0                 0
1                15                0                 0
1                16                0                 0
1                17                0                 0
1                18                0                 0
1                19                0                 0
1                20                0                 0
1                21                0                 0
1                22                0                 0
1                23                0                 0
1                24                0                 0
1                25                0                 0
1               26                 0                 0
1                27                0                 0
1                28                0                 0
1                29                0                 0
1                30                0                 0
1                31                0                 0
1                32                0                 0
    END OF VARIABLE DECLARATION FILE / START OF ER DECLARATION FILE
    END OF ER DECLARATION FILE / START OF EQUATION DECLARATION FILE
1                 1                0
1                 2                0
1                 3                0
1                 4                0
1                 5                0
1                 6                0
1                 7                0
1                 8                0
1                 9                0
1                10                0
1                11                0
1                12                0
    END OF EQUATION DECLARATION FILE / START OF CONSTRAINT DECLARATION FILE
    END OF CONSTRAINT DECLARATION FILE / START OF CONSTANT VALUE FILE
1                 1                                  -0.120E+00
1                 2                                   +0.32E+04
1                 3                                   +0.23E+01
1                 4                                   +0.76E+01
1                 5                                   +0.39E+04
1                 6                                   +0.83E+01
1                 7                                    +0.1E+01
1                 8                                    +0.2E+01
    END OF CONSTANT VALUE FILE / START OF VARIABLE VALUE FILE
    END OF VARIABLE VALUE FILE / START OF ER NAME FILE
    END OF ER NAME FILE / START OF VARIABLE NAME FILE
1                 1                F
1                 2                Z1
```

Figure 14 SECIAO Data for a Distillation Tray

```
1                3                Z2
1                4                HF
1                5                LIN
1                6                XLIN1
1                7                XLIN2
1                8                HLIN
1                9                VIN
1               10                YVIN1
1               11                YVIN2
1               12                HVIN
1               13                L
1               14                X1
1               15                X2
1               16                HL
1               17                V
1               18                Y1
1               19                Y2
1               20                HV
1               21                K1
1               22                K2
1               23                T
1               24                P
1               25                HV10
1               26                HV20
1               27                ALPHA1
1               28                ALPHA2
1               29                BETA1
1               30                BETA2
1               31                LAMBDA
1               32                Q
   END OF VARIABLE NAME FILE / START OF ER FILE
   END OF ER FILE / START OF EQUATION FILE
1                1                1                0
1              211                5                0
1              151                1                0
1             2020                2
1              181                9                0
1             1020                2
   END OF EQUATION 1 / START OF EQUATION 2
1                2                1                0
1              221                5                0
1              151                1                0
1             2020                2
1              191                9                0
1             1020                2
   END OF EQUATION 2 / START OF EQUATION 3
1                3                1                0
1               12
1               22
1             2020                2
1              231                5                0
1             1030                2
1               32
1               42
1             2020                2
1             2010                2
1               90                1
1              241                9                0
1             1030                2
1              211                1                0
   END OF EQUATION 3 / START OF EQUATION 4
1                4                1                0
```

Figure 14 (Continued)

```
1                12
1                52
1              2020                2
1               231                5                0
1              1030                2
1                32
1                62
1              2020                2
1              2010                2
1                90                1
1               241                8                0
1              1030                2
1               221                1                0
1              1020                2
     END OF EQUATION 4 / START OF EQUATION 5
1                 5                1                0
1               141                1                0
1               131                5                0
1              2020                2
1               181                8                0
1               171                9                0
1              2020                2
1              2010                2
1                21                5                0
1                11                8                0
1              2020                2
1              1020                2
1                61                9                0
1                51                9                0
1              2020                2
1              1020                2
1               101                9                0
1                91                9                0
1              2020                2
1              1020                2
     END OF EQUATION 5 / START OF EQUATION 6
1                 6                1                0
1               151                1                0
1               131                5                0
1              2020                2
1               191                8                0
1               171                9                0
1              2020                2
1              2010                2
1                31                5                0
1                11                8                0
1              2020                2
1              1020                2
1                71                9                0
1                51                9                0
1              2020                2
1              1020                2
1               111                9                0
1                91                9                0
1              2020                2
1              1020                2
     END OF EQUATION 6 / START OF EQUATION 7
1                 7                1                0
1               131                1                0
1               171                9                0
1              2010                2
1                11                9                0
```

Figure 14 (Continued)

```
1                1020                  2
1                  51                  9                  0
1                1020                  2
1                  91                  9                  0
1                1020                  2
     END OF EQUATION 7 / START OF EQUATION 8
1                   8                  1                  0
1                  21                  5                  0
1                  31                  5                  0
1                2020                  2
1                  72
1                1020                  2
     END OF EQUATION 8 / START OF EQUATION 9
1                   9                  1                  0
1                  61                  5                  0
1                  71                  5                  0
1                2020                  2
1                  72
1                1020                  2
     END OF EQUATION 9 / START OF EQUATION 10
1                  10                  1                  0
1                 161                  1                  0
1                 131                  5                  0
1                2020                  2
1                 201                  1                  0
1                 171                  8                  0
1                2020                  2
1                2010                  2
1                  41                  1                  0
1                  11                  7                  0
1                2020                  2
1                1020                  2
1                  81                  9                  0
1                  51                  9                  0
1                2020                  2
1                1020                  2
1                 121                  9                  0
1                  91                  9                  0
1                2020                  2
1                1020                  2
1                 321                  5                  0
1                1020                  2
     END OF EQUATION 10 / START OF EQUATION 11
1                  11                  1                  0
1                 201                  5                  0
1                 161                  1                  0
1                1020                  2
1                 311                  9                  0
1                1020                  2
     END OF EQUATION 11 / START OF EQUATION 12
1                  12                  1                  0
1                 201                  1                  0
1                 251                  9                  0
1                 181                  8                  0
1                2020                  2
1                1020                  2
1                 261                  9                  0
1                  72
1                 181                  8                  0
1                1020                  2
1                2020                  2
1                1020                  2
```

Figure 14 (Continued)

```
1          271          9          0
1          181          8          0
1         2020          2
1          281          9          0
1           72
1          181          8          0
1         1020          2
1         2020          2
1         2010          2
1          231          1          0
1         2020          2
1         1020          2
1          291          9          0
1          181          8          0
1         2020          2
1          301          9          0
1           72
1          181          8          0
1         1020          2
1         2020          2
1         2010          2
1          231          1          0
1           82
1         1040          2
1         2020          2
1         2010          2
    END OF EQUATION 12
    END OF EQUATION FILE / START OF CONSTRAINT FILE
    END OF UNIT
```

Figure 14 (Continued)

data prepared for SECIAO.

VIII.1. A Simple Analysis Strategy

Both of the example problems share a common, simple analysis strategy. This strategy consists of the following steps:

1. Call HASSAL to perform output set assignment
2. Call SPEDUP to precedence order the acyclic equations and to collect the cyclic equations into groups
3. Call BEMOAN to select the tear variables for a cyclic group and to perform precedence ordering within a cyclic group
4. Repeat from step 2 if any cyclic groups remain unanalyzed

The analysis strategy is implemented in the main program prepared by the user.

In addition to the analysis strategy, the main program must also perform the administrative duties of preparing memory for allocation (via COIN), creating the information block and creating SECEDE (via SECIAO). The data required by COIN is contained in Figure 15. The LEND's selected for these example problems specify all data fields, except those for the seven part indices, to be whole words each. The example problems are sufficiently small to make memory conservation tactics unnecessary for the IBM-370/165. The whole word data items have the distinct advantage of rendering an allocatable core map (i.e. JCCORE) readable. The values of the information block components reflect the use of whole word data fields. These values are presented in Table 24.

VIII.2. Equilibrium Flash

An equilibrium flash is essentially a distillation tray for which the L_{I+1} and V_{I-1} are both zero. This situation is depicted in Figure 16. There are at least two acceptible ways of adapting the 12 equations

```
32          4000          32          8
 4
19             2
 1             0           1          0
 2             0           1          0
 1             1           1          0
 2             2           1          0
 2             2           1          0
99            99           1          0
 3             3           1          0
 4             4           1          0
 1             0   536870912          3
 1             0    16777216         31
 1             0     8388608          1
 1             0     4194304          1
 1             0       65536         63
 1             0       32768          1
 1             0           1      32767
 3             4           1          0
 1             1           1          0
 2             2           1          0
 5             5           1          0
22             2
 1             0           1          0
 2             0           1          0
 3             4           1          0
 5             7           1          0
 8            10           1          0
 1             0           1          0
11            14           1          0
15            18           1          0
 1             0   536870912          3
 1             0    16777216         31
 1             0     8388608          1
 1             0     4194304          1
 1             0       65536         63
 1             0       32768          1
 1             0           1      32767
11            11           1          0
12            12           1          0
13            13           1          0
14            14           1          0
15            15           1          0
 4             4           1          0
 5             5           1          0
14             2
 1             0           1          0
 2             0           1          0
 1             1           1          0
 3             3           1          0
 1             1           1          0
 2             2           1          0
 3             3           1          0
 4             4           1          0
 5             5           1          0
 6             6           1          0
 7             7           1          0
 8             8           1          0
 9             9           1          0
 2             2           1          0
 7             2
 1             0           1          0
 2             0           1          0
 3             3           1          0
 4             4           1          0
 5             5           1          0
 6             6           1          0
 7             0           1          0
```

Figure 15 Data for COIN

TABLE 24

Values for Information Block Components

Component	Value	Component	Value
1	32	15	0
2	1	16	32
3	2	17	2
4	5	18	6
5	3	19	9
6	1	20	3
7	100	21	12
8	3	22	0
9	18	23	0
10	15	24	0
11	4	25	0
12	5	26	0
13	0	27	100
14	14	28	0.001

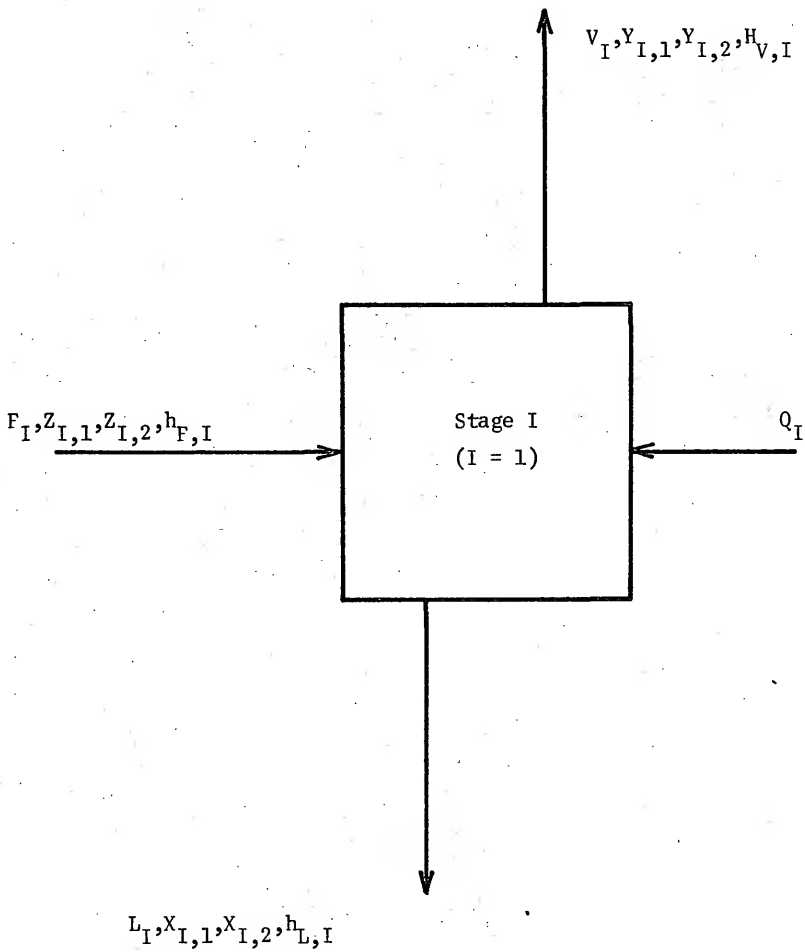

Figure 16 Equilibrium Flash.

describing a distillation tray to the equilibrium flash problem.

One method is suggested by the nature of the data required by SECIAO for establishing SECEDE. Each constituent of an equation (whether it be a variable, an operator or a constant) appears on a separate card. That is to say, an equation is constructed in SECEDE by supplying SECIAO with an ordered sequence of cards, each of which represents a component of the equation. Consequently, it is possible to remove cards to effect the elimination of the variables pertaining to the (I+1) and (I-1) stages. Care must be exercised in editing the equation, set by the simple expedient of removing cards, to avoid the introduction of syntac errors by failing to eliminate the variables associated with the eliminated variables.

Aside from the physical editing of the equation set, another alternative is available. If the variables pertaining to stages I+1 and I-1 could somehow be declared to be decision variables, and if the values of L_{I+1} and V_{I-1} could somehow be declared to be zero, then the transformation of the distillation tray equation set into the equilibrium flash equation set would be accomplished. If it is desired, the decision variable file and the variable value file supplied as data to SECIAO may be ammended to achieve the above declarations. This procedure, as was the case for the method of the preceding paragraph, entails the alteration of the SECIAO data.

If the alteration of the SECIAO data is to be avoided, while still accomplishing the task of declaring the variables L_{I+1} and V_{I-1} to be decisions valued at zero, recourse to a small amount of programming will be necessary. The program VARIAO must be employed to set the values for L_{I+1} and V_{I-1} at zero. This task may be installed in the main program at any point prior to interpretation (or any other use for variable

values, such as sensitivity analysis). It is suggested that VARIAO precede the use of LINKED.

The task of declaring the I+1 and I-1 stage variables to be decisions must be completed prior to initiation of equation set analysis. In other words, the declaration must precede the use of HASSAL. Although programs whose names begin with J are not normally recommended for direct user call, the program J2TADV may be employed to obtain decision variable list entries for the variables of stages I+1 and I-1. These entries, when linked together and installed on the group decision variable list, establish the variables of stages I+1 and I-1 as decision variables. The modification of the decision variable list is best accomplished immediately after geneation of the crude GENDER list by CUDGEL.

If J2TADV and VARIAO are to be employed to effect the tailoring of the distillation tray equations to equilibrium flash, the input data for SECEDE is shown in Figure 17. The symbol TRAY represents the inclusion of the unit data contained in Figure 14. The data required by VARIAO is presented in Figure 18. Should the reader desire to run this problem, using the data from Figures 14, 17 and 18, a comparison of the results obtained may be made with Appendix C. The machine listing presented in Appendix C was created from the GENIAO card output. Since this card output is consistent with the input format expected by GENIAO, the input format described in the prefatory comment cards for GENIAO (Appendix A) should be employed as the key for deciphering Appendix C.

The declaration of decision variables without revising the SECIAO data may be extended to other decision variables. For example, the elimination of the I+1 and I-1 stage variables leaves 16 variables in a

```
1                125                    0
1                126                    0
1                127                    0
1                128                    0
1                129                    0
1                130                    0
1                131                    0
1                132                    0
    END OF DECISION VARIABLE FILE / START OF TEAR VARIABLE FILE
    END OF TEAR VARIABLE FILE / START OF OPERATOR NAME FILE
1                                       UNARY
1                  1                    COMP
1                  2                    SIN
1                  3                    COS
1                  4                    TAN
1                  5                    ARCSIN
1                  6                    ARCCOS
1                  7                    ARCTAN
1                  8                    ALOG
1                  9                    EXP
1                 10                    RECIP
1                 11                    SQRT
1                 12                    SQUARE
1                                       BINARY
1                101                    =
1                102                    -
1                103                    /
1                104                    **
1                                       OTHER
1                201                    *N
1                202                    *N
    END OF OPERATOR NAME FILE / START OF METHOD NAME FILE
    END OF METHOD NAME FILE / START OF UNIT 0
    END OF UNIT 0 / START OF UNIT 1
TRAY
    END OF SECEDE
```

Figure 17 SECIAO Data for Equilibrium Flash

```
1                 105                    0                                  +0.0E+00
1                 109                    0                                  +0.0E+00
    END OF VARIABLE VALUE LIST
```

Figure 18 Data for VARIAO

problem involving only 12 equations. Obviously, four of the remaining variables must be decisions. The inclusion of the four decision variables on the decision variable list follows exactly the procedure for the I+1 and I-1 stage variables. However, unlike L_{I+1} and V_{I-1}, the values for the decision variables supplied to VARIAO via data will in general be unequal to zero.

Edie (1970) selected $Y_{I,1}$, P_I, V_I and F_I to be the four decision variables reducing the problem to a 12 variable/12 equation problem. Note that the user selection of decision variables is not necessary. HASSAL effects decision variable selection by regarding as decisions all variables remaining unassigned at the conclusion of the Hungarian algorithm. Weighting the variable/equation incidences can force HASSAL to select $Y_{I,1}$, P_I, V_I and F_I to be the decisions. In fact, the output assignments selected by Edie (1970) can be duplicated by HASSAL with the proper weights (based on the maximum product criteria).

In retrospect, the key feature presented in this section have been (1) the modification of a general problem formulation to a specific problem and (2) a procedure for the declaration of user selected decision variables, with particular emphasis on procedures for effecting the necessary alterations without recourse to revisions of the SECIAO data. Great importance has been placed on general problem formulations and maintaining the integrity of SECIAO data primarily because the preparation and modification of SECIAO data are tedious operations. A library of general formulations for common processing units prepared as SECIAO data should prove invaluable to the design engineer.

VIII.3. Binary Distillation

Figure 19 depicts a three tray distillation column employing a reboiler and a total condenser. The trays have been diagrammed separately

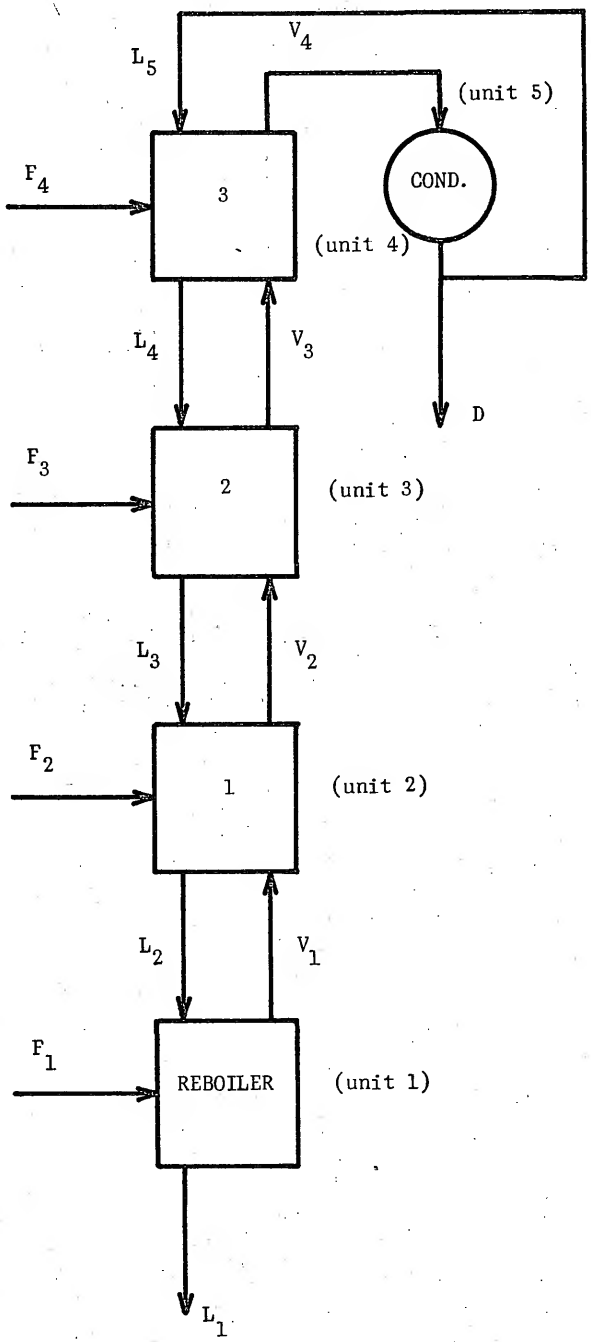

Figure 19 Simple Distillation Column.

giving the column a disjointed representation. This has been done to emphasize the designation of each tray as a separate unit. The primary purpose of this example is to illustrate the SECEDE provisions for assembling a collection of units into a processing scheme.

Units 1 through 4 are designated as the reboiler and the three trays respectively. Unit 5 represents the partitioning of the overhead vapor into a product and a recycle stream. The equations associated with total condensation are

(13) $X_{I,1} - Y_{I-1} = 0$

(14) $X_{I,1} + X_{I,2} - 1 = 0$

(15) $L_I + D - V_{I-1} = 0$

(16) $L_1/D - R = 0$

(17) $H_{V,I-1} V_{I-1} - H_{L,I} V_{I-1} - Q = 0$

(18) $H_{V,I-1} - H_{L,I} - 2 = 0$

The variables R and D represent the recycle ratio and the distillate product flow rate.

Two steps are required to declare commonality between variables of several units. First, a common variable value storage location must be reserved in unit 0. This is accomplished with the variable declaration file for unit 0. Second, the code name of the unit 0 variable must be installed in the declaration file entry for each of the unit variables for which commonality is to be established. The second step is accomplished via the variable declaration files for the affected units.

The use of the variable declaration files to provide the connections between processing units is illustrated by Figure 20. Figure 20 is the SECIAO data required to establish SECEDE for the three tray distillation column. The variable declaration file appears explicitly in Figure 20 for all units. The remaining files for units 1 through 4

```
1                2                    0
1                3                    0
1                4                    0
1                5                    0
1                6                    0
1                7                    0
1                8                    0
1                9                    0
1               10                    0
1               11                    0
1               12                    0
1               13                    0
1              109                    0
1              110                    0
1              111                    0
1              112                    0
    END OF DECISION VARIABLE FILE / START OF TEAR VARIABLE FILE
    END OF TEAR VARIABLE FILE / START OF OPERATOR NAME FILE
1                                   UNARY
1                1                  COMP
1                2                  SIN
1                3                  COS
1                4                  TAN
1                5                  ARCSIN
1                6                  ARCCOS
1                7                  ARCTAN
1                8                  ALOG
1                9                  EXP
1               10                  RECIP
1               11                  SQRT
1               12                  SQUARE
1                                   BINARY
1              101                  =
1              102                  -
1              103                  /
1              104                  **
1                                   OTHER
1              201                  +N
1              202                  *N
    END OF OPERATOR NAME FILE / START OF METHOD NAME FILE
    END OF METHOD NAME FILE / START OF UNIT 0
1               47
1                1                    0
1                2                    0
1                3                    0
1                4                    0
1                5                    0
1                6                    0
1                7                    0
1                8                    0
1                9                    0
1               10                    0
1               11                    0
1               12                    0
1               13                    0
1               14                    0
1               15                    0
1               16                    0
1               17                    0
1               18                    0
1               19                    0
1               20                    0
```

Figure 20 SECIAO Data for Simple Distillation Column.

```
1                21                0
1                22                0
1                23                0
1                24                0
1                25                0
1                26                0
1                27                0
1                28                0
1                29                0
1                30                0
1                31                0
1                32                0
1                33                0
1                34                0
1                35                0
1                36                0
1                37                0
1                38                0
1                39                0
1                40                0
1                41                0
1                42                0
1                43                0
1                44                0
1                45                0
1                46                0
1                47                0
    END OF COMMON VAR. DEC. FILE / START OF COMMON VAR. VALUE FILE
    END OF COMMON VAR. VALUE FILE / START OF COMMON VAR. NAME FILE
1                 1                P
1                 2                HV10
1                 3                HV20
1                 4                ALPHA1
1                 5                ALPHA2
1                 6                BETA1
1                 7                BETA2
1                 8                LAMBDA
1                 9                Q
1                10                F
1                11                Z1
1                12                Z2
1                13                HF
1                14                V1
1                15                Y11
1                16                Y12
1                17                HL1
1                18                L2
1                19                X21
1                20                X22
1                21                HL2
1                22                V2
1                23                Y21
1                24                Y22
1                25                HV2
1                26                L3
1                27                X31
1                28                X32
1                29                HL3
1                30                V3
1                31                Y31
1                32                Y32
1                33                HV3
```

Figure 20 (Continued)

```
1                34              L4
1                35              X41
1                36              X42
1                37              HL4
1                38              V4
1                39              Y41
1                40              Y42
1                41              HV4
1                42              L5
1                43              X51
1                44              X52
1                45              HL5
1                46              K1
1                47              K2
    END OF UNIT 0 / START OF UNIT 1
1                32                0              12              0
1                 1               10               0
1                 2               11               0
1                 3               12               0
1                 4               13               0
1                 5               18               0
1                 6               19               0
1                 7               20               0
1                 8               21               0
1                 9                0               0
1                10                0               0
1                11                0               0
1                12                0               0
1                13                0               0
1                14                0               0
1                15                0               0
1                16                0               0
1                17               14               0
1                18               15               0
1                19               16               0
1                20               17               0
1                21               46               0
1                22               47               0
1                23                0               0
1                24                1               0
1                25                2               0
1                26                3               0
1                27                4               0
1                28                5               0
1                29                6               0
1                30                7               0
1                31                8               0
1                32                0               0
    END OF VARIABLE DECLARATION FILE
TRAY
    END OF UNIT 1 / START OF UNIT 2
1                32                0              12              0
1                 1               10               0
1                 2               11               0
1                 3               12               0
1                 4               13               0
1                 5               26               0
1                 6               27               0
1                 7               28               0
1                 8               29               0
1                 9               14               0
1                10               15               0
```

Figure 20 (Continued)

```
1                11                16                0
1                12                17                0
1                13                18                0
1                14                19                0
1                15                20                0
1                16                21                0
1                17                22                0
1                18                23                0
1                19                24                0
1                20                25                0
1                21                46                0
1                22                47                0
1                23                 0                0
1                24                 1                0
1                25                 2                0
1                26                 3                0
1                27                 4                0
1                28                 5                0
1                29                 6                0
1                30                 7                0
1                31                 8                0
1                32                 9                0
    END OF VARIABLE DECLARATION FILE
TRAY
    END OF UNIT 2 / START OF UNIT 3
1                32                 0               12                0
1                 1                 0                0
1                 2                 0                0
1                 3                 0                0
1                 4                 0                0
1                 5                34                0
1                 6                35                0
1                 7                36                0
1                 8                37                0
1                 9                22                0
1                10                23                0
1                11                24                0
1                12                25                0
1                13                26                0
1                14                27                0
1                15                28                0
1                16                29                0
1                17                30                0
1                18                31                0
1                19                32                0
1                20                33                0
1                21                46                0
1                22                47                0
1                23                 0                0
1                24                 1                0
1                25                 2                0
1                26                 3                0
1                27                 4                0
1                28                 5                0
1                29                 6                0
1                30                 7                0
1                31                 8                0
1                32                 9                0
    END OF VARIABLE DECLARATION FILE
TRAY
    END OF UNIT 3 / START OF UNIT 4
1                32                 0               12                0
```

Figure 20 (Continued)

```
1                1                10               0
1                2                11               0
1                3                12               0
1                4                13               0
1                5                40               0
1                6                41               0
1                7                42               0
1                8                43               0
1                9                30               0
1               10                31               0
1               11                32               0
1               12                33               0
1               13                34               0
1               14                35               0
1               15                36               0
1               16                37               0
1               17                38               0
1               18                39               0
1               19                40               0
1               20                41               0
1               21                46               0
1               22                47               0
1               23                 0               0
1               24                 1               0
1               25                 2               0
1               26                 3               0
1               27                 4               0
1               28                 5               0
1               29                 6               0
1               30                 7               0
1               31                 8               0
1               32                 9               0
    END OF VARIABLE DECLARATION FILE
TRAY
    END OF UNIT 4 / START OF UNIT 5
1               11                 0               7               0
1                1                38               0
1                2                39               0
1                3                41               0
1                4                42               0
1                5                43               0
1                6                44               0
1                7                45               0
1                8                 0               0
1                9                 0               0
1               10                 8               0
1               11                 0               0
    END OF VARIABLE DECLARATION FILE / START OF ER DECLARATION FILE
    END OF ER DECLARATION FILE / START OF EQUATION DECLARATION FILE
1                1                 0
1                2                 0
1                3                 0
1                4                 0
1                5                 0
1                6                 0
1                7                 0
    END OF EQUATION DECLARATION FILE / START OF CONSTRAINT DECLARATION FILE
    END OF CONSTRAINT DECLARATION FILE / START OF CONSTANT VALUE FILE
1                1                                 +0.1E+01
    END OF CONSTANT VALUE FILE / START OF VARIABLE VALUE FILE
    END OF VARIABLE VALUE FILE / START OF ER NAME FILE
    END OF ER NAME FILE / START OF VARIABLE NAME FILE
```

Figure 20 (Continued)

```
1                1                VIN
1                2                YVIN1
1                3                HVIN
1                4                L
1                5                X1
1                6                X2
1                7                HL
1                8                O
1                9                R
1               10                LAMBDA
1               11                Q
    END OF VARIABLE NAME FILE / START OF ER FILE
    END OF ER FILE / START OF EQUATION FILE
1                1                1                0
1               51                1                0
1               21                9                0
1             1020                2
    END OF EQUATION 1 / START OF EQUATION 2
1                2                1                0
1               51                9                0
1               61                1                0
1             2010                2
1               12
    END OF EQUATION 2 / START OF EQUATION 3
1                3                1                0
1               41                5                0
1               81                5                0
1             2010                2
1               11                9                0
1             1020                2
    END OF EQUATION 3 / START OF EQUATION 4
1                4                1                0
1               41                5                0
1               81                5                0
1             1030                2
1               91                9                0
1             1020                2
    END OF EQUATION 4 / START OF EQUATION 5
1                5                1                0
1               71                9                0
1               31                9                0
1             1020                2
1              101                1                0
1             1020                2
    END OF EQUATION 5 / START OF EQUATION 6
1                6                1                0
1               31                9                0
1               71                1                0
1             1020                2
1              111                1                0
1              101                9                0
1             1020                2
    END OF EQUATION 6
    END OF EQUATION FILE / START OF CONSTRAINT FILE
    END OF UNIT 5
    END OF SECEOE
```

Figure 20 (Continued)

are indicated by the symbol TRAY. These files for units 1 through 4 are identical to the files presented in Figure 14. Note that unit represents the reboiler, for which the V_{I-1}, Y_{I-1} and H_{I-1} variables are meaningless. These variables should be eliminated from consideration prior to the algorithmic analysis phase according to the instructions of the proceeding section.

As a specific example of commonality, let us consider the variable L_2. This particular variable appears in both units 1 and 2. Let us assign the code name 1 to the unit 0 variable we shall employ for the variable L_2. An inspection of the variable declaration file for unit 1 reveals a common variable declaration of 1 for the variable L_{I+1}. Similarly, the variable L_I for unit 2 also carries 1 as the common variable declaration. Consequently, L_{I+1} for I = 1 and L_I for I = 2 share a common value location (with the code name 1) in unit 0.

Should the reader wish to run this problem, the fully analyzed GENDER list may be compared with the machine listing in Appendix D. The SECIAO data for the distillation problem may be extracted from Figures 14 and 20. It is recommended that a single copy of the Figure 14 data be prepared and tested with SECIAO. Once the bugs, if any, have been removed, this card deck should be employed as the master for generating the three additional copies required on a duplicating card punch. When collated with the data prepared from Figure 20, the unit data decks will be integrated into a processing problem. Appendix D, following the example of Appendix C, is the machine listing generated from the GENIAO card output.

CHAPTER IX

CONCLUSION

The GENDER System has been developed to assist the design engineer with large scale design problems. This initial version of GENDER, however, is not completely suitable for a production system. Modifications and additions to GENDER have already been planned (1) to improve convenience aspects and (2) to include additional analysis algorithms.

IX.1. Convenience Aspects

Let us consider the contact of the design engineer with GENDER. To employ GENDER, an engineer must encode the equation set in the format expected by SECIAO. Further, the output of SECEDE via SECIAO produces code. Similar considerations apply to GENIAO and the GENDER list. The coded input/output is a usable medium for communication with GENDER, but is not a convenient medium. GENDER would be simpler to use, if the input/output language was already familiar to the design engineer. The adaptation of algebraic equations for the input/output medium is contemplated, perhaps ultimately resembling the input/output language developed by Soylemez (1971).

Another aspect of the engineer/GENDER interface involves the user prepared main program. In the current version of GENDER, a main program directs the solution procedure development. This program must be devised to properly react to error conditions recognized as a consequence of analysis or perhaps even interpretation. It is most probable that many of the preconceived error conditions would never be realized for a particular design problem. Thus, the effort of developing a complex error

recovery mechanism is somewhat wasted. So far we have been describing the utilization of GENDER in the batch mode. Let us now consider a real time interactive environment. The responsibility for creating the main program would be relieved from the user entirely. Instead the user would employ a console to direct GENDER in performing an analysis strategy. The user need not consider alternative strategies until an error condition has been detected. If a recovery procedure cannot be immediately conceived, the user may temporarily leave the system after employing GENIAO and SECIAO to output the GENDER list and SECEDE. Analysis may be resumed whenever a corrective course of action has been charted. Further, the interactive environment facilitates monitoring either the development or the interpretation of a solution procedure at the discretion of the user.

Optimization is an essential factor in design. GENDER has been developed to permit convenient constrained optimization. For the present, the search for optimality has been left to the user. The development of an optimization supervisor to the GENDER System has been proposed. The theoretical foundation for the optimization program has already been established by DeBrosse (1971). The existence of this program should relieve the design engineer of a considerable burden.

IX.2. Additional Algorithms

In Chapter VII, we noted that the algorithms provided with GENDER allow only a single analysis strategy. Albeit a realistic strategy, one cannot expect it to suffice in all instances. Several algorithms have been proposed for inclusion in the analysis repertoire.

In section V.1. it was pointed out that the weights employed by HASSAL in performing output set assignment could be derrived from a sensitivity analysis. The resulting output set assignment would thus be rapidly

convergent (Edie and Westerberg, 1971).

The use of HASSAL to obtain output set assignments enhancing the convergence of a solution procedure is expected to be relatively simple to implement. However, there is an inherent disadvantage. HASSAL makes no attempt to differentiate between the convergence properties of every output set assignment. An algorithm developed by Edie (1970), based on the creation of a dynamic programming network, selects the best output set from the standpoint of convergence and singularity of the resulting Jacobian. Once implemented, this algorithm should be regarded as something of a last resort since the extensive network manipulations are expected to entail high execution times.

The influence of the output set assignment extends beyond the convergence properties of a solution procedure. For a particular design problem, the output set determines the minimum number of tear variables. The criterion for output set assignment may be arbitrary weights, user preference or sensitivities. Let us consider minimum tear as the criterion for output set selection. With this criterion, the output set selected would be the set for which the minimum number of tear variables would be the minimum over all possible output sets. The adaptation of this criterion to an algorithm performing both output set and minimum tear selections has been accomplished by Christensen (1970). As in all cases where an improvement in the results of an optimization is expected, the Christensen algorithm is more complex than BEMOAN. In fact, it is vastly more complex. Consequently, a user must cautiously weigh the advantages of perhaps reducing the number of tear variables against the increased execution time over BEMOAN.

The algorithms discussed in the proceeding paragraphs already exist in the literature, there remaining only the adaptation to GENDER. Another

algorithm currently being devised focuses on the compaction of GENDER lists via indexing considerations. These considerations parallel the simplification of FORTRAN programs by employing DO statements. As one might suspect, pattern recognition plays a key role in the identification of indexed GENDER list components in a repeating sequence. Considerable simplification to the GENDER list and to the SIM's is anticipated for all staged and discretized unit processes, such as distillation and absorption.

IX.3. A Final Note on GENDER

The current version of GENDER is heavily diagnostic. That is, many more tests, with provisions for diagnostic messages, are performed than are essential to the task of solution procedure manipulation. These tests must be removed before GENDER can begin to assume the configuration of a production system. The tests were originally devised as debugging conveniences. However, by retaining the myriad tests, GENDER affords the research engineer with a convenient medium for developing and testing analysis algorithms, and for studying the influence of analysis strategies on solution procedures.

APPENDICES

APPENDIX A

```
C
C*****************************************************************
C                          **************
C                          *   GENDER     *
C                          *   GENERAL    *
C                          * INFORMATION  *
C                          **************
C
C       SPECIAL INSTRUCTIONS/JOB STEPS
C           A USER PREPARED MAIN PROGRAM MUST
C           1. CALL COIN
C           2. ESTABLISH INFORMATION BLOCK
C           3. CALL SECIAO
C           4. CALL GENIAO (OPTIONAL)
C           5. EXECUTE ANALYSIS STRATEGY
C           6. OUTPUT RESULTS
C
C       SPECIAL INSTRUCTIONS/DEFAULT OPTIONS
C           THE PROGRAM DFAULT IS PROVIDED TO CREATE THE DATA
C           STRUCTURES REQUIRED FOR ANALYSIS AUTOMATICALLY.  THE USER
C           MAY, IF DESIRED, OMIT SIM GENERATION, CRUDE GENDER LIST
C           GENERATION AND OTHER PROCEDURES FROM THE USER PREPARED
C           MAIN PROGRAM.
C
C       CURRENT STATUS
C           OPERATIONAL, EXCEPT THE INTERPRETER.  THE INTERPRETER
C           IS CURRENTLY IN THE CODE/DEBUG STAGE.
C
C*****************************************************************
C
```

```
C
C*****************************************************************
C                              *********
C                              * DFAULT *
C                              *********
C       PURPOSE
C           DFAULT PROVIDES DEFAULT OPTIONS FOR THE GENDER SYSTEM.
C
C       USAGE
C           CALL DFAULT(JCGRUP,JCASIM,JCMODE,JCCLNO)
C
C       DATA FORMAT
C           N/A
C
C       DESCRIPTION OF PARAMETERS
C           JCGRUP = THE ADDRESS OF A GENDER GROUP
C           JCASIM = THE ADDRESS OF A SIM INFORMATION BLOCK
C           JCMODE = AN INTEGER HAVING THE FOLLOWING VALUES AND MEANING
C                    0 = ALL OPTIONS
C                    1 = PROVIDE GROUP IF JCGRUP = 0
C                    2 = OPTION 1 AND PROVIDE SIM IF JCASIM = 0
C                    3 = OPTIONS 1 THROUGH 2 AND PROVIDE OUTPUT SET
C                        IF THE SIM HAS AT LEAST ONE UNASSIGNED ROW
C           JCCLNO = A USER ASSIGNED CALL STATEMENT PARAMETER
C
C       REMARKS
C           THE INTEGER 1 IN THE NAMED COMMON STATEMENT MUST, FOR
C           SOME MACHINES, BE REPLACE BY THE VALUE OF JCTCIA.
C           MORE DEFAULT OPTIONS WILL BE ADDED AS THEY ARE DEVELOPED.
C
C       SUBROUTINE AND FUNCTION SUBPORGRAMS REQUIRED
C           CUDGEL
C           FIND
C           FRMCEL
C           HASSAL
C           J70SUP
C           SIMGEN
C           SYSENT
C           SYSEXT
C           TOCELL
C
C       ERROR CODES FOR THIS PROGRAM
C           NONE
C
C       METHOD
C           SELF-EXPLANATORY
C
C                              *********
C
      SUBROUTINE DFAULT(JCGRUP,JCASIM,JCMODE,JCCLNO)
      COMMON/ALLOC/JCCORE(1)
      DIMENSION S(4),R(2),D(1),O(4)
      INTEGER S,R,D,O
      DATA S/1,8,1,1/,R/1,0/,O/0,1,13,1,1/
C                 ENTER
      CALL SYSENT(1,6,1,1,JCCLNO)
C                 MA=ADDRESS OF SPUR FOR GENDER GROUP
      M1=JCCORE(16)+1
      M2=JCCORE(17)+8
      MA=JCCORE(M1)+5*JCCORE(M2)-5
C                 CRUDE GENDER LIST
      IF(JCGRUP.EQ.0)CALL CUDGEL(0,JCGRUP,1)
```

```
      IF(JCMODE.EQ.1)GO TO 99
C                   SIM
      CALL FRMCEL(JCCORE(MA),JCGRUP,JCASIM,S,1)
      IF(JCASIM.EQ.0)CALL SIMGEN(JCGRUP,JCASIM,1,1)
      CALL TOCELL(JCCORE(MA),JCGRUP,JCASIM,S,1)
      IF(JCMODE.EQ.2)GO TO 99
C                   MB=ADDRESS OF SPUR FOR ORDINATE
      M1=JCASIM+2
      MB=JCCORE(M1)-5
C                   MC=ADDRESS OF LOAD FOR ROW
      MC=J7OSUP(JCASIM,1,0,0,1)
      MD=MC+7
C                   OUTPUT SET ASSIGNMENT
      R(2)=1-JCCORE(MD)
      M1=1
      O(1)=0
      CALL FIND(JCCORE(MB),R,D,O,R,JCCORE(MC),M1,1)
      IF(M1.NE.0)CALL HASSAL(JCGRUP,JCASIM,1)
      IF(JCMODE.EQ.3)GO TO 99
C                   MORE DEFAULT OPTIONS TO BE ADDED LATER
C                   EXIT
   99 CALL SYSEXT
      RETURN
      END
```

```
C
C*****************************************************************
C                          ***************
C                          *   USRPAC    *
C                          *   GENERAL   *
C                          * INFORMATION *
C                          ***************
C
C       SPECIAL INSTRUCTIONS/DATA TYPES
C           LIST TYPES 1 (SECEDE) AND 2 (GENDER LIST) MUST BE FORWARD-
C           BACKWARD.
C
C       SPECIAL INSTRUCTIONS/LEND
C           THE FOLLOWING DATA STRUCTURES MUST BE PERMITTED BY THE
C           LENDS.
C           SECEDE - LIST TYPE 1
C               SEE LEND REQUIREMENTS FOR REMOTE.
C               (1, 5 - TYPE FLAG, 7 - SELECTION COST, 8 - CODE NAME,
C                   19 - DIMENSIONALITY)
C               (1, 5- TYPE FLAG, 16 - CODE NAME, 16 - ORDER OF
C                   OPERATOR)
C           GENDER LIST - LIST TYPE 2
C               SEE LEND REQUIREMENTS FOR NETPAC (IN SUPPAC).
C               DATA TYPE 6 MUST HAVE A LEND OF (1,0,1,0).
C               (1, 2, 3, 3 - TYPE FLAG, 4, 4, 4 - WORKING SPACE, 5, 5,
C                   5, 7 - POINTER, 7 - POINTER, 7 - POINTER, 7 -
C                   DEPTH, 8 - POINTER, 8 - POINTER, 8 - POINTER, 8 -
C                   METHOD CODE)
C               (1, 2, 3, 3 - TYPE FLAG, 4, 4, 4 - WORKING SPACE, 5, 5,
C                   5, 16 - POINTER, 17 - POINTER, 18 - POINTER, 19 -
C                   DIMENSIONALITY, 20 - NUMBER OF OUTPUTS)
C               (1, 2, 3, 21 - CODE NAME, 22 - DIMENSIONALITY)
C               (1, 3, 23 - CONSTANT VALUE)
C
C       SPECIAL INSTRUCTIONS/DEBUG
C           THE FOLLOWING TABLE GIVES THE PACKAGE AND PROGRAM ID.
C           NUMBERS FOR USRPAC.
C           PROGRAM          PKG. ID./PROG. ID.
C           IOPAC
C           GENIAO           1/1
C           SECIAO           1/2
C           VARIAO           1/3
C           J1AR8S           1/4
C           J1CONS           1/5
C           J1DECL           1/6
C           J1DIMS           1/7
C           J1FILE           1/8
C           J1GMBL           1/9
C           J1LEOL           1/10
C           J1NAME           1/11
C           J1RAFE           1/12
C           J1RESO           1/13
C           J1VALU           1/14
C           J1VARI           1/15
C           J1VTVF           1/16
C           INTERP
C           COVERT           2/1
C           GLINT            2/2
C           LINKED           2/3
C           NUMBER           2/4
C           J1FATV           2/5
C
```

```
C         CURRENT STATUS
C             IOPAC - OPERATIONAL
C             INTERP - CODE/DEBUG
C
C**************************************************************************
C
```

```
C
C*****************************************************************
C                              **********
C                              * GENIAO *
C                              **********
C
C       PURPOSE
C           GENIAO HANDLES THE INPUT AND OUTPUT OF THE DATA DESCRIBING
C           A GENDER LIST.
C
C       USAGE
C           CALL GENIAO(JCSENS,JCCLNO)
C
C       DATA FORMAT
C           THE SYMBOLS  , + AND $ ARE USED TO DELIMIT LOOPS.  B
C           INDICATES A BLANK CARD.  THE SYMBOLS *N* DO NOT APPEAR ON
C           THE CARDS READ, BUT ARE USED HERE TO INDICATE A CARD TYPE
C           NUMBER, N, WHICH CORRESPONDS TO THE FOLLOWING DATA FORMATS.
C               CARD TYPE     FORMAT
C               1             I1,I19,3I20
C               2             I5,I20,2I5,I20,I5,I20
C               3             I1,I19,2I20
C               4             I1,I19,I20
C               5             4I20
C
C           +(LEVEL 0)
C           +(GROUPS)
C           +  1, IDENTIFICATION FLAG (=0), PROTECTION FLAG, DEPTH,
C                 METHOD CODE *1*
C           ++
C           +++ INDICES *2* (MIN. AND MAX.)
C           ++$
C           ++(DECISION VARIABLE LIST)
C           ++
C           +++ 1, VARIABLE CODE NAME, DIMENSIONALITY, SELECTION FLAG
C                  *3*
C           +++
C           ++++ INDICES *2* (MIN. AND MAX.)
C           ++$$
C           ++B
C           ++(TEAR VARIABLE LIST)
C           ++
C           +++ 1, VARIABLE CODE NAME, DIMENSIONALITY, SELECTION FLAG
C                  *3*
C           +++
C           ++++ INDICES *2* (MIN. AND MAX.)
C           ++$$
C           ++B
C           +$
C           +B
C           +(LINKAGE)
C           +
C           ++ 1, NUMBER OF PATHS OUT OF A NODE (FORWARD), SEQUENCE
C                  NUMBER OF LEADING NODE *4*
C           ++
C           +++ SEQUENCE NUMBERS OF TRAILING NODES *5*
C           +$$
C           +B
C           +(LEVEL 1 - REPEATED FOR ADDITIONAL LEVELS)
C           +(GROUP BODY ENTRIES - MAY BE A GROUP AS IN LEVEL 0)
C           +
C           ++ 1, IDENTIFICATION FLAG (1 ER, 2 EVALUATED ER, 3
```

```
C                EQUATION, 4 SOLVED EQUATION, 5 EVALUATED EQUATION, 6
C                CONSTRAINT, 7 SOLVED CONSTRAINT, 8 EVALUATED
C                CONSTRAINT), CODE NAME, DIMESNIONALITY, NUMBER OF
C                OUTPUTS *3* -OR- 1, IDENTIFICATION FLAG (9 METHOD
C                INSERT), NUMBER OF ENTRIES *4*
C              ++
C              +++ INDICES (OMIT FOR ID. FLAG 9) *2* (MIN. AND MAX.)
C              ++$
C              ++(OUTPUT VARIABLE LIST - OMIT FOR ID. FLAG 9)
C              ++
C              +++ 1, VARIABLE CODE NAME, DIMENSIONALITY, SELECTION FLAG
C                   *3*
C              +++
C              ++++ INDICES *2*
C              ++$$
C              ++B
C              ++(LINK EDITOR LIST - OMIT FOR ID. FLAG 9)
C              ++
C              +++ 1, IDENTIFICATION FLAG (0 OPERATOR, 1 VARIABLE, 2
C                     CONSTANT, 3 OUTPUT VARIABLE, 4 EQUATION VALUE),
C                     CODE NAME, DIMENSIONALITY (ORDER OF OPERATOR) *1*
C              +++
C              ++++ INDICES *2*
C              ++$$
C              ++B
C              +B
C              +(LINKAGE)
C              +
C              ++ 1, NUMBER OF PATHS OUT OF A NODE (FORWARD), SEQUENCE
C                    NUMBER OF LEADING NODE *4*
C              ++
C              +++ SEQUENCE NUMBERS OF TRAILING NODES *5*
C              +$$
C              +B
C              $
C              B
C              B
C
C         DESCRIPTION OF PARAMETERS
C             SEE SECIAO
C
C         REMARKS
C             THE NUMBER REPRESENTED BY JCTCIA MUST, FOR SOME MACHINES,
C             REPLACE THE 1 IN THE NAMED COMMON STATEMENT.
C             NOTE THAT TWO BLANK CARDS IN ADDITION TO THE BLANK
C             TERMINATING THE LAST LINKAGE INPUT ARE REQUIRED TO
C             TERMINATE GENDER INPUT.
C
C         SUBROUTINE AND FUNCTION SUBPROGRAMS REQUIRED
C             CORN
C             COUPLE
C             FETCH
C             FRMCEL
C             J1DIMS
C             J1FILE
C             J1GMBL
C             J1LEDL
C             J1VARI
C             J1VTVF
C             LNKBNT
C             LNKFWD
C             NEWCEL
```

```
C           NEXT
C           POPUP
C           PUSH
C           STORE
C           SYSENT
C           SYSERR
C           SYSEXT
C           TOCELL
C
C       ERROR CODES FOR THIS PROGRAM
C           CODE FATAL  ERROR(DATA PROVIDED)
C           1    NF     NO GIM8L (JCCORE(16))
C           2    NF     NO GENDER LIST (JCCORE(JCCORE(16)))
C           3    NF     NO ENTRIES FOR LEVEL (LEVEL)
C
C       METHOD
C           SELF-EXPLANATORY
C
C                              **********
C
      SUBROUTINE GENIAO(JCSENS,JCCLNO)
      COMMON/ALLOC/JCCORE(1)
      DIMENSION S(5),C(5),D(8),L(5),F(2),Z(30),G(13),B(19),U(10),M(4),
     1 K(7),P(4),V(7),H(4),E(4),N(4),Y(4),A(4),O(4),X(4),T(30),W(7)
      INTEGER S,C,D,F,Z,G,B,U,P,V,H,E,Y,A,Q,X,T,SAVE,W
      DATA S/0,0,1,0,2/,C/0,0,1,0,1/,D(1)/1/,L/3,1,1,1,0/,F/1,2/,Z(15)
     1 /-1/,Z(29)/0/,Z(30)/18/,G/4,3,1,2,7,4,4,8,4,4,7,2,3/,B/6,3,1,2,
     2 17,1,1,19,1,1,20,1,1,18,1,1,16,1,1/,U/3,3,1,2,17,1,1,16,1,1/,M/1,
     3 6,0,0/,K/2,21,1,1,22,1,1/,P/1,1,1,1/,V/2,5,2,2,2,1,1/,H/1,7,1,1/,
     4 E/1,1,1,2/,N/1,4,3,3/,Y/1,1,2,2/,A/1,21,1,1/,Q/1,3,1,1/,
     5 X/1,4,1,1/,T(15)/-1/,T(29)/0/,T(30)/18/,W/2,5,1,1,1,1,1/
C                  ENTER
      CALL SYSENT(1,5,1,1,JCCLNO)
      SAVE=JCCORE(18)
C                  IN OR OUT
      IF(JCSENS.LT.1)GO TO 1
C                  SET UP GIM8L
      CALL J1GM8L(MA,MB,MC,1)
      GO TO 2
C                  MA=ADDRESS OF GIM8L
    1 MA=JCCORE(16)
      M1=MA
      M2=1
      IF(MA.LE.0)GO TO 41
C                  M8=ADDRESS OF SPUR BLOCK
      M1=MA+1
      MB=JCCORE(M1)
C                  MC=MAXIMUM CORE BLOCK SERVICIBLE WITH THE SPUR BLOCK
      M1=JCCORE(17)+8
      MC=JCCORE(M1)
      M1=M1+1
      M2=JCCORE(M1)
      M1=M1+5
      M2=M2+JCCORE(M1)
      IF(M2.GT.MC)MC=M2
      IF(10.GT.MC)MC=10
C                  MD=ADDRESS OF SPUR FOR GROUP
    2 M1=JCCORE(17)+8
      MD=MB+5*(JCCORE(M1)-1)
C                  ME=ADDRESS OF SPUR FOR GROUP BODY ENTRY (NO INDICES)
      M1=M1+1
      ME=MB+5*(JCCORE(M1)-1)
```

```
C                   MF=ADDRESS OF SPUR FOR OLD VALUE UPDATE INSERT
      M1=M1+3
      MF=M8+5*(JCCORE(M1)-1)
C                   MG=ADDRESS OF SPUR FOR METHOD INSERT
      M1=M1+1
      MG=M8+5*(JCCORE(1)-1)
C                   S(4)
      M1=M1-2
      S(4)=JCCORE(M1)
C                   C(4)
      M1=M1-11
      C(4)=JCCORE(M1)
      L(2)=1
      L(3)=1
      L(5)=0
C                   IN OR OUT
      IF(JCSENS.LT.1)GO TO 16
C                   INPUT
C                   MH=ADDRESS OF SPUR FOR LARGE CORE BLOCKS
      MH=M8-5
C                   MI=INPUT DATA SET
      MI=JCCORE(10)
C                   MJ=INDICATOR
      MJ=1
C                   MK=METHOD LIST HEADCELL
      MK=0
C                   MM=TEMPORARY LIST ENTRY
      M1=M1+8
      M2=S(4)
      S(4)=JCCORE(M1)
      CALL NEWCEL(S,MM,1)
      S(4)=M2
C                   READ
    3 READ(MI,101)(D(I),I=1,5)
  101 FORMAT(I1,I19,3I20)
      IF(D(1).EQ.0)GO TO 10
C                   TYPE
      M1=C(2)+1
      GO TO (4,5,5,5,5,5,5,5,5,7),M1
C                   GROUP
    4 CALL J1FILE(C,L(MJ),MO,M1,M2,1,F,Z,1)
      MJ=3
      M2=M2-1
      IF(C(4).GT.0)CALL J1DIMS(M2,1,2,2,1,1)
      D(2)=1
      D(6)=0
      D(7)=0
      M2=0
      CALL J1VARI(M2,D(6),2,M2,1,1)
      CALL J1VARI(M2,D(7),2,M2,1,2)
      CALL TOCELL(JCCORE(MO),M1,D(2),G,1)
      GO TO 3
C                   GROUP BODY
    5 M1=ME+D(4)*10
      CALL J1FILE(C,L(MJ),M1,M2,M3,1,F,Z,2)
      MJ=3
      M3=M3-D(4)*2+1
      IF(C(4).GT.0)CALL J1DIMS(M3,D(4),2,2,1,2)
      D(2)=C(2)-1
      D(1)=2
      D(6)=0
      D(7)=0
```

```
      CALL J1VARI(JCSPUR,D(6),1,MK,1,3)
      CALL J1LEDL(S,D(7),MK,1,0,1)
      CALL TOCELL(JCCORE(M1),M2,D,8,2)
      GO TO 3
C                     INSERT - METHOD
    7 CALL J1FILE(C,L(MJ),MG,M1,M2,1,F,Z,4)
      MJ=3
      CALL J1VTVF(MG,M1,D(3),1)
      GO TO 3
C                     LINKAGE INPUT
   10 D(5)=0
      D(6)=0
      MN=S(4)
      M1=L(2)
      M2=3-M1
C                     READ LINKAGE (PART I)
   11 READ(MI,101)M3,M4,L(3)
      IF(M3.EQ.0)GO TO 15
C                     MODE
      M3=MN
      IF(L(3).LE.0)GO TO 12
      L(5)=L(3)-Z(8)
      CALL FETCH(C,L(4),M3,P,L,Z,0,1)
C                     READ LINKAGE (PART II)
   12 M5=4
      IF(M5.GT.M4)M5=M4
      M4=M4-M5
      READ(MI,102)(O(I),I=1,M5)
  102 FORMAT(4I20)
C                     SET LINKAGES
      DO 13 I=1,M5
      L(5)=D(I)-Z(8)
      CALL FETCH(C,L(4),M6,P,L,Z,0,2)
      CALL COUPLE(MA,M3,M6,-1,-1,0,I)
   13 IF(L(3).LE.0)CALL TOCELL(JCCORE(MD),M6,D(5),V,5)
C                     ITERATE
      IF(M4.GT.0)GO TO 12
      IF(L(3).GT.0)GO TO 11
C                     MULTI-LEVEL
      IF(L(3).LT.0)GO TO 14
C                     GIMBL POINTER
      JCCORE(MA)=LNKFWD(S,MM,1)
      GO TO 141
C                     GROUP BODY POINTER
   14 L(2)=M2
      L(3)=-L(3)
      CALL FETCH(C,L,M6,P,L,T,0,3)
      M3=LNKFWD(S,MM,2)
      CALL TOCELL(JCCORE(MD),M6,M3,H,6)
  141 CALL TOCELL(JCCORE(MD),MM,D(5),W,7)
      GO TO 11
C                     ITERATE
   15 IF((O(I)+M3).EQ.0)GO TO 40
      S(4)=MN
      L(5)=0
      L(3)=1
      L(2)=M2
      MJ=1
      GO TO 3
C                     OUTPUT
C                     MH AND MI ARE OUTPUT DATA SETS
   16 MH=JCCORE(11)
```

```
      MI=IABS(JCSENS)
C                     MJ AND MK ARE GROUP LIST HEADCELLS
      MJ=0
      MK=0
C                     ML=TAILCELL CORRESPONDING TO MJ
      ML=0
C                     MM=LEVEL
      MM=0
C                     FILE ENTRY FOR GIMBL
      D(2)=0
      D(3)=0
      CALL STORE(C,L,D(2),E,F,Z,1)
C                     MN=FILE ENTRY NUMBER OF CURRENT PARENT
      MN=1
C                     MO=PARENT GROUP INDICATOR
      MO=0
C                     M
      M1=JCCORE(17)+13
      M(3)=JCCORE(M1)
      M(4)=M(3)
C                     PREPARE FOR TRACE
      M1=JCCORE(MA)
      M2=2
      IF(M1.LE.0)GO TO 41
      M2=MA+2
      M2=JCCORE(M2)+1
      M3=0
      M4=0
      G(3)=2
      B(3)=2
      M(4)=1
      U(1)=1
      U(3)=2
      M5=M1
      M6=-1
C                     PREFACE
      WRITE(MH,103)
  103 FORMAT(115H1THE FOLLOWING TABLE GIVES THE VARIOUS OUTPUT FIELDS WH
     1ICH MAY APPEAR AS OUTPUT.  EACH LINE OF OUTPUT BEGINS WITH A)
      WRITE(MH,104)
  104 FORMAT(118H LETTER DESIGNATION.  THIS LETTER CORRESPONDS TO ONE OF
     1 THE ENTRIES IN THIS TABLE, THUS IDENTIFYING THE VARIOUS FIELDS)
      WRITE(MH,105)
  105 FORMAT(30H IN THE LINE OF GENDER OUTPUT.)
      WRITE(MH,106)
  106 FORMAT(60H A  IDENTIFICATION FLAG, PROTECTION FLAG, DEPTH, METHOD
     1CODE)
      WRITE(MH,107)
  107 FORMAT(107H B  MAPPING FLAG, MAPPING INDEX, OPERATOR FLAG, SCALE S
     1IGN FLAG, SCALE, OFFSET SIGN FLAG, OFFSET (OR INDEX))
      WRITE(MH,108)
  108 FORMAT(54H C  VARIABLE CODE NAME, DIMENSIONALITY, SELECTION FLAG)
      WRITE(MH,109)
  109 FORMAT(69H D  IDENTIFICATION FLAG, CODE NAME, DIMENSIONALITY, NUMB
     1ER OF OUTPUTS)
      WRITE(MH,110)
  110 FORMAT(70H E  IDENTIFICATION FLAG, CODE NAME, DIMENSIONALITY (OROE
     1R OF OPERATOR))
      WRITE(MH,111)
  111 FORMAT(39H F  IDENTIFICATION FLAG, DIMENSIONALITY)
      WRITE(MH,112)
  112 FORMAT(42H G  IDENTIFICATION FLAG, NUMBER OF ENTRIES)
```

```
      WRITE(MH,113)
  113 FORMAT(78H H  NUMBER OF PATHS OUT OF A NODE (FORWARD), SEQUENCE NU
     1MBER OF NODE (LEADING))
      WRITE(MH,114)
  114 FORMAT(113H I  SEQUENCE NUMBER OF NODE (TRAILING), SEQUENCE NUMBER
     1 OF NODE, SEQUENCE NUMBER OF NODE, SEQUENCE NUMBER OF NODE)
      WRITE(MH,115)
  115 FORMAT(7H1GENDER)
C                     LIST ENTRY DATA
   17 WRITE(MH,116)MN
  116 FORMAT(24H0LIST ENTRIES FOR LEVEL I2)
      JCCORE(13)=6
C                     MP=FILE ENTRY NUMBER OF CURRENT ENTRY
      MP=1
C                     MQ=LINEAR SEQUENCE NUMBER OF CURRENT ENTRY
      MQ=0
C                     NEXT
   18 IF(M5.EQ.0)GO TO 371
      M1=NEXT(MA,M5,M2,1,M6,M3,D,0,1)
      IF(MM.EQ.0)M4=M2
      IF((M1*M6).EQ.0)GO TO 27
C                     INCREASE COUNTERS
      MP=MP+1
      MQ=MQ+1
C                     RECORD LINKAGES
      M5=M1
   19 M6=LNKBNT(MA,M5,M7,0,1)
      L(5)=MN
      IF(M6.EQ.0)GO TO 20
      CALL FRMCEL(JCCORE(MD),M6,L(5),N,1)
   20 L(5)=L(5)-Z(8)
      CALL FETCH(C,L(4),M6,Y,F,Z,0,4)
      M5=M6
      CALL PUSH(S,M6,MQ,A,1,1)
      L(5)=0
      IF(M5.EQ.0)CALL STORE(C,L(4),M6,Y,F,Z,2)
      M5=M7
      IF(M7.GT.0)GO TO 19
      L(5)=MP-Z(8)
C                     TEST FLAG
      CALL FRMCEL(JCCORE(MD),M1,D(2),Q,2)
      M5=D(2)
      GO TO (21,22,24),M5
C                     GROUP
   21 D(2)=0
      CALL FRMCEL(JCCORE(MD),M1,D(3),G,3)
      WRITE(MH,117)(D(I),I=2,5)
  117 FORMAT(4H A  4I29)
      IF(JCSENS.LT.0)WRITE(MI,101)(D(I),I=1,5)
      JCCORE(13)=6
      CALL J1FILE(C,L(4),MO,M1,M5,0,F,Z,5)
      M5=M5-1
      IF(D(4).GT.0)CALL J1DIMS(M5,1,2,2,JCSENS,4)
      WRITE(MH,118)
  118 FORMAT(25H (DECISION VARIABLE LIST))
      JCCORE(13)=6
      CALL J1VARI(0,D(6),2,0,JCSENS,4)
      IF(JCSENS.LT.0)WRITE(MI,119)
  119 FORMAT(1H )
      WRITE(MH,120)
  120 FORMAT(21H (TEAR VARIABLE LIST))
      JCCORE(13)=6
```

```
      CALL J1VARI(0,D(7),2,0,JCSENS,5)
      IF(JCSENS.LT.0)WRITE(MI,119)
C                     TEMPORARY GROUP LIST
      D(6)=M1
      D(7)=MQ
      CALL PUSH(S,ML,D(6),K,2,2)
      IF(MJ.EQ.0)MJ=ML
      GO TO 26
C                     ER, EQUATION OR CONSTRAINT
   22 CALL FRMCEL(JCCORE(ME),M1,D(2),8,4)
      D(2)=D(2)+1
      WRITE(MH,121)(D(I),I=2,5)
  121 FORMAT(4H D  4I29)
      IF(JCSENS.LT.0)WRITE(MI,101)(D(I),I=1,5)
      JCCORE(13)=6
      CALL J1FILE(C,L(4),ME,M1,M5,0,F,Z,6)
      M5=M5+1
      IF(D(4).GT.0)CALL J1DIMS(M5,D(4),2,2,JCSENS,5)
      WRITE(MH,122)
  122 FORMAT(23H (OUTPUT VARIABLE LIST))
      JCCORE(13)=6
      CALL J1VARI(0,D(6),1,0,JCSENS,6)
      IF(JCSENS.LT.0)WRITE(MI,119)
   23 WRITE(MH,123)
  123 FORMAT(19H (LINK EDITOR LIST))
      JCCORE(13)=6
      CALL J1LEDL(0,D(7),0,JCSENS,0,3)
      IF(JCSENS.LT.0)WRITE(MI,119)
      GO TO 26
C                     INSERT
   24 CALL FRMCEL(JCCORE(MG),M1,D(2),U,5)
      D(2)=D(2)+8
C                     METHOD
   25 CALL FRMCEL(JCCORE(MG),M1,D(3),M,7)
      WRITE(MH,125)(D(I),I=2,3)
  125 FORMAT(4H G  2I29)
      IF(JCSENS.LT.0)WRITE(MI,101)(D(I),I=1,3)
      JCCORE(13)=6
      CALL J1FILE(C,L(4),MF,M1,M5,0,F,Z,8)
C                     STORE LINEAR SEQUENCE NUMBER
   26 CALL TOCELL(JCCORE(MD),M1,MP,N,8)
C                     ITERATE
      M5=M1
      M6=-1
      GO TO 18
C                     NEXT PARENT GROUP
   27 IF(MO.EQ.0)GO TO 29
      IF(MK.EQ.0)GO TO 30
      MO=MO+1
      MP=MP+1
      MN=MP
      L(5)=1
   28 CALL POPUP(S,MK,MK,D(6),K,1,2)
      CALL J1FILE(C,L(4),MD,-D(7),M5,0,F,Z,9)
      CALL FRMCEL(JCCORE(MD),D(6),M2,X,8)
      M5=D(6)
      M6=MM-1
      IF(MO.EC.1)GO TO 17
      GO TO 18
C                     LINKAGE OUTPUT - PART I
   29 MO=2
   30 L(5)=1-Z(8)
```

```
      IF(JCSENS.LT.0)WRITE(MI,119)
      WRITE(MH,126)MM
  126 FORMAT(19H LINKAGE FOR LEVEL I2)
      JCCORE(13)=6
      M5=0
      M6=0
   31 M5=M5+1
      IF(M5.GT.MP)GO TO 37
      CALL FETCH(C,L(4),D(3),E,F,Z,0,5)
      L(5)=0
      CALL STORE(C,L(4),L(5),Y,F,Z,3)
      IF(D(3).LE.0)GO TO 32
      M6=M6+1
      D(3)=M6
   32 IF(D(4).EQ.0)GO TO 36
      D(2)=0
   33 M7=D(4)
      D(2)=D(2)+1
      D(4)=LNKFWD(S,M7,3)
      IF(D(4).GT.0)GO TO 33
      WRITE(MH,127)(D(I),I=2,3)
  127 FORMAT(4H H  2I29)
      IF(JCSENS.LT.0)WRITE(MI,101)(D(I),I=1,3)
      JCCORE(13)=6
C                     LINKAGE OUTPUT - PART II
   34 M8=2
   35 M8=M8+1
      CALL POPUP(S,M7,M7,D(M8),A,2,3)
      IF(((M8-6)*(D(2)+2-M8)).NE.0)GO TO 35
      D(2)=D(2)-M8+2
      WRITE(MH,128)(D(I),I=3,M8)
  128 FORMAT(4H I  4I29)
      IF(JCSENS.LT.0)WRITE(MI,101)(D(I),I=3,M8)
      JCCORE(13)=6
      IF(D(2).GT.0)GO TO 34
C                     NEXT ENTRY
   36 L(5)=1
      GO TO 31
C                     NEXT LEVEL
   37 IF(JCSENS.LT.0)WRITE(MI,119)
      IF(MJ.EQ.0)GO TO 38
      MK=MJ
      MM=MM+1
      L(5)=1-Z(8)
      MN=1
      MO=1
      MJ=0
      ML=0
      GO TO 28
C                     BREAK/BREAK
  371 CALL SYSERR(3,MM)
   38 IF(JCSENS.GE.0)GO TO 39
      WRITE(MI,119)
      WRITE(MI,119)
   39 WRITE(MH,129)
  129 FORMAT(14HOEND OF GENDER)
      JCCORE(13)=6
      M1=MA+2
      JCCORE(M1)=M4
C                     RETURN CORE CHARGED TO S AND C
   40 CALL CORN(S,1)
      CALL CORN(C,2)
```

```
      GO TO 42
C                    ERROR
   41 CALL SYSERR(M2,M1)
C                    EXIT
   42 JCCORE(18)=SAVE
      CALL SYSEXT
      RETURN
      END
```

```
C
C*****************************************************************
C                              **********
C                              * SECIAO *
C                              **********
C
C       PURPOSE
C           SECEDE HANDLES THE INPUT AND OUTPUT OF THE DATA ASSOCIATED
C           WITH THE SERVICE MODULE.
C
C       USAGE
C           CALL SECIAO(JCSENS,JCCLNO)
C
C       DATA FORMAT
C           THE SYMBOLS  , + AND $ ARE USED TO DELIMIT LOOPS.  8
C           INDICATES A BLANK CARD.  THE SYMBOLS *N* DO NOT APPEAR ON
C           THE CARDS READ, BUT ARE USED HERE TO INDICATE A CARD TYPE
C           NUMBER, N, WHICH CORRESPONDS TO THE FOLLOWING DATA FORMATS.
C               CARD TYPE     FORMAT
C               1             I1,I19,I20
C               2             4I20
C               3             I1,I19,SKIP 14,A6
C               4             I1,I19
C               5             I1,I19,I20,D40.28
C               6             I1,I19,3I20
C               7             I1,I19,2I20
C               8             I1,I19,D40.28
C               9             I1,I19,I1,I19,I20
C              10             I5,I20,2I5,I20,I5,I20
C           THE DATA ORGANIZATION FOLLOWS, WITH COMMENTS ENCLOSED IN
C           PARENTHESIS.
C           (DECISION VARIABLE FILE)
C
C           + 1, DECISION VARIABLE CODE NAME, DIMENSIONALITY *1*
C           +
C           ++ INDICES *10* (MIN. AND MAX.)
C           $$
C           8
C           (TORN VARIABLE FILE)
C
C           + 1, TORN VARIABLE CODE NAME, DIMENSIONALITY *1*
C           +
C           ++ INDICES *10* (MIN. AND MAX.)
C           $$
C           8
C           (OPERATOR NAME FILE)
C
C           + 1, OPERATOR CODE NAME, OPERATOR ALPHAMERIC NAME *3*
C           $
C           8
C           (METHOD NAME FILE)
C
C           + 1, METHOD CODE NAME, METHOD ALPHAMERIC NAME *3*
C           $
C           8
C           (UNIT 0 - CONNECTIONS)
C
C           + 1, NUMBER OF COMMON VARIABLES *4*
C           +(COMMON VARIABLE DECLARATION FILE)
C           +
C           ++ 1, COMMON VARIABLE CODE NAME, DIMENSIONALITY *1*
C           ++
```

```
C           +++ MAXIMUM INDICES *2*
C           +$$
C           +B
C           +(COMMON VARIABLE VALUE FILE)
C           +
C           ++ 1, VARIABLE CODE NAME, DIMENSIONALITY, VALUE *5*
C           ++
C           +++ INDICES *2*
C           +$$
C           +B
C           +(COMMON VARIABLE NAME FILE)
C           +
C           ++ 1, COMMON VARIABLE CODE NAME, COMMON VARIABLE ALPHAMERIC
C           ++     NAME *3*
C           $$
C           B
C           (UNIT - REPEAT AS NECESSARY)
C
C           + 1, NUMBER OF VARIABLES, NUMBER OF ER'S, NUMBER OF
C           +     EQUATIONS, NUMBER OF CONSTRAINTS *6*
C           +(VARIABLE DECLARATION FILE)
C           +
C           ++ 1, VARIABLE CODE NAME, COMMON VARIABLE CODE NAME,
C           ++     DIMENSIONALITY *7*
C           ++
C           +++ MAXIMUM INDICES *2*
C           +$$
C           +B
C           +(ER DECLARATION FILE)
C           +
C           ++ 1, ER CODE NAME, DIMENSIONALITY *1*
C           ++
C           +++ MAXIMUM INDICES *2*
C           +$$
C           +B
C           +(EQUATION DECLARATION FILE)
C           +
C           ++ 1, EQUATION CODE NAME, DIMENSIONALITY *1*
C           ++
C           +++ MAXIMUM INDICES *2*
C           +$$
C           +B
C           +(CONSTRAINT DECLARATION FILE)
C           +
C           ++ 1, CONSTRAINT CODE NAME, DIMENSIONALITY *1*
C           ++
C           +++ MAXIMUM INDICES *2*
C           +$$
C           +B
C           +(CONSTANT VALUE FILE)
C           +
C           ++ 1, CONSTANT CODE NAME, VALUE *8*
C           +$
C           +B
C           +(VARIABLE VALUE FILE)
C           +
C           ++ 1, VARIABLE CODE NAME, DIMENSIONALITY, VALUE *5*
C           ++
C           +++ INDICES *2*
C           +$$
C           +B
C           +(ER NAME FILE)
```

```
C            +
C            ++ 1, ER CODE NAME, ER ALPHAMERIC NAME *3*
C            +$
C            +B
C            +(VARIABLE NAME FILE)
C            +
C            ++ 1, VARIABLE CODE NAME, VARIABLE ALPHAMERIC NAME *3*
C            +$
C            +B
C            +(ER FILE)
C            +
C            ++ 1, ER CODE NAME, NUMBER OF OUTPUTS, DIMENSIONALITY *7*
C            ++
C            +++ INDICES *10* (MIN. AND MAX.)
C            ++$
C            ++
C            +++ 1, CODE NAME, IDENTIFICATION FLAG, COST (ORDER OF
C            +++      OPERATOR), DIMENSIONALITY *9*
C            +++
C            ++++ INDICES *10*
C            ++$$
C            ++B
C            +$
C            +B
C            +(EQUATION FILE)
C            +
C            ++ 1, EQUATION CODE NAME, NUMBER OF OUTPUTS, DIMENSIONALITY
C                  *7*
C            ++
C            +++ INDICES *10* (MIN. AND MAX.)
C            ++$
C            ++
C            +++ 1, CODE NAME, IDENTIFICATION FLAG, COST (ORDER OF
C            +++      OPERATOR), DIMENSIONALITY *9*
C            +++
C            ++++ INDICES *10*
C            ++$$
C            ++B
C            +$
C            +B
C            +(CONSTRAINT FILE)
C            +
C            ++ 1, CONSTRAINT CODE NAME, NUMBER OF OUTPUTS,
C                  DIMENSIONALITY *7*
C            ++
C            +++ INDICES *10* (MIN. AND MAX.)
C            ++$
C            ++
C            +++ 1, CODE NAME, IDENTIFICATION FLAG, COST (ORDER OF
C            +++      OPERATOR), DIMENSIONALITY *9*
C            +++
C            ++++ INDICES *10*
C            ++$$
C            ++B
C            +$
C            +B
C            $
C            B
C
C       DESCRIPTION OF PARAMETERS
C           JCSENS = -N, 0 OR 1 AS RESPECTIVELY PUNCHED (AND WRITTEN)
C                    OUTPUT, WRITTEN ONLY OUTPUT OR INPUT IS DESIRED.
```

```
C                    N = THE CARD PUNCH DATA SET REFERENCE NUMBER
C               JCCLNO = A USER SPECIFIED CALL STATEMENT ID. NUMBER
C
C          REMARKS
C               THE NUMBER REPRESENTED BY JCTCIA MUST, FOR SOME MACHINES,
C               REPLACE THE 1 IN THE NAMED COMMON STATEMENT.
C               THE FUNCTIONS AND CONSTRAINTS MUST BE PROVIDED IN REVERSE
C               POLISH NOTATION WITH =0 UNDERSTOOD.
C
C          SUBRCUTINE AND FUNCTION SUBPROGRAMS REQUIRED
C               FETCH
C               J1CONS
C               J1DECL
C               J1NAME
C               J1RAFE
C               J1VALU
C               NEWCEL
C               STORE
C               SYSENT
C               SYSERR
C               SYSEXT
C
C          ERROR CODES FOR THIS PROGRAM
C               CODE FATAL  ERROR(DATA PROVIDED)
C               1    NF     NO VARIABLES DECLARED (UNIT NUMBER)
C               2    NF     NO ER'S, FUNCTIONS OR CONSTRAINTS (UNIT NUMBER)
C               3    NF     NO UNITS (UNIT NUMBER)
C
C          METHOD
C               SELF-EXPLANATORY
C
C                                   **********
C
      SUBROUTINE SECIAO(JCSENS,JCCLNO)
      COMMON/ALLOC/JCCORE(1)
      DIMENSION L(7),O(8),F(4),P(4)
      INTEGER C,F,P,U
C                    ENTER
      CALL SYSENT(1,5,1,2,JCCLNO)
      DO 91 I=1,7
   91 L(I)=1
      DO 92 I=1,4
      F(I)=1
   92 P(I)=1
C                    SECBL SET UP
      IF(JCSENS.LT.1)GO TO 4
C                    D IS TEMPORARY SPUR
      IF(JCCORE(15).GT.0)CALL SYSERR(1,JCCORE(15))
      D(1)=0
      D(2)=0
      D(3)=1
      D(4)=60
      D(5)=1
C                    LOAD BLOCK
      CALL NEWCEL(D,MB,1)
      MBA=MB+28
C                    SPUR BLOCK
      M1=JCCORE(17)
      M1=JCCORE(M1)
      M2=M1/2+2
      D(2)=0
      D(4)=M2*5
```

```
      CALL NEWCEL(0,MA,2)
C                     INITIALIZE SPUR BLOCK
      M3=MA
      DO 1 I=1,M2
      DO 1 J=1,5
      JCCORE(M3)=C(1)*(1/J)+(J/3)+(I-2+M1*(1/I))*(4/J)*(J/4)
    1 M3=M3+1
C                     SECBL
      M1=MA+20
      CALL NEWCEL(JCCORE(M1),M2,3)
      JCCORE(15)=M2
      M1=M2+1
      JCCORE(M1)=MA
      M1=M1+1
      JCCORE(M1)=MB
C                     INITIALIZE LOAD BLOCK
      M3=MB+14
      DO 2 I=1,2
      DO 2 J=1,2
      JCCORE(M3)=-1+(M2+1)*(J/2)
    2 M3=M3+15
C                     INDEX STORAGE
      M2=JCCORE(17)+14
      M2=2*JCCORE(M2)
      M3=JCCORE(17)
      M3=JCCORE(M3)/2
      M4=M2
      IF(M2.LE.M3)GO TO 3
      M4=M3+1
      M3=MA+5*M4+3
      JCCORE(M3)=M2
    3 M3=MA+5*M4
      CALL NEWCEL(JCCORE(M3),M2,4)
      M1=M1+1
      JCCORE(M1)=M2
C                     MC=INPUT DATA SET
      MC=JCCORE(10)
      GO TO 5
C                     MA=ADDRESS OF SPUR
    4 M1=JCCORE(15)+1
      MA=JCCORE(M1)
C                     MB=ADDRESS OF LOAD
      M1=M1+1
      MB=JCCORE(M1)
      MBA=MB+28
C                     MC AND MD ARE OUTPUT DATA SETS
      MC=JCCORE(11)
      MD=IABS(JCSENS)
C                     PREFACE
      WRITE(MC,101)
  101 FORMAT(115H1THE FOLLOWING TABLE GIVES THE VARIOUS OUTPUT FIELDS WH
     1ICH MAY APPEAR AS OUTPUT.  EACH LINE OF OUTPUT BEGINS WITH A)
      WRITE(MC,102)
  102 FORMAT(118H LETTER DESIGNATION.  THIS LETTER CORRESPONDS TO ONE OF
     1 THE ENTRIES IN THIS TABLE, THUS IDENTIFYING THE VARIOUS FIELDS)
      WRITE(MC,103)
  103 FORMAT(30H IN THE LINE OF SECEDE OUTPUT.)
      WRITE(MC,104)
  104 FORMAT(29H A  CODE NAME, DIMENSIONALITY)
      WRITE(MC,105)
  105 FORMAT(107H B  MAPPING FLAG, MAPPING INDEX, OPERATOR FLAG, SCALE S
     1IGN FLAG, SCALE, OFFSET SIGN FLAG, OFFSET (OR INDEX))
```

```
      WRITE(MC,106)
  106 FORMAT(30H C  CODE NAME, ALPHAMERIC NAME)
      WRITE(MC,107)
  107 FORMAT(30H D  NUMBER OF COMMON VARIABLES)
      WRITE(MC,108)
  108 FORMAT(62H E  MAXIMUM INDEX, MAXIMUM INDEX, MAXIMUM INDEX, MAXIMUM
     1 INDEX)
      WRITE(MC,109)
  109 FORMAT(45H F  VARIABLE CODE NAME, DIMENSIONALITY, VALUE)
      WRITE(MC,110)
  110 FORMAT(30H G  INDEX, INDEX, INDEX, INDEX)
      WRITE(MC,111)
  111 FORMAT(83H H  NUMBER OF VARIABLES, NUMBER OF ER'S, NUMBER OF EQUAT
     1IONS, NUMBER OF CONSTRAINTS)
      WRITE(MC,112)
  112 FORMAT(65H I  VARIABLE CODE NAME, COMMON VARIABLE CODE NAME, DIMEN
     1SIONALITY)
      WRITE(MC,113)
  113 FORMAT(29H J  CONSTANT CODE NAME, VALUE)
      WRITE(MC,114)
  114 FORMAT(48H K  CODE NAME, NUMBER OF OUTPUTS, DIMENSIONALITY)
      WRITE(MC,115)
  115 FORMAT(76H L  CODE NAME, IDENTIFICATION FLAG, COST (ORDER OF OPERA
     1TOR), DIMENSIONALITY)
      WRITE(MC,116)
  116 FORMAT(7H1SECEDE)
      WRITE(MC,117)
  117 FORMAT(23HDECISION VARIABLE FILE)
      JCCORE(13)=6
C                   DECISION VARIABLE FILE
    5 L(1)=4
      F(1)=2
      JCCORE(MBA)=-1
      CALL JIRAFE(L,1,JCSENS,F,1)
      IF(JCSENS.LT.0)WRITE(MD,118)
  118 FORMAT(1H )
C                   TEAR VARIABLE FILE
      L(3)=2
      L(4)=1
      IF(JCSENS.EQ.1)GO TO 51
      WRITE(MC,119)
  119 FORMAT(19H TEAR VARIABLE FILE)
      JCCORE(13)=6
   51 CALL JIRAFE(L,1,JCSENS,F,2)
      IF(JCSENS.LT.0)WRITE(MD,118)
C                   OPERATOR NAME FILE
      L(1)=3
      L(2)=2
      L(3)=1
      L(4)=1
      F(1)=1
      M1=JCCORE(17)+2
      F(2)=JCCORE(M1)+1
      JCCORE(MBA)=0
      IF(JCSENS.EQ.1)GO TO 52
      WRITE(MC,120)
  120 FORMAT(19HOOPERATOR NAME FILE)
      JCCORE(13)=6
   52 CALL JINAME(L,JCSENS,F,1)
      IF(JCSENS.LT.0)WRITE(MD,118)
C                   METHOD NAME FILE
      L(2)=3
```

```
      IF(JCSENS.EQ.1)GO TO 53
      WRITE(MC,121)
  121 FORMAT(17HOMETHOD NAME FILE)
      JCCORE(13)=6
   53 CALL JINAME(L,JCSENS,F,2)
      IF(JCSENS.LT.0)WRITE(MD,118)
C                     UNIT 0 - CONNECTIONS
      IF(JCSENS.EQ.1)GO TO 54
      WRITE(MC,122)
  122 FORMAT(21HOUNIT 0 = CONNECTIONS)
      JCCORE(13)=6
C                     NUMBER OF COMMON VARIABLES
   54 L(2)=4
      L(3)=2
      F(1)=4
      F(2)=1
      IF(JCSENS.LT.1)GO TO 6
C                     INPUT
      READ(MC,123)M1,M2
  123 FORMAT(I1,I19,3I20)
      IF(M1.EQ.0)GO TO 8
      CALL STORE(JCCORE(MA),L,M2,P,F,JCCORE(MB),1)
      GO TO 7
C                     OUTPUT
    6 M1=1
      CALL FETCH(JCCORE(MA),L,M2,P,F,JCCORE(MB),M1,1)
      IF(M1.EQ.-1)GO TO 74
      WRITE(MC,124)M2
  124 FORMAT(4H 0  I29)
      IF(JCSENS.LT.0)WRITE(MD,123)M1,M2
C                     COMMON VARIABLE DECLARATION FILE
    7 L(1)=4
      L(3)=1
      L(4)=1
      F(2)=M2
      JCCORE(MBA)=-1
      IF(JCSENS.EQ.1)GO TO 71
      WRITE(MC,125)
  125 FORMAT(33H COMMON VARIABLE DECLARATION FILE)
      JCCORE(13)=6
   71 CALL J1DECL(L,JCSENS,F,1)
      IF(JCSENS.LT.0)WRITE(MD,118)
C                     COMMON VARIABLE VALUE FILE
      L(3)=3
      L(4)=1
      M1=JCCORE(17)+1
      F(3)=JCCORE(M1)
      IF(JCSENS.EQ.1)GO TO 72
      WRITE(MC,126)
  126 FORMAT(27H COMMON VARIABLE VALUE FILE)
      JCCORE(13)=6
   72 CALL J1VALU(L,JCSENS,F,1)
      IF(JCSENS.LT.0)WRITE(MD,118)
C                     COMMON VARIABLE NAME FILE
      L(3)=4
      L(4)=1
      M1=M1+1
      JCCORE(MBA)=0
      F(3)=JCCORE(M1)+1
      IF(JCSENS.EQ.1)GO TO 73
      WRITE(MC,127)
  127 FORMAT(26H COMMON VARIABLE NAME FILE)
```

```
      JCCORE(13)=6
   73 CALL JINAME(L,JCSENS,F,3)
   74 IF(JCSENS.LT.0)WRITE(MD,118)
C                    UNIT
    8 U=0
    9 U=U+1
      L(1)=4
      L(2)=4+U
      L(3)=1
      F(1)=8
      F(2)=8
      F(3)=1
      P(4)=8
      JCCORE(M8A)=-1
      IF(JCSENS.LT.1)GO TO 11
C                    INPUT
      DO 10 I=1,8
   10 D(I)=0
      READ(MC,123)M1,D(2),D(4),D(6),D(8)
      IF(M1.EQ.0)GO TO 19
      CALL STORE(JCCORE(MA),L,D,P,F,JCCORE(MB),2)
      GO TO 12
C                    OUTPUT
   11 M1=1
      CALL FETCH(JCCORE(MA),L,D,P,F,JCCORE(M8),M1,2)
      IF(M1.EQ.-1)GO TO 20
      WRITE(MC,128)U
  128 FORMAT(8HOUNIT = I5)
      WRITE(MC,129)D(2),D(4),D(6),D(8)
  129 FORMAT(4H H  4(29)
      IF(JCSENS.LT.0)WRITE(MD,123)M1,D(2),D(4),D(6),D(8)
C                    VARIABLE DECLARATION FILE
   12 L(1)=5
      F(3)=D(2)
      IF(JCSENS.EQ.1)GO TO 1201
      WRITE(MC,130)
  130 FORMAT(26H VARIABLE DECLARATION FILE)
      JCCORE(13)=6
 1201 CALL J1DECL(L,JCSENS,F,2)
      IF(JCSENS.LT.0)WRITE(MD,118)
C                    ER DECLARATION FILE
      L(4)=3
      L(5)=1
      F(3)=D(4)
      IF(JCSENS.EQ.1)GO TO 1202
      WRITE(MC,131)
  131 FORMAT(20H ER DECLARATION FILE)
      JCCORE(13)=6
 1202 CALL J1DECL(L,JCSENS,F,3)
      IF(JCSENS.LT.0)WRITE(MD,118)
C                    EQUATION DECLARATION FILE
      L(4)=5
      L(5)=1
      F(3)=D(6)
      IF(JCSENS.EQ.1)GO TO 1203
      WRITE(MC,132)
  132 FORMAT(26H EQUATION DECLARATION FILE)
      JCCORE(13)=6
 1203 CALL J1DECL(L,JCSENS,F,4)
      IF(JCSENS.LT.0)WRITE(MD,118)
C                    CONSTRAINT DECLARATION FILE
      L(4)=7
```

```
      L(5)=1
      F(3)=D(8)
      IF(JCSENS.EQ.1)GO TO 1204
      WRITE(MC,133)
  133 FORMAT(28H CONSTRAINT DECLARATION FILE)
      JCCCRE(13)=6
 1204 CALL J1DECL(L,JCSENS,F,5)
      IF(JCSENS.LT.0)WRITE(MO,118)
C                    CONSTANT VALUE FILE
      L(1)=4
      L(3)=2
      L(4)=1
      M1=JCCORE(17)+1
      F(2)=1
      F(3)=JCCORE(M1)
      JCCCRE(M8A)=0
      IF(JCSENS.EQ.1)GO TO 1205
      WRITE(MC,134)
  134 FORMAT(20H CONSTANT VALUE FILE)
      JCCCRE(13)=6
 1205 CALL J1CONS(L,JCSENS,F,1)
      IF(JCSENS.LT.0)WRITE(MO,118)
C                    VARIABLE VALUE FILE
      L(3)=3
      F(2)=D(2)
      IF(JCSENS.EQ.1)GO TO 1206
      WRITE(MC,135)
  135 FORMAT(20H VARIABLE VALUE FILE)
      JCCCRE(13)=6
 1206 CALL J1VALU(L,JCSENS,F,2)
      IF(JCSENS.LT.0)WRITE(MO,118)
C                    ER NAME FILE
      L(3)=4
      L(4)=1
      F(2)=D(4)
      M1=M1+1
      F(3)=JCCORE(M1)+1
      IF(JCSENS.EQ.1)GO TO 1207
      WRITE(MC,136)
  136 FORMAT(13H ER NAME FILE)
      JCCCRE(13)=6
 1207 CALL J1NAME(L,JCSENS,F,4)
      IF(JCSENS.LT.0)WRITE(MO,118)
C                    VARIABLE NAME FILE
      L(3)=5
      F(2)=D(2)
      IF(JCSENS.EQ.1)GO TO 1208
      WRITE(MC,137)
  137 FORMAT(19H VARIABLE NAME FILE)
      JCCCRE(13)=6
 1208 CALL J1NAME(L,JCSENS,F,5)
      IF(JCSENS.LT.0)WRITE(MO,118)
C                    ER FILE
      L(1)=5
      L(3)=6
      L(5)=1
      F(2)=D(4)
      F(3)=1
      JCCCRE(M8A)=-1
      IF(JCSENS.EQ.1)GO TO 13
      WRITE(MC,138)
  138 FORMAT(8H ER FILE)
```

```
      JCCCRE(13)=6
   13 IF((L(4).GT.F(2)).AND.(JCSENS.LE.0))GO TO 14
      CALL J1RAFE(L,2,JCSENS,F,3)
      IF(JCSENS.LT.0)WRITE(MD,118)
      L(4)=L(4)+1
      IF(L(4).GT.1)GO TO 13
   14 IF(JCSENS.LT.0)WRITE(MD,118)
C                     EQUATION FILE
      L(3)=7
      L(4)=1
      F(2)=D(6)
      IF(JCSENS.EQ.1)GO TO 15
      WRITE(MC,139)
  139 FORMAT(14H EQUATION FILE)
      JCCCRE(13)=6
   15 IF((L(4).GT.F(2)).AND.(JCSENS.LE.0))GO TO 16
      CALL J1RAFE(L,2,JCSENS,F,4)
      IF(JCSENS.LT.0)WRITE(MD,118)
      L(4)=L(4)+1
      IF(L(4).GT.1)GO TO 15
   16 IF(JCSENS.LT.0)WRITE(MD,118)
C                     CONSTRAINT FILE
      L(3)=8
      L(4)=1
      F(2)=D(8)
      IF(JCSENS.EQ.1)GO TO 17
      WRITE(MC,140)
  140 FORMAT(16H CONSTRAINT FILE)
      JCCCRE(13)=6
   17 IF((L(4).GT.F(2)).AND.(JCSENS.LE.0))GO TO 18
      CALL J1RAFE(L,2,JCSENS,F,5)
      IF(JCSENS.LT.0)WRITE(MD,118)
      L(4)=L(4)+1
      IF(L(4).GT.1)GO TO 17
   18 IF(JCSENS.LT.0)WRITE(MD,118)
C                     END OF UNIT
      L(1)=4
      L(4)=1
      IF(JCSENS.LT.1)GO TO 9
      IF(D(2).LE.0)CALL SYSERR(1,U)
      IF((D(4)+D(6)+F(8)).LE.0)CALL SYSERR(2,U)
      GO TO 9
C                     END OF SECEDE
   19 IF(U.LE.1)CALL SYSERR(3,U)
   20 IF(JCSENS.EQ.1)GO TO 21
      WRITE(MC,141)
  141 FORMAT(14HOEND OF SECEDE)
      JCCCRE(13)=6
      IF(JCSENS.LT.0)WRITE(MD, 118)
C                     EXIT
   21 CALL SYSEXT
      RETURN
      END
```

```
C
C*****************************************************************
C                          **********
C                          * VARIAO *
C                          **********
C
C       PURPOSE
C           VARIAO HANDLES THE INPUT AND OUTPUT OF VARIABLE VALUES AND
C           NAMES WHEN INPUT OR OUTPUT OF THIS INFORMATION ONLY IS
C           DESIRED.
C
C       USAGE
C           CALL VARIAO(JCLENT,JCMODE,JCSENS,JCCLNO)
C
C       DATA FORMAT
C           SEE TEAR AND DECISION VARIABLE FILE INPUT IN SECIAO.
C
C       DESCRIPTION OF PARAMETERS
C           JCLENT = THE ADDRESS OF A GROUP. AN AUXILIARY LIST HEADCELL
C                    OR 0.  IF 0, THE SOURCE OR DESTINATION OF THE DATA
C                    IS SECEDE.  IF JCLENT IS THE ADDRESS OF A LIST
C                    HEADCELL, THE VARIABLES WHOSE VALUES ARE TO BE
C                    WRITTEN ARE SPECIFIED BY THE LIST ENTRIES.  IF
C                    JCLENT IS THE ADDRESS OF A GROUP, THE DECISION AND
C                    TEAR VARIABLE NAMES WILL BE WRITTEN.
C           JCMODE = -2, -1, OR N, WHERE N IS A NON-NEGATIVE INTEGER.
C                    IF JCMODE = -1, OUTPUT OF THE VARIABLE NAMES IN
C                    THE DECISION AND TEAR VARIABLE FILES OR LISTS
C                    (DEPENDING ON JCLENT) IS REQUESTED.  WHEN JCMODE
C                    IS NON-NEGATIVE THE OUTPUT OR INPUT WILL CONSIST
C                    OF VARIABLE VALUES.  THE SOURCE OR DESTINATION OF
C                    THE VALUES IS INDICATED BY JCLENT.  THE N IS THE
C                    UNIT NUMBER WHEN JCLENT = 0.  IF JCMODE = -2,
C                    VARIABLE VALUES FOR EVERY UNIT WILL BE INPUT OR
C                    OUTPUT.
C           JCSENS = SEE J1DIMS.  JCSENS CAN NOT EQUAL -1.
C           JCCLNO = A USER ASSIGNED CALL STATEMENT ID. PARAMETER
C           THE AUXILIARY LIST, WHEN USED, MUST CONTAIN LIST ENTRIES OF
C           THE SAME SIZE AND PATTERN OF THE GENDER TEAR AND DECISION
C           VARIABLE LISTS.
C
C       REMARKS
C           THE NUMBER REPRESENTED BY JCTCIA MUST, FOR SOME MACHINES,
C           REPLACE THE 1 IN THE NAMED COMMON STATEMENT.
C           JCSENS MUST BE 0 IF JCLENT IS POSITIVE OR JCMODE IS
C           NEGATIVE.
C           THE FOLLOWING TABLE SHOULD ASSIST IN EXPLAINING THE VARIAO
C           MODES OF OPERATION.  (A=AUXILIARY LIST HEADCELL, G=GROUP
C           AND N=NON-NEGATIVE INTEGER)
C           VARIAO CALL PARAMETERS
C           RESPONSE/SOURCE  SECEDE       GENDER      AUX. LIST
C           NAMES(OUTPUT)  (0,-1,0,N)  (G,-1,0,N)        -
C           VALUES(INPUT)  (0, N,1,N)        -           -
C           VALUES(OUTPUT) (0, N,0,N)        -       (A, 0,0,N)
C
C       SUBROUTINE AND FUNCTION SUBPROGRAMS REQUIRED
C           FETCH
C           FRMCEL
C           J1RAFE
C           J1RESO
C           J1VALU
C           J1VARI
```

```
C            J7ICOM
C            SYSENT
C            SYSEXT
C
C         ERROR CODES FOR THIS PROGRAM
C             NONE
C
C         METHOD
C             SELF-EXPLANATORY
C
C                                  **********
C
      SUBROUTINE VARIAO(JCLENT,JCMODE,JCSENS,JCCLNO)
      COMMON/ALLOC/JCCORE(1)
      DIMENSION L(5),F(3),P(4),V(4),D(2),A(7),Q(4)
      DOUBLE PRECISION VALUE
      INTEGER F,P,V,D,A,C
      DATA L/4,1,1,1,1/,F/4,1,0/,P/1,1,1,1/,V/1,7,2,3/,A/2,2,1,1,21,1,1/
     1 ,G/1,1,1,1/
C                  ENTER
      CALL SYSENT(1,4,1,3,JCCLNO)
      IF(JCSENS.EQ.1)GO TO 21
      M1=JCCORE(11)
      WRITE(M1,201)
  201 FORMAT(115H1THE FOLLOWING TABLE GIVES THE VARIOUS OUTPUT FIELDS WH
     1ICH MAY APPEAR AS OUTPUT.  EACH LINE OF OUTPUT BEGINS WITH A)
      WRITE(M1,202)
  202 FORMAT(118H LETTER DESIGNATION.  THIS LETTER CORRESPONDS TO ONE OF
     1 THE ENTRIES IN THIS TABLE, THUS IDENTIFYING THE VARIOUS FIELDS)
      WRITE(M1,203)
  203 FORMAT(30H IN THE LINE OF VARIAO OUTPUT.)
      WRITE(M1,204)
  204 FORMAT(38H A  VARIABLE CODE NAME, DIMENSIONALITY)
      WRITE(M1,205)
  205 FORMAT(107H B  MAPPING FLAG, MAPPING INDEX, OPERATOR FLAG, SCALE S
     1IGN FLAG, SCALE, OFFSET SIGN FLAG, OFFSET (OR INDEX))
      WRITE(M1,206)
  206 FORMAT(54H C  VARIABLE CODE NAME, DIMENSIONALITY, SELECTION FLAG)
      WRITE(M1,207)
  207 FORMAT(45H F  VARIABLE CODE NAME, DIMENSIONALITY, VALUE)
      WRITE(M1,208)
  208 FORMAT(30H G  INDEX, INDEX, INDEX, INDEX)
      WRITE(M1,209)
  209 FORMAT(9H1VARIABLE)
C                  MODE
   21 IF((JCLENT+JCMODE).NE.-1)GO TO 2
C                  DECISION AND TEAR VARIABLE FILES
      M1=JCCORE(11)
      WRITE(M1,101)
  101 FORMAT(23HODECISION VARIABLE FILE)
    1 CALL J1RAFE(L,1,0,F,1)
      L(3)=L(3)+1
      IF(L(3).GT.2)GO TO 14
      WRITE(M1,102)
  102 FORMAT(19HOTEAR VARIABLE FILE)
      GO TO 1
C                  MODE
    2 IF(JCLENT.GT.0)GO TO 5
C                  VALUE FILE
      M1=JCCORE(17)+1
      F(3)=JCCORE(M1)
      L(3)=3
```

```
      M1=JCMODE
      IF(M1.EQ.-1)M1=0
      L(2)=M1+4
      M3=JCCORE(15)+1
      M2=JCCORE(M3)
      M3=M3+1
      M3=JCCORE(M3)
    3 M4=1
      CALL FETCH(JCCORE(M2),L,M5,P,F,JCCORE(M3),M4,1)
      IF(M4.EQ.-1)GO TO 14
      IF(JCSENS.GT.0)GO TO 4
      M4=JCCORE(11)
      WRITE(M4,104)M1
  104 FORMAT(26HOVARIABLE VALUES FOR UNIT I5)
    4 CALL J1VALU(L,JCSENS,F,1)
      IF(M1.EQ.JCMODE)GO TO 14
      M1=M1+1
      GO TO 3
C                     MODE
    5 IF(JCMODE.NE.-1)GO TO 7
C                     LIST
      M1=JCCORE(16)+1
      M2=JCCORE(17)+8
      M1=JCCORE(M1)+5*JCCORE(M2)-5
C                     DECISION AND TEAR VARIABLE LISTS
      CALL FRMCEL(JCCORE(M1),JCLENT,C,V,1)
      M1=JCCORE(11)
      WRITE(M1,105)
  105 FORMAT(23HODECISION VARIABLE LIST)
      M2=1
    6 CALL J1VARI(0,D(M2),2,0,0,1)
      M2=M2+1
      IF(M2.NE.2)GO TO 14
      WRITE(M1,106)
  106 FORMAT(19HOTEAR VARIABLE LIST)
      GO TO 6
C                     AUXILIARY LIST
    7 M1=JCCORE(16)+1
      M2=JCCORE(17)+10
      M1=JCCORE(M1)+5*JCCORE(M2)-5
      M2=M2-4
      M2=JCCORE(M2)
      M3=JCCORE(17)+1
      Q(4)=JCCORE(M3)
      M0=JCCORE(11)
      WRITE(M0,107)
  107 FORMAT(16HOVARIABLE VALUES)
      M3=JCCORE(15)+2
      M4=JCCORE(M3)
      M3=M3-1
      M3=JCCORE(M3)
      M5=M1+3
      M5=JCCORE(M5)
      M6=JCCORE(16)+8
      M6=JCCORE(M6)
      M7=JCLENT
    8 CALL FRMCEL(JCCORE(M2),M7,C,A,2)
      IF(C(1).EQ.0)GO TO 10
      M8=M7+M5
      M9=4
    9 CALL J7ICOM(M1,M8,0,0,0,M9,D(1),0,M10,1)
      IF(M10.EQ.-1)GO TO 13
```

```
      IF(M9.EQ.-4)GO TO 11
   10 M10=D(2)/M2
      L(2)=M10+4
      L(1)=5-4/L(2)
      M11=L(1)
      L(4)=1
      L(M11)=D(2)-M10*M2
      CALL FETCH(JCCORE(M3),L,M11,P,L,JCCORE(M4),0,2)
      IF(D(1).EQ.0)GO TO 12
      M12=M11
   11 M11=J1RESO(M12,M6,1)
   12 CALL FRMCEL(JCCORE(M3),M11,VALUE,Q,3)
      WRITE(MO,108)D(2),D(1),VALUE
  108 FORMAT(4H F  2I29,E48.28)
      IF(D(1).EQ.0)GO TO 13
      M11=M3+D(1)-1
      WRITE(MO,109)(JCCORE(I),I=M6,M11)
  109 FORMAT(4H G  4I29)
      M9=-4
      GO TO 9
   13 M7=LNKFWD(JCCORE(M1),M7,1)
      IF(M7.GT.0)GO TO 8
   14 M1=JCCORE(11)
      WRITE(M1,110)
  110 FORMAT(13HOEND VARIABLE)
C                 EXIT
      CALL SYSEXT
      RETURN
      END
```

```
C
C*******************************************************************
C                              **********
C                              * J1ARBS *
C                              **********
C
C        PURPOSE
C            J1ARBS SETS UP THE DIMENSIONED VARIABLE ADDRESS RESOLUTION
C            BLOCKS (DIVAR8'S), PROVIDES SPACE ALLOCATION FOR VARIABLE
C            VALUE STORAGE AND ESTABLISHES THE ABSOLUTE LINKAGES BETWEEN
C            THE DECLARATION AND VALUE FILES.
C
C        USAGE
C            CALL J1ARBS(JCLIST,JCFINE,JCCLNO)
C
C        DATA FORMAT
C            N/A
C
C        DESCRIPTION OF PARAMETERS
C            SEE J1RAFE
C
C        REMARKS
C            THE NUMBER REPRESENTED BY JCTCIA MUST, FOR SOME MACHINES,
C            REPLACE THE 1 IN THE NAMED COMMON STATEMENT.
C
C        SUBROUTINE AND FUNCTION SUBPROGRAMS REQUIRED
C            FETCH
C            NEWCEL
C            STORE
C            SYSENT
C            SYSEXT
C            TOCELL
C
C        ERROR CODES FOR THIS PROGRAM
C            CODE FATAL  ERROR(DATA PROVIDED)
C            1    NF     COMMON VARIABLE WITHOUT LINKAGE (CODE NUMBER)
C
C        METHOD
C            SELF-EXPLANATORY
C
C                              **********
C
      SUBROUTINE J1ARBS(JCLIST,JCFINE,JCCLNO)
      COMMON/ALLOC/JCCORE(1)
      DIMENSION JCLIST(6),JCFINE(3),L(7),C(6),D(3),W(4),P(4),R(5),A(7)
      INTEGER C,D,W,P,R,A
      DATA L/5,0,1,1,1,1,0/,C/4,4,1,1,1,0/,W/1,1,1,1/,P/1,1,1,1/,R/1,1,
     1 1,1,-1/,A/2,17,1,1,18,1,1/
C                     ENTER
      CALL SYSENT(1,5,1,4,JCCLNO)
      JCLIST(5)=1
C                     MA=ADDRESS OF SPUR
      M1=JCCORE(15)+1
      MA=JCCORE(M1)
C                     MB=ADDRESS OF LOAD
      M1=M1+1
      MB=JCCORE(M1)
      MC=MB+5
      MD=MB+4
      ME=MB+26
      MF=MB+20
      MG=MB+30
```

```
      M2=MG+28
      JCCORE(M2)=-1
C                     L
      MO=5-4/JCLIST(2)
      L(1)=MO
      L(2)=JCLIST(2)
      L(4)=1
C                     MH=CONTROL PARAMETER
      MH=1-4/L(2)
C                     W(4)
      W(4)=2+MH
      MI=W(4)
C                     MJ=NUMBER OF VARIABLES
      MJ=JCFINE(2)
C                     MK=ADDRESS OF WORKING SPACE FOR INDICES
      M1=M1+1
      MK=JCCORE(M1)
C                     DO LOOP
      DO 7 I=1,MJ
C                     GET RELATIVE POINTER
      L(MO)=I
      CALL FETCH(JCCORE(MA),L,L(7),P,L,JCCORE(MB),0,1)
      IF(L(7).EQ.0)GO TO 7
C                     ACCESS DECLARATION
      CALL FETCH(JCCORE(MA),L(6),D,W,L,JCCORE(MB),0,2)
      IF(D(1).GT.0)GO TO 7
      MM=JCCORE(MO)
C                     COMMON DECLARATION
      IF((MH*D(2)).EQ.0)GO TO 1
      C(4)=D(2)
      CALL FETCH(JCCORE(MA),C,C(6),P,L,JCCORE(MG),0,3)
      CALL FETCH(JCCORE(MA),C(5),MN,P,L,JCCORE(MG),0,4)
      IF(MN.GT.0)GO TO 6
      CALL SYSERR(1,C(4))
C                     GET INDEX RANGES
    1 M1=2+MH
      L(7)=M1
      M1=D(M1)
      JCLIST(4)=I
      R(4)=M1
      IF(M1.GT.0)CALL FETCH(JCCORE(MA),L(6),JCCORE(MK),R,L,JCCORE(MB),0,
     1 5)
C                     ADDRESS STORAGE LOCATION
      CALL STORE(JCCORE(MA),JCLIST,D,R(5),JCFINE,JCCORE(MB),1)
      MN=JCCORE(MO)
      IF(M1.EQ.0)GO TO 6
C                     STORE RELATIVE POINTER
      D(1)=21
      D(2)=(JCCORE(MF)-JCCORE(ME))/JCFINE(3)
      CALL TOCELL(JCCORE(MA),MN,D,A,1)
C                     MULTIPLY RANGES
      M2=1
      M3=MK
      DO 2 J=1,M1
      M2=M2*JCCORE(M3)
    2 M3=M3+1
C                     SPACE REQUIRED
      M2=M2*JCFINE(3)
C                     LENGTH OF DIVARB
      M3=JCCORE(17)
      M3=JCCORE(M3)
      M1=M1+M2/M3+3
```

```
      M4=M1
      IF(M1.LE.(M3/2))GO TO 3
      M4=M3/2+1
      M5=MA+5*M4+3
      JCCORE(M5)=M1
C                     SPUR FOR DIVARB
    3 M5=MA+5*M4
C                     GET DIVARB
      CALL NEWCEL(JCCORE(M5),MN,1)
C                     STORE RANGES
      JCCORE(MN)=R(4)
      M4=MN+1
      M5=MK
      M6=R(4)
      DO 4 J=1,M6
      JCCORE(M4)=JCCORE(M5)
      M4=M4+1
    4 M5=M5+1
C                     STORE RECORD ADDRESSES
      M5=M4+1
      M6=MN+M1-1
      JCLIST(6)=D(2)
      DO 5 J=M5,M6
      CALL STORE(JCCORE(MA),JCLIST(5),D,R(5),JCFINE,JCCORE(MB),2)
      JCLIST(6)=M3/JCLIST(3)
    5 JCCORE(J)=JCCORE(MC)
C                     STORE OFFSET
      JCCORE(M4)=JCCORE(MD)-JCCORE(MC)
C                     ADDRESS LAST COMPONENT
      JCLIST(6)=(M2-(M6-M5)*M3+JCCORE(M4))/JCFINE(3)-1
      CALL STORE(JCCORE(MA),JCLIST(5),D,R(5),JCFINE,JCCORE(MB),3)
C                     STORE ADDRESS OF DIVARB
    6 CALL TOCELL(JCCORE(MA),MM,MN,P,2)
    7 CONTINUE
C                     EXIT
      CALL SYSEXT
      RETURN
      END
```

```
C
C*********************************************************************
C                                  **********
C                                  * J1CONS *
C                                  **********
C
C         PURPOSE
C             J1CONS HANDLES THE INPUT AND OUTPUT OF CONSTANT VALUE FILES
C
C         USAGE
C             CALL J1CONS(JCLIST,JCSENS,JCFINE,JCCLND)
C
C         DATA FORMAT
C             SEE SECEDE
C
C         DESCRIPTION OF PARAMETERS
C             SEE J1RAFE
C
C         REMARKS
C             THE NUMBER REPRESENTED BY JCTCIA MUST, FOR SOME MACHINES,
C             REPLACE THE 1 IN THE NAMED COMMON STATEMENT.
C
C         SUBROUTINE AND FUNCTION SUBPROGRAMS REQUIRED
C             FETCH
C             STORE
C             SYSENT
C             SYSEXT
C
C         ERROR CODES FOR THIS PROGRAM
C             NONE
C
C         METHOD
C             SELF-EXPLANATORY
C
C                                  **********
C
      SUBROUTINE J1CONS(JCLIST,JCSENS,JCFINE,JCCLND)
      COMMON/ALLOC/JCCORE(1)
      DIMENSION JCLIST(6),JCFINE(3),V(4)
      INTEGER V
      DATA V/1,1,1,1/
      DOUBLE PRECISION VALUE
C                 ENTER
      CALL SYSENT(1,5,1,5,JCCLND)
C                 MA=ADDRESS OF SPUR
      M1=JCCORE(15)+1
      MA=JCCORE(M1)
C                 MB=ADDRESS OF LOAD
      M1=M1+1
      MB=JCCORE(M1)
      MBA=MB+7
      MBB=MB+8
      M1=MB+21
      M2=MB+22
      M3=JCCORE(M1)+JCCORE(M2)+1
      M4=(M3/2)*2
      IF(((JCFINE(3)-1)*(M3-M4)*JCSENS).GT.0)JCCORE(M1)=JCCORE(M1)+1
C                 MC=INPUT DATA SET
      MC=JCCORE(10)
C                 MD AND ME ARE OUTPUT DATA SETS
      MD=JCCORE(11)
      ME=IABS(JCSENS)
```

```
C                    IN OR OUT
      MF=1
      JCLIST(5)=1
      JCLIST(6)=1
      V(4)=JCFINE(3)
      IF(JCSENS.LT.1)GO TO 2
C                    INPUT
    1 READ(MC,101)M1,M2,VALUE
  101 FORMAT(I1,I19,D40.28)
      IF(M1.EQ.0)GO TO 3
C                    STORE VALUE
      CALL STORE(JCCORE(MA),JCLIST(MF),VALUE,V,JCFINE,JCCORE(MB),1)
      MF=5
      GO TO 1
C                    OUTPUT
    2 M1=1
      CALL FETCH(JCCORE(MA),JCLIST(MF),VALUE,V,JCFINE,JCCORE(MB),M1,1)
      MF=5
      IF(M1.EQ.-1)GO TO 3
C                    WRITE OUTPUT
      WRITE(MD,102)JCCORE(MBA),VALUE
  102 FORMAT(4H J   I29,D48.28)
      JCCORE(13)=6
      IF(JCSENS.LT.0)WRITE(ME,101)M1,JCCORE(MBA),VALUE
C                    ITERATE
      IF(JCCORE(MBA).LT.JCCORE(MBB))GO TO 2
C                    EXIT
    3 CALL SYSEXT
      RETURN
      END
```

```
C
C*******************************************************************
C                              **********
C                              * J1DECL *
C                              **********
C
C       PURPOSE
C           J1DECL HANDLES THE INPUT AND OUTPUT OF VARIABLE, ER,
C           FUNCTION AND CONSTRAINT DECLARATIONS.
C
C       USAGE
C           CALL J1DECL(JCLIST,JCSENS,JCFINE,JCCLNO)
C
C       DATA FORMAT
C           SEE SECEDE
C
C       DESCRIPTION OF PARAMETERS
C           SEE J1RAFE
C
C       REMARKS
C           THE NUMBER REPRESENTED BY JCTCIA MUST, FOR SOME MACHINES,
C           REPLACE THE 1 IN THE NAMED COMMON STATEMENT.
C
C       SUBROUTINE AND FUNCTION SUBPROGRAMS REQUIRED
C           FETCH
C           J1DIMS
C           STORE
C           SYSENT
C           SYSEXT
C
C       ERROR CODES FOR THIS PROGRAM
C           NONE
C
C       METHOD
C           SELF-EXPLANATORY
C
C                              **********
C
      SUBROUTINE J1DECL(JCLIST,JCSENS,JCFINE,JCCLNO)
      COMMON/ALLOC/JCCORE(1)
      DIMENSION JCLIST(7),JCFINE(4),D(4),W(4),P(5)
      INTEGER P,D,W
      DATA W/1,1,1,1/,P/1,1,1,1,-1/,D(1)/1/
C                     ENTER
      CALL SYSENT(1,5,1,6,JCCLNO)
      D(1)=1
      JCLIST(6)=1
C                     MA=ADDRESS OF SPUR
      M1=JCCORE(15)+1
      MA=JCCORE(M1)
C                     MB=ADDRESS OF LOAD
      M1=M1+1
      MB=JCCORE(M1)
      MH=MB+4
      MC=MB+26
      MD=MB+20
C                     ME=INPUT DATA SET
      ME=JCCORE(10)
C                     MF AND MG ARE OUTPUT DATA SETS
      MF=JCCORE(11)
      MG=IABS(JCSENS)
C                     W
```

```
      M1=4/JCLIST(1)
      M2=1/JCLIST(4)
      W(3)=1+M1+(1-M1)*M2
      W(4)=W(3)+(1-M1)*M2
      MI=W(4)-W(3)+3
C                   IN OR OUT
      IF(JCSENS.LT.1)GO TO 2
C                   INPUT
      M1=JCLIST(1)-2
      M2=M1+2
C                   READ DECLARATION
    1 READ(ME,101)(D(I),I=1,MI)
  101 FORMAT(II,I19,2I20)
      IF(D(1).EQ.0) GO TO 6
C                   ACCESS RELATIVE POINTER
      CALL STORE(JCCORE(MA),JCLIST,D,P(5),JCFINE,JCCORE(MB),1)
C                   STORE DATA
      M3=JCCORE(MH)
      JCCORE(M3)=JCCORE(MD)-JCCORE(MC)
      JCLIST(7)=JCCORE(M3)
      CALL STORE(JCCORE(MA),JCLIST(6),D(3),W,JCFINE,JCCORE(MB),2)
      JCLIST(7)=W(4)
      IF(D(MI).GT.0)CALL J1DIMS(JCLIST(7),D(MI),1,0,1,1)
      JCLIST(7)=JCLIST(7)-1
      IF(JCLIST(7).GT.0)CALL STORE(JCCORE(MA),JCLIST(6),D,P(5),JCFINE,
     1 JCCORE(MB),3)
C                   INCREMENT
      JCLIST(M2)=JCLIST(M2)+1
      GO TO 1
C                   OUTPUT
    2 M1=JCLIST(1)
      M3=M1-2
C                   FETCH RELATIVE POINTER
    3 M2=1
      CALL FETCH(JCCORE(MA),JCLIST,JCLIST(7),P,JCFINE,JCCORE(MB),M2,2)
      IF((M2*JCLIST(7)).LE.0)GO TO 6
C                   FETCH DATA
      CALL FETCH(JCCORE(MA),JCLIST(6),D(3),W,JCFINE,JCCORE(MB),0,3)
      D(2)=JCLIST(M1)
C                   WRITE DATA
      IF(MI.EQ.4)GO TO 4
      WRITE(MF,102)(D(I),I=2,3)
  102 FORMAT(4H A  2I29)
      GO TO 5
    4 WRITE(MF,103)(D(I),I=2,4)
  103 FORMAT(4H I  3I29)
    5 IF(JCSENS.LT.0)WRITE(MG,101)(D(I),I=1,MI)
      JCLIST(7)=W(4)
      JCCORE(13)=6
      IF(D(MI).GT.0)CALL J1DIMS(JCLIST(7),D(MI),1,0,JCSENS,2)
C                   INCREMENT
      JCLIST(M1)=JCLIST(M1)+1
      IF(JCLIST(M1).LE.JCFINE(M3))GO TO 3
C                   EXIT
    6 CALL SYSEXT
      RETURN
      END
```

```
C
C*******************************************************************
C                              **********
C                              * J1DIMS*
C                              **********
C
C       PURPOSE
C           J1DIMS HANDLES THE INPUT AND OUTPUT OF INDICES.
C
C       USAGE
C           CALL J1DIMS(JCPOFP,JCDIMS,JCMODE,JCTYPE,JCSENS,JCCLNO)
C
C       DATA FORMAT
C           SEE SECEDE
C
C       DESCRIPTION OF PARAMETERS
C           JCPOFP = THE NUMBER OF WORDS TO ADVANCE TO REACH THE FIRST
C                    INDEX OR THE ADDRESS OF THE FIRST INDEX (SEE MODE)
C           JCDIMS = THE DIMENSIONALITY
C           JCMODE = 1 OR 2 AS RESPECTIVELY THE INDEX STORAGE MEDIUM IS
C                    SECEDE OR GENDER
C           JCTYPE = 0, 1 OR 2 AS RESPECTIVELY THE INDICES ARE WHOLE
C                    WORD, SINGLE SEVEN PART OR DOUBLE SEVEN PART
C           JCSENS = -N, 0 OR 1 AS RESPECTIVELY PUNCHED (AND WRITTEN)
C                    OUTPUT, WRITTEN ONLY OR INPUT IS DESIRED
C           JCCLNO = A USER ASSIGNED CALL STATEMENT ID. PARAMETER
C
C       REMARKS
C           THE NUMBER REPRESENTED BY JCTCIA MUST, FOR SOME MACHINES,
C           REPLACE THE 1 IN THE NAMED COMMON STATEMENT.
C
C       SUBROUTINE AND FUNCTION SUBPROGRAMS REQUIRED
C           FETCH
C           FRMCEL
C           STORE
C           SYSENT
C           SYSEXT
C           TOCELL
C
C       ERROR CODES FOR THIS PROGRAM
C           NONE
C
C       METHOD
C           SELF-EXPLANATORY
C
C                              **********
C
      SUBROUTINE J1DIMS(JCPOFP,JCDIMS,JCMODE,JCTYPE,JCSENS,JCCLNO)
      COMMON/ALLOC/JCCORE(1)
      DIMENSION L(2),W(4),X(22),D(7)
      INTEGER X,D,W
      DATA L(1)/1/,W/1,1,1,1/,X/7,9,1,1,10,1,1,11,1,1,12,1,1,13,1,1,14,
     1 1,1,15,1,1/
C                   ENTER
      CALL SYSENT(1,5,1,7,JCCLNO)
C                   MA=ADDRESS OF SPUR
      M1=JCCORE(15)+1
      MA=JCCORE(M1)
C                   MB=ADDRESS OF LOAD
      M1=M1+1
      MB=JCCORE(M1)
      L(2)=JCPOFP
```

```
C                   MC=INPUT DATA SET
      MC=JCCORE(10)
C                   MO ANC ME ARE OUTPUT DATA SETS
      MO=JCCORE(11)
      ME=IABS(JCSENS)
C                   TYPE OF INDEX
      IF(JCTYPE.GT.0)GO TO 4
C                   WHOLE WORD
      M1=JCDIMS
    1 M2=4
      IF(M1.LT.4)M2=M1
      M1=M1-M2
      W(4)=M2
      IF(JCSENS.LT.1)GO TO 2
C                   INPUT
      READ(MC,101)(D(I),I=1,M2)
  101 FORMAT(4I20)
      CALL STORE(JCCORE(MA),L,D,W,L,JCCORE(MB),1)
      GO TO 3
C                   OUTPUT
    2 CALL FETCH(JCCORE(MA),L,D,W,L,JCCORE(MB),0,1)
      IF(JCSENS.LT.0)WRITE(ME,101)(D(I),I=1,M2)
      WRITE(MO,102)(D(I),I=1,M2)
  102 FORMAT(4H E  4I29)
      JCCORE(13)=6
C                   REPEAT
    3 L(2)=M2
      IF(M1.GT.0)GO TO 1
      JCPOFP=M2
      GO TO 8
C                   SEVEN PART
    4 M1=JCDIMS*JCTYPE
    5 M1=M1-1
      IF(JCSENS.LT.1)GO TO 6
C                   INPUT
      READ(MC,103)(D(I),I=1,7)
  103 FORMAT(I5,I20,2I5,I20,I5,I20)
      IF(JCMODE.EQ.1)CALL STORE(JCCORE(MA),L,D,X,L,JCCORE(MB),2)
      IF(JCMODE.EQ.2)CALL TOCELL(JCCORE(MA),JCPOFP,D,X,1)
      GO TO 7
C                   OUTPUT
    6 IF(JCMODE.EQ.1)CALL FETCH(JCCORE(MA),L,D,X,L,JCCORE(MB),0,2)
      IF(JCMODE.EQ.2)CALL FRMCEL(JCCORE(MA),JCPOFP,D,X,1)
      IF(JCSENS.LT.0)WRITE(ME,103)(D(I),I=1,7)
      WRITE(MO,104)(D(I),I=1,7)
  104 FORMAT (4H B  I10,I25,2I10,I25,I10,I25)
      JCCORE(13)=6
C                   REPEAT
    7 L(2)=1
      IF(M1.GT.0)GO TO 5
      JCPOFP=1
C                   EXIT
    8 CALL SYSEXT
      RETURN
      END
```

```
C
C                                  **********
C                                  * J1FILE *
C                                  **********
C
C       PURPOSE
C           J1FILE ENTERS THE OCCURRENCE OF A LIST ENTRY ONTO THE
C           TEMPORARY FILE.
C
C       USAGE
C           CALL J1FILE(JCSPUR,JCLIST,JCSPRG,JCPOFP,JCPOSP,JCSENS,
C           JCFINE,JCLOAD,JCCLNO)
C
C       DATA FORMAT
C           N/A
C
C       DESCRIPTION OF PARAMETERS
C           JCSPUR = THE SPUR FOR THE FILE
C           JCLIST = THE LISTER VECTOR
C           JCSPRG = THE ADDRESS OF A GENDER LIST SPUR
C           JCPOFP = THE ADDRESS OF THE LIST ENTRY
C           JCPOSP = THE ADDRESS OF THE LIST ENTRY LAST WORD
C           JCSENS = SEE J1DIMS
C           JCFINE = THE FINE VECTOR
C           JCLOAD = THE LOAD VECTOR
C           JCCLNO = A USER ASSIGNED CALL STATEMENT ID. PARAMETER
C
C       REMARKS
C           THE NUMBER REPRESENTED BY JCTCIA MUST, FOR SOME MACHINES,
C           REPLACE THE 1 IN THE NAMED COMMON STATEMENT.
C
C       SUBROUTINE AND FUNCTION SUBPROGRAMS REQUIRED
C           NEWCEL
C           STORE
C           SYSENT
C           SYSEXT
C
C       ERROR CODES FOR THIS PROGRAM
C           NONE
C
C       METHOD
C           SELF-EXPLANATORY
C
C                                  **********
C
      SUBROUTINE J1FILE(JCSPUR,JCLIST,JCSPRG,JCPOFP,JCPOSP,JCSENS,
     1 JCFINE,JCLOAD,JCCLNO)
      COMMON/ALLOC/JCCORE(1)
      DIMENSION JCSPUR(5),JCLIST(3),JCFINE(2),JCLOAD(30),D(2),F(4)
      INTEGER D,F
      DATA D(2)/9/,F/1,1,1,2/
C                 ENTER
      CALL SYSENT(1,5,1,8,JCCLNO)
C                 GET LIST ENTRY
      IF(JCSENS.GT.0)CALL NEWCEL(JCCORE(JCSPRG),JCPOFP,1)
      M1=JCSPRG+3
      JCPOSP=JCCORE(M1)+JCPOFP-1
C                 ENTER ON FILE
      D(1)=JCPOFP
      CALL STORE(JCSPUR,JCLIST,D,F,JCFINE,JCLOAD,1)
C                 EXIT
      CALL SYSEXT
```

```
      RETURN
      END
```

```
C
C                                  **********
C                                  * J1GMBL *
C                                  **********
C
C       PURPOSE
C           J1GMBL SETS UP THE GIMBL.
C
C       USAGE
C           CALL J1GMBL(JCGIMB,JCSPBL,JCMAXB,JCCLNO)
C
C       DATA FORMAT
C           N/A
C
C       DESCRIPTION OF PARAMETERS
C           JCGIMB = THE ADDRESS OF THE GIMBL
C           JCSPBL = THE ADDRESS OF THE SPUR BLOCK
C           JCMAXB = THE MAXIMUM CORE BLOCK SERVICIBLE WITH THE SPUR
C                    BLOCK
C           JCCLNO = A USER ASSIGNED CALL STATEMENT ID. PARAMETER
C
C       REMARKS
C           THE NUMBER REPRESENTED BY JCTCIA MUST, FOR SOME MACHINES,
C           REPLACE THE 1 IN THE NAMED COMMON STATEMENT.
C
C       SUBROUTINE AND FUNCTION SUBPROGRAMS REQUIRED
C           NEWCEL
C           SYSENT
C           SYSEXT
C
C       ERROR CODES FOR THIS PROGRAM
C           NONE
C
C       METHOD
C           SELF-EXPLANATORY
C
C                                  **********
C
      SUBROUTINE J1GMBL(JCGIMB,JCSPBL,JCMAXB,JCCLNO)
      COMMON/ALLOC/JCCORE(1)
      DIMENSION S(5)
      INTEGER S
      DATA S/0,0,1,0,2/
C                 ENTER
      CALL SYSENT(1,5,1,9,JCCLNO)
C                   S
      S(1)=0
      S(2)=0
      M1=JCCORE(17)+8
      M2=JCCORE(M1)+2
      M1=M1+1
      M3=JCCORE(M1)
      M1=M1+5
      M3=M3+JCCORE(M1)*2
      IF(M3.GT.M2)M2=M3
      IF(M2.LT.10)M2=10
      JCMAXB=M2
      S(4)=5*M2+5
C                   SPUR BLOCK
      CALL NEWCEL(S,M2,1)
      M1=S(4)/5
      JCSPBL=M2+5
```

```
      DO 1 I=1,M1
      DO 1 J=1,5
      JCCORE(M2)=S(J)+(J/4)*(4/J)*(I-S(4)-1)
    1 M2=M2+1
C                    GIMBL
      M1=JCSPBL+45
      CALL NEWCEL(JCCORE(M1),JCGIMB,2)
      JCCORE(16)=JCGIMB
      M1=JCGIMB
      DO 2 I=1,10
      JCCORE(M1)=(I/2)*(2/I)*JCSPBL
    2 M1=M1+1
C                    INDEX STORAGE
      M1=JCCORE(17)+14
      IF(JCCORE(M1).EQ.0)GO TO 4
      M1=JCSPBL+5*(JCCORE(M1)-1)
      M2=JCGIMB+3
      M3=M2+6
      DO 3 I=M2,M3
    3 CALL NEWCEL(JCCORE(M1),JCCORE(I),3)
C                    EXIT
    4 CALL SYSEXT
      RETURN
      END
```

```
C
C*****************************************************************
C                              **********
C                              * J1LEDL *
C                              **********
C
C       PURPOSE
C           J1LEDL HANDLES THE INPUT AND OUTPUT OF LINK EDITOR LISTS.
C
C       USAGE
C           CALL J1LEDL(JCSPUR,JCFIRS,JCMTHD,JCSENS,JCERFC,JCCLNO)
C
C       DATA FORMAT
C           N/A
C
C       DESCRIPTION OF PARAMETERS
C           JCSPUR = THE SPUR FOR THE TEMPORARY LISTS
C           JCFIRS = THE ADDRESS OF THE FIRST LIST ENTRY
C           JCMTHD = THE METHOD TEMPORARY LIST
C           JCSENS = SEE J1OIMS.  JCSENS MAY = +M, WHERE M IS AN ER,
C                    EQUATION OR CONSTRAINT CODE NAME, WITH THE DATA
C                    SOURCE BEING SECEDE.
C           JCERFC = THE GROUP BODY IDENTIFICATION FLAG
C           JCCLNO = A USER ASSIGNED CALL STATEMENT ID. PARAMETER
C
C       REMARKS
C           THE NUMBER REPRESENTED BY JCTCIA MUST, FOR SOME MACHINES,
C           REPLACE THE 1 IN THE NAMED COMMON STATEMENT.
C
C       SUBROUTINE AND FUNCTION SUBPROGRAMS REQUIRED
C           FETCH
C           FRMCEL
C           FSTLNK
C           J1DIMS
C           J7COMV
C           LNK8WD
C           NEWCEL
C           PUSH
C           STORE
C           SYSENT
C           SYSERR
C           SYSEXT
C           TOCELL
C
C       ERROR CODES FOR THIS PROGRAM
C           CODE FATAL  ERROR(DATA PROVIDED)
C           1    NF     ILLEGAL ENTRY ON LINK EDITOR LIST (ID. FLAG)
C
C       METHOD
C           SELF-EXPLANATORY
C
C                              **********
C
      SUBROUTINE J1LEDL(JCSPUR,JCFIRS,JCMTHD,JCSENS,JCERFC,JCCLNO)
      COMMON/ALLOC/JCCORE(1)
      DOUBLE PRECISION VALUE
      DIMENSION JCSPUR(5),D(8),E(16),T(7),P(5),L(9),C(17),X(4),V(4)
      INTEGER D,E,T,P,C,X,V
      DATA D(1)/0/,E/5,1,1,1,2,1,1,3,1,1,21,1,1,22,1,1/,T/2,21,1,
     1 1,22,1,1/,P/1,1,1,1,-1/,C/2,1,1,1,5,1,1,2,8,1,1,19,1,1,16,1,1/,
     2 L/0,0,0,0,0,1,1,1,0/,V/1,23,1,1/
C                 ENTER
```

```
      CALL SYSENT(1,5,1,10,JCCLNO)
C                     MA=VARIABLE SIZE WITHOUT INDICES (LINK ED.)
      M1=JCCORE(17)+11
      MA=JCCORE(M1)
C                     MB=ADDRESS OF SPUR FOR VARIABLE OR EQUATION VALUE
      M2=JCCORE(16)+1
      M2=JCCORE(M2)
      MB=M2+5*(MA-1)
C                     MC=ADDRESS OF SPUR FOR CONSTANT AND OPERATOR
      M1=M1-1
      MC=M2+5*(JCCORE(M1)-1)
C                     IN OR OUT
      IF(JCSENS.LT.1)GO TO 13
C                     ME=SCALE FACTOR FOR UNIT NUMBER
      M1=M1-4
      ME=JCCORE(M1)
C                     MEB=LENGTH OF CONSTANT AND OPERATOR
      M1=M1-2
      MEB=JCCORE(M1)
C                     MEA=LENGTH OF SCALAR VARIABLE
      M1=M1-1
      MEA=JCCORE(M1)
C                     MF=ADDRESS OF SPUR FOR VALUE STORAGE LOCATION
      M1=M1-2
      MF=M2+5*(JCCORE(M1)-1)
C                     MG=ADDRESS OF SPUR FOR SECEDE
      M1=JCCORE(15)+1
      MG=JCCORE(M1)
C                     MH AND MI ARE LOAD ADDRESSES
      M1=M1+1
      MH=JCCORE(M1)
      M1=MH+28
      JCCORE(M1)=-1
      MI=M1+2
C                     MJ=INPUT DATA SET
      MJ=JCCORE(10)
C                     MK=LAST LIST ENTRY
      JCFIRS=0
      MK=0
C                     ML=ADDRESS OF SPUR FOR LOCAL CONSTANT
      M1=JCCORE(17)+1
      V(4)=JCCORE(M1)
      M1=M1+6
      M2=JCCORE(16)+1
      ML=JCCORE(M2)+5*(JCCORE(M1)+V(4))-5
C                     READ OR FETCH
      IF(JCSENS.GT.1)GO TO 2
C                     READ
    1 READ(MJ,101)M1,(D(I),I=3,6)
  101 FORMAT(I1,I19,3I20)
      IF(M1.EQ.0)GO TO 21
      M3=1
      GO TO 4
C                     FETCH
    2 L(1)=5
      L(2)=JCSENS/ME
      L(3)=6+(JGERFC+1)/3
      L(4)=JCSENS-ME*L(2)
      L(2)=L(2)+4
      L(5)=1
C                     ACCESS ER,EQUATION OR CONSTRAINT
      CALL FETCH(JCCORE(MG),L,L(7),P,L,JCCORE(MH),0,1)
```

```
C                    ACCESS CONSTITUENT
    3 CALL FETCH(JCCORE(MG),L(6),D(2),C,L,JCCORE(MH),0,2)
      M1=D(2)
      M3=2
C                    TYPE
    4 M2=D(3)+1
      GO TO (5,6,8,6,9,82,20,20),M2
C                    OPERATOR
    5 IF(JCSENS.GT.1)CALL FETCH(JCCORE(MG),L(8),D(4),C(14),L,JCCORE(MH),
     1 0,3)
      D(2)=D(4)
      D(4)=D(5)
      D(5)=0
      M2=MC
      L(7)=ME8
      E(1)=4
      GO TO 10
C                    VARIABLE
    6 IF(JCSENS.EQ.1)GO TO 61
      CALL FETCH(JCCORE(MG),L(8),D(4),C(8),L,JCCORE(MH),0,4)
      D(4)=D(4)+ME*(L(2)-4)
   61 M2=MB+5*D(5)
      E(1)=5
      CALL J7COMV(MG,D(4),MI,1)
C                    NEW VALUE
      L(2)=D(4)/ME
      L(1)=5-1/(L(2)+1)
      M4=L(1)
      L(3)=1
      L(4)=1
      L(M4)=D(4)-ME*L(2)
      L(2)=L(2)+4
      CALL FETCH(JCCORE(MG),L,L(7),P,L,JCCORE(MI),0,5)
      CALL FETCH(JCCORE(MG),L(6),D(2),P,L,JCCORE(MI),0,51)
      L(7)=MEA
      GO TO 10
C                    RETRIEVE ADDRESS
    7 CALL FRMCEL(JCSPUR,M3,D(2),E(13),1)
      GO TO 10
C                    CONSTANT
    8 IF(JCSENS.EQ.1)GO TO 81
      C(17)=1
      CALL FETCH(JCCORE(MG),L(8),D(4),C(14),L,JCCORE(MH),0,6)
      C(17)=2
      D(4)=D(4)+ME*(L(2)-4)
   81 D(5)=0
      L(7)=ME8
      M2=MC
      E(1)=4
C                    STORAGE LOCATION
      L(1)=4
      L(2)=D(4)/ME
      L(3)=2
      L(4)=D(4)-ME*L(2)
      L(2)=L(2)+4
      CALL FETCH(JCCORE(MG),L,M3,P(5),L,JCCORE(MI),0,7)
      M3=MI+4
      D(2)=JCCORE(M3)
      GO TO 10
C                    LOCAL CONSTANT
   82 M2=ML
      E(1)=3
```

```
      D(2)=0
      GO TO 10
   83 READ(MJ,1010)VALUE
 1010 FORMAT(E40.28)
      CALL TOCELL(JCCORE(ML),M3,VALUE,V,11)
      GO TO 1
C                     VALUE OF ER, EQUATION OR CONSTRAINT
    9 M2=M8+5*D(5)
      L(7)=MEA
      E(1)=5
C                     GET STORAGE LOCATION
      CALL NEWCEL(JCCORE(MF),D(2),2)
      CALL PUSH(JCSPUR,JCMTHD,D(2),E(13),1,1)
C                     GET LIST ENTRY
   10 CALL NEWCEL(JCCORE(M2),M3,3)
C                     SET LINK
      IF(JCFIRS.EQ.0)JCFIRS=M3
      IF(MK.GT.0)CALL FSTLNK(JCCORE(M2),MK,M3,1)
      MK=M3
C                     STORE DATA
      CALL TOCELL(JCCORE(M2),M3,D,E,2)
      IF(D(3).EQ.5)GO TO 83
      IF(JCSENS.GT.1)GO TO 11
C                     READ INDICES
      IF(D(5).EQ.0)GO TO 1
      M3=M3+MA
      CALL J1DIMS(M3,D(5),2,1,1,1)
      IF(M2.NE.5)GO TO 1
C                     SPECIAL PROVISIONS REQUIRED IF FUNCTION IS TO BE
C                     PERMITTED AN INDEX RANGE.
      GO TO 1
C                     FETCH INDICES
   11 IF(D(5).EQ.0)GO TO 12
      M3=M3+MA
      X(4)=D(5)
      CALL FETCH(JCCORE(MG),L(6),JCCORE(M3),X,L,JCCORE(MH),0,8)
      L(7)=D(5)
   12 IF(M1.GT.0)GO TO 3
      GO TO 21
C                     OUTPUT
C                     ME=CURRENT LIST ENTRY
   13 ME=JCFIRS
C                     MF AND MG ARE OUTPUT DATA SETS
      MF=JCCORE(11)
      MG=IABS(JCSENS)
C                     GET DATA
   14 IF(ME.EQ.0)GO TO 21
      CALL FRMCEL(JCCORE(M8),ME,D,E,2)
      M1=ME+MA
      ME=D(1)
C                     TEST FLAG
      M2=D(3)+1
      M3=3
      GO TO (15,16,17,16,18,171,20,20),M2
C                     OPERATOR
   15 D(5)=D(4)
      D(4)=D(2)
      M2=4
      GO TO 19
C                     VARIABLE
   16 M2=5
      GO TO 19
```

```
C                       CONSTANT
   17 M2=4
      D(5)=0
      GO TO 19
C                       LOCAL CONSTANT
  171 D(2)=1
      CALL FRMCEL(JCCORE(ML),ME,VALUE,V,3)
      WRITE(MF,1020)VALUE
 1020 FORMAT(15H LOCAL CONSTANT,E40.28)
      JCCORE(13)=6
      IF(JCSENS.GE.0)GO TO 14
      WRITE(MG,101)D(2),D(3)
      WRITE(MG,1010)VALUE
      GO TO 14
C                       VALUE OF ER, FUNCTION OR CONSTRAINT
   18 M2=5
C                       WRITE DATA
   19 D(2)=1
      WRITE(MF,102)(D(I),I=3,M2)
  102 FORMAT(4H E  4I29)
      IF(JCSENS.LT.0)WRITE(MG,101)(D(I),I=2,M2)
      JCCORE(13)=6
      IF(D(3).EQ.0)GO TO 14
      IF(D(5).GT.0)CALL J1DIMS (M1,D(5),2,1,JCSENS,2)
      GO TO 14
C                       ERROR
   20 CALL SYSERR(1,(M2-1))
      GO TO (1,2,14),M3
C                       EXIT
   21 CALL SYSEXT
      RETURN
      END
```

```
C
C*****************************************************************
C                              **********
C                              * J1NAME *
C                              **********
C
C       PURPOSE
C           J1NAME HANDLES THE INPUT AND OUTPUT OF NAME FILES.
C
C       USAGE
C           CALL J1NAME(JCLIST,JCSENS,JCFINE,JCCLNO)
C
C       DATA FORMAT
C           SEE SECECE
C
C       DESCRIPTION OF PARAMETERS
C           SEE J1RAFE
C
C       REMARKS
C           THE NUMBER REPRESENTED BY JCTCIA MUST BE SUPPLIED IN THE
C           COMMON STATEMENT FOR SOME MACHINES.
C
C       SUBROUTINE AND FUNCTION SUBPROGRAMS REQUIRED
C           FETCH
C           STORE
C           SYSENT
C           SYSEXT
C
C       ERROR CODES FOR THIS SUBPROGRAM
C           NONE
C
C       METHOD
C           SELF-EXPLANATORY
C
C                              **********
C
      SUBROUTINE J1NAME(JCLIST,JCSENS,JCFINE,JCCLNO)
      COMMON/ALLOC/JCCORE(1)
      DIMENSION JCLIST(6),JCFINE(3),N(4),O(5)
      INTEGER O
      DATA N/1,1,1,1/,O(1)/1/
C                    ENTER
      CALL SYSENT(1,5,1,11,JCCLNO)
      JCLIST(5)=1
      JCLIST(6)=1
      M1=JCLIST(1)-1
      N(4)=JCFINE(M1)
C                    MA=ADDRESS OF SPUR
      M1=JCCORE(15)+1
      MA=JCCORE(M1)
C                    MB=ADDRESS OF LOAD
      M1=M1+1
      MB=JCCORE(M1)
C                    MC=INPUT DATA SET
      MC=JCCORE(10)
C                    MD AND ME ARE OUTPUT DATA SETS
      MD=JCCORE(11)
      ME=IABS(JCSENS)
C                    IN OR OUT
      MG=N(4)-1
      O(1)=1
      MH=MG+2
```

```
      MF=1
      IF(JCSENS.LT.1)GO TO 5
C                     INPUT
    1 GO TO (2,3),MG
    2 READ(MC,101)(D(I),I=1,4)
  101 FORMAT(II,I19,I14,A6)
      GO TO 4
    3 READ(MC,102)(D(I),I=1,5)
  102 FORMAT(I1,I19,I14,A4,A2)
    4 IF(D(1).EQ.0)GO TO 8
      D(3)=D(2)
      CALL STCRE(JCCORE(MA),JCLIST(MF),D(3),N,JCFINE,JCCORE(MB),1)
      MF=5
      GO TO 1
C                     OUTPUT
    5 M1=1
      CALL FETCH(JCCORE(MA),JCLIST(MF),D(2),N,JCFINE,JCCORE(MB),M1,1)
      IF((M1.EQ.-1).OR.((D(3)+D(MH)).EQ.0))GO TO 8
      MF=5
      GO TO (6,7),MG
    6 WRITE(MD,103)(D(I),I=2,3)
  103 FORMAT(4H C   I19,14H                A6)
      IF(JCSENS.LT.0)WRITE(ME,101)(D(I),I=1,3)
      JCCORE(13)=6
      GO TO 5
    7 WRITE(MD,104)(D(I),I=2,4)
  104 FORMAT(4H C   I19,14H                A4,A2)
      IF(JCSENS.LT.0)WRITE(ME,102)(D(I),I=1,4)
      JCCORE(13)=6
      GO TO 5
C                     EXIT
    8 CALL SYSEXT
      RETURN
      END
```

```
C
C*******************************************************************
C                              **********
C                              * J1RAFE *
C                              **********
C
C       PURPOSE
C           J1RAFE HANDLES THE INPUT AND OUTPUT OF FILE ENTRIES
C           CONTAINING RELATIVE POINTERS.
C
C       USAGE
C           CALL J1RAFE(JCLIST,JCMODE,JCSENS,JCFINE,JCCLNO)
C
C       DATA FORMAT
C           SEE SECEDE
C
C       DESCRIPTION OF PARAMETERS
C           JCLIST = THE LISTER VECTOR
C           JCMODE = 1 IF THE FILE IS THE TEAR OR DECISION VARIABLE
C                    FILE, 2 IF THE FILE IS A FUNCTION, ER OR
C                    CONSTRAINT FILE, OR 3 IF THE ENTRIES ARE INCIDENCE
C                    FILE ENTRIES
C           JCSENS = SEE J1OIMS
C           JCFINE = THE FINE VECTOR
C           JCCLNO = A USER ASSIGNED CALL STATEMENT ID. PARAMETER
C
C       REMARKS
C           THE NUMBER REPRESENTED BY JCTCIA MUST BE SUPPLIED IN THE
C           COMMON STATEMENT FOR SOME MACHINES.
C
C       SUBROUTINE AND FUNCTION SUBPROGRAMS REQUIRED
C           FETCH
C           J1OIMS
C           STORE
C           SYSENT
C           SYSEXT
C
C       ERROR CODES FOR THIS SUBPROGRAM
C           NONE
C
C       METHOD
C           SELF-EXPLANATORY.
C
C                              **********
C
      SUBROUTINE J1RAFE(JCLIST,JCMODE,JCSENS,JCFINE,JCCLNO)
      COMMON/ALLOC/JCCORE(1)
      DIMENSION JCLIST(7),C(4),O(29),D(5),JCFINE(4)
      INTEGER C,O,D
      DATA C/1,1,1,3/,O/3,1,1,1,16,1,1,5,1,1,16,2,2,5,1,1,1,8,1,1,5,1,1,
     1 7,1,1,19,1,1/
C                     ENTER
      CALL SYSENT(1,5,1,12,JCCLNO)
      C(1)=1
      JCLIST(6)=1
C                     MA=ADDRESS OF SPUR
      M1=JCCCRE(15)+1
      MA=JCCORE(M1)
C                     MB=ADDRESS OF LOAD
      M1=M1+1
      MB=JCCCRE(M1)
      MG=MB+4
```

```
C                   MC=INPUT DATA SET
      MC=JCCORE(10)
C                   MD ANC ME ARE OUTPUT DATA SETS
      MD=JCCORE(11)
      ME=IABS(JCSENS)
C                   MODE
      MF=1
      IF(JCMODE.GT.2)GO TO 5
C                   IN OR OUT
      IF(JCSENS.LT.1)GO TO 3
C                   INPUT
    1 IF(JCLIST(1).EQ.5)GO TO 11
      READ(MC,101)(D(I),I=1,3)
  101 FORMAT(11,I19,2120)
      GO TO 12
   11 READ(MC,101)D(1),M2,(D(I),I=2,3)
   12 IF(D(1).GT.0)GO TO 2
      IF(JCMODE.GT.1)JCLIST(4)=0
      IF(MF.EQ.1)GO TO 10
      GO TO 9
    2 D(1)=3+2*D(3)
      CALL STORE(JCCORE(MA),JCLIST(MF),D,C,JCFINE,JCCORE(MB),1)
      JCLIST(7)=3
      M1=JCCORE(MG)
      IF(D(3).GT.0)CALL J1DIMS(JCLIST(7),D(3),1,2,1,1)
      MF=6
      GO TO (1,5),JCMODE
C                   OUTPUT
    3 M1=1
      CALL FETCH(JCCORE(MA),JCLIST(MF),D,C,JCFINE,JCCORE(MB),M1,1)
      IF(M1.GE.0)GO TO 31
      IF(JCMODE.GT.1)JCLIST(4)=JCFINE(2)
      GO TO 10
   31 M3=D(1)
      MF=6
      D(1)=1
      IF(JCLIST(1).EQ.5)GO TO 4
      WRITE(MC,102)(D(I),I=2,3)
  102 FORMAT(4H A  2I29)
      IF(JCSENS.LT.0)WRITE(ME,101)(D(I),I=1,3)
      GO TO 41
    4 WRITE(MC,103)JCLIST(4),(D(I),I=2,3)
  103 FORMAT(4H K  3I29)
      IF(JCSENS.LT.0)WRITE(ME,101)D(1),JCLIST(4),(D(I),I=2,3)
   41 JCLIST(7)=3
      JCCORE(13)=6
      IF(D(3).GT.0)CALL J1DIMS(JCLIST(7),D(3),1,2,JCSENS,2)
      IF(M3.EQ.0)GO TO 10
      GO TO (3,5),JCMODE
C                   IN OR OUT
    5 M2=JCCORE(17)+4
      MH=JCCORE(M2)
      M2=M2-1
      MI=JCCORE(M2)-MH
      IF(JCSENS.LT.1)GO TO 8
C                   INPUT
    6 READ(MC,104)(D(I),I=1,5)
  104 FORMAT(I1,I19,I1,I19,I20)
      IF(D(1).GT.0)GO TO 7
      IF(MF.EQ.1)GO TO 10
      GO TO 9
    7 M2=1
```

```
      IF(C(3).EQ.1)M2=14
      O(1)=3
      IF(C(3).EQ.0)O(1)=4
      D(1)=MH+(M2/14)*(MI+2*D(5))+O(1)-3
      CALL STORE(JCCORE(MA),JCLIST(MF),D,O(M2),JCFINE,JCCORE(MB),2)
      JCLIST(7)=MH+(M2/14)*MI+O(1)-3
      MI=JCCORE(MG)
      IF(D(5).GT.0)CALL J1DIMS(JCLIST(7),O(5),1,1,1,3)
      MF=6
      GO TO 6
C                    OUTPUT
    8 M1=1
      CALL FETCH(JCCORE(MA),JCLIST(MF),D,O,JCFINE,JCCORE(MB),M1,2)
      IF(M1.EQ.-1)GO TO 10
      M4=C(1)
      MF=6
      D(1)=1
      M2=1
      IF(C(3).EQ.1)M2=14
      O(1)=3
      IF(C(3).EQ.0)O(1)=4
      JCLIST(7)=0
      IF(M2.EQ.14)CALL FETCH(JCCORE(MA),JCLIST(6),D,O(14),JCFINE,
     1 JCCORE(MB),O,3)
      M3=3+2*(M2/14)
      WRITE(MD,105)(O(I),I=2,M3)
  105 FORMAT(4H L  I29,I14,2I29)
      JCCORE(13)=6
      IF(JCSENS.LT.0)WRITE(ME,104)(O(I),I=1,M3)
      JCLIST(7)=MH+(M2/14)*MI+O(1)-3
      IF((M2-1)*C(5)).GT.0)CALL J1DIMS(JCLIST(7),O(5),1,1,JCSENS,4)
      IF(M4.GT.0) GO TO 8
      GO TO 10
C                    ACCESS LAST WORD OF LAST FILE ENTRY
    9 JCLIST(7)=JCLIST(7)-1
      JCCORE(M1)=0
      IF(JCLIST(7).EQ.0)GO TO 10
      C(1)=-1
      CALL STORE(JCCORE(MA),JCLIST(6),C,C,C,JCCORE(MB),3)
C                    EXIT
   10 CALL SYSEXT
      RETURN
      END
```

```
C
C*********************************************************************
C                              **********
C                              * J1RESO *
C                              **********
C       PURPOSE
C           J1RESO PERFORMS VARIABLE VALUE ADDRESS RSOLUTION
C
C       USAGE
C           J1RESO(JCARBL,JCINDC,JCCLNO)
C
C       DATA FORMAT
C           N/A
C
C       DESCRIPTION OF PARAMETERS
C           JCARBL = THE ADDRESS OF A DIVARB
C           JCINDC = THE ADDRESS OF AN INDEX SET
C           JCCLNO = A USER ASSIGNED CALL STATEMENT ID. PARAMETER
C
C       REMARKS
C           THE NUMBER REPRESENTED BY JCTCIA MUST, FOR SOME MACHINES,
C           REPLACE THE 1 IN THE NAMED COMMON STATEMENT.
C
C       SUBROUTINE AND FUNCTION SUBPROGRAMS REQUIRED
C           NONE
C
C       ERROR CODES FOR THIS PROGRAM
C           NONE
C
C       METHOD
C           SELF-EXPLANATORY
C
C                              **********
C
      FUNCTION J1RESO(JCARBL,JCINDC,JCCLNO)
      COMMON/ALLOC/JCCORE(1)
C                     ENTER
      CALL SYSENT(1,5,1,13,JCCLNO)
C                     MA=DIMENSIONALITY-1
      MA=JCCORE(JCARBL)-1
C                     COMPONENT
      M1=JCARBL+MA
      M2=JCINDC+MA-1
      M3=M2+1
      M3=JCCORE(M3)
      IF(MA.LE.0)GO TO 2
      DO 1 I=1,MA
      M3=M3+JCCORE(M1)*(JCCORE(M2)-1)
      M1=M1-1
    1 M2=M2-1
C                     MA=ADDRESS OF OFFSET
    2 MA=MA+2+JCARBL
C                     MB=WORDS PER VALUE
      M1=JCCORE(17)+1
      MB=JCCORE(M1)
C                     WORD
      M1=JCCORE(MA)+MB*(M3-1)
C                     SEGMENT
      M2=JCCORE(17)
      M3=M1/M2
      M4=MA+M3+1
      M4=JCCORE(M4)
```

```
C                    J1RESO
      J1RESO=M4+M1-M3*M2
C                    EXIT
      CALL SYSEXT
      RETURN
      END
```

```
C
C*********************************************************************
C                              **********
C                              * J1VALU *
C                              **********
C
C       PURPOSE
C           J1VALU HANDLES THE INPUT AND OUTPUT OF VARIABLE VALUE FILES
C
C       USAGE
C           CALL J1VALU(JCLIST,JCSENS,JCFINE,JCCLNO)
C
C       DATA FORMAT
C           SEE SECEDE
C
C       DESCRIPTION OF PARAMETERS
C           SEE J1RAFE
C
C       REMARKS
C           THE NUMBER REPRESENTED BY JCTCIA MUST, FOR SOME MACHINES,
C           REPLACE THE 1 IN THE NAMED COMMON STATEMENT.
C       SUBROUTINE AND FUNCTION SUBPROGRAMS REQUIRED
C           FETCH
C           FRMCEL
C           J1ARBS
C           J1RESO
C           STORE
C           SYSENT
C           SYSERR
C           SYSEXT
C           TOCELL
C
C       ERROR CODES FOR THIS PROGRAM
C           CODE FATAL  ERROR(DATA PROVIDED)
C           1    F      VARIABLE NOT DECLARED (VARIABLE CODE NAME)
C           2    F      DIMENSIONALITY MISMATCH (VARIABLE CODE NAME)
C
C       METHOD
C           SELF-EXPLANATORY
C
C                              **********
C
      SUBROUTINE J1VALU(JCLIST,JCSENS,JCFINE,JCCLNO)
      COMMON/ALLOC/JCCORE(1)
      DIMENSION JCLIST(6),JCFINE(3),D(6),V(4),A(4),L(7),P(4),R(4)
      INTEGER D,V,A,P,R
      DATA L/0,0,1,1,1,1,0/,V/1,1,1,1/,A/1,1,1,3/,P/1,1,1,1/,R/1,1,1,1/
      DOUBLE PRECISION VALUE
C                 ENTER
      CALL SYSENT(1,5,1,14,JCCLNO)
C                 MA=ADDRESS OF SPUR
      M1=JCCORE(15)+1
      MA=JCCORE(M1)
C                 MB=ADDRESS OF LOAD
      M1=M1+1
      MB=JCCORE(M1)
C                 MC=ADDRESS OF INDEX STORAGE
      M1=M1+1
      MC=JCCORE(M1)
C                 L
      L(1)=5-4/JCLIST(2)
      L(2)=JCLIST(2)
```

```
      L(4)=1
      L(5)=1
C                     MD=INPUT DATA SET
      MD=JCCORE(10)
C                     ME AND MF ARE OUTPUT DATA SETS
      ME=JCCORE(11)
      MF=IABS(JCSENS)
C                     V(4)
      V(4)=JCFINE(3)
      M1=MB+21
      M2=MB+22
      M3=JCCORE(M1)+JCCORE(M2)+1
      M4=(M3/2)*2
      IF(((V(4)-1)*(M3-M4)*JCSENS).GT.0)JCCORE(M1)=JCCORE(M1)+1
      M2=MB+28
      JCCORE(M2)=-1
C                     R(4)
      R(4)=L(1)-2
      MG=R(4)+3
C                     IN OR OUT
      D(1)=1
      IF(JCSENS.LT.1)GO TO 3
C                     INPUT
      CALL J1ARBS(JCLIST,JCFINE,1)
C                     READ
    1 READ(MD,101)(D(I),I=1,3),VALUE
  101 FORMAT(I1,I19,I20,D40.28)
      IF(D(1).EQ.0)GO TO 10
C                     INDICES
      IF(D(3).EQ.0)GO TO 2
      M1=MC+D(3)-1
      READ(MD,102)(JCCORE(I),I=MC,M1)
  102 FORMAT(4I20)
C                     DECLARATION
    2 M1=L(1)
      L(M1)=D(2)
      CALL FETCH(JCCORE(MA),L,L(7),P,L,JCCORE(MB),0,1)
      IF(L(7).LE.0)CALL SYSERR(-1,D(2))
      CALL FETCH(JCCORE(MA),L(6),D(4),R,L,JCCORE(MB),0,2)
      IF(D(3).NE.D(MG))CALL SYSERR(-2,D(2))
C                     DIMENSIONED
      IF(D(3).GT.0)D(4)=J1RESD(D(4),MC,1)
C                     STORE VALUE
      CALL TOCELL(JCCORE(MA),D(4),VALUE,V,1)
      GO TO 1
C                     OUTPUT
C                     ACCESS DECLARATION
    3 M1=1
      CALL FETCH(JCCORE(MA),L,L(7),P,L,JCCORE(MB),M1,3)
      IF(M1.EQ.-1)GO TO 10
      IF(L(7).LE.0)GO TO 9
      CALL FETCH(JCCORE(MA),L(6),D(4),R,L,JCCORE(MB),0,4)
C                     D(2) AND D(3)
      M1=L(1)
      D(2)=L(M1)
      D(3)=D(MG)
C                     DIMENSIONED
      M1=0
      IF(D(3).EQ.0)GO TO 6
      M2=MC+D(3)-1
C                     SET INDICES TO 1
      DO 4 I=MC,M2
```

```
    4 JCCORE(I)=1
      M1=D(4)
C                     ADDRESS OF VALUE
    5 D(4)=J1RESO(MI,MC,2)
C                     VALUE
    6 CALL FRMCEL(JCCORE(MA),D(4),D(5),V,1)
      M3=V(4)+4
      IF((D(5)+D(M3)).EQ.0)GO TO 7
      CALL FRMCEL(JCCORE(MA),D(4),VALUE,V,2)
C                     WRITE
      WRITE(ME,103)(D(I),I=2,3),VALUE
  103 FORMAT(4H F  2I29,D48.28)
      IF(M1.GT.0)WRITE(ME,104)(JCCORE(I),I=MC,M2)
  104 FORMAT(4H G  4I29)
      IF(JCSENS.GE.0)GO TO 7
      WRITE(MF,101)(D(I),I=1,3),VALUE
      IF(M1.GT.0)WRITE(MF,102)(JCCORE(I),I=MC,M2)
C                     INCREMENT INDICES
    7 IF(M1.EQ.0)GO TO 9
      M5=D(3)
      M3=M2
      M4=M1+M5-1
      M6=1
      DO 8 I=1,M5
      IF(M5.EQ.0)GO TO 8
      JCCORE(M3)=JCCORE(M3)+1
      M6=0
      IF(JCCORE(M3).LE.JCCORE(M4))GO TO 8
      M6=1
      JCCORE(M3)=1
      M3=M3-1
      M4=M4-1
    8 CONTINUE
C                     ITERATE
      IF(M6.EQ.0)GO TO 5
C                     INCREMENT L(L(1))
    9 M1=L(1)
      L(M1)=L(M1)+1
      IF(L(M1).LE.JCFINE(2))GO TO 3
C                     EXIT
   10 CALL SYSEXT
      RETURN
      END
```

```
C
C*******************************************************************
C                              **********
C                              * J1VARI *
C                              **********
C
C       PURPOSE
C           J1VARI HANDLES THE INPUT AND OUTPUT FOR VARIABLE LISTS.
C
C       USAGE
C           CALL J1VARI(JCSPUR,JCFIRS,JCMODE,JCMTHD,JCSENS,JCCLNO)
C
C       DATA FORMAT
C           N/A
C
C       DESCRIPTION OF PARAMETERS
C           JCSPUR = THE SPUR FOR TEMPORARY LIST
C           JCFIRS = THE ADDRESS OF THE FIRST LIST ENTRY
C           JCMODE = 1 FOR AN OUTPUT LIST AND 2 OTHERWISE
C           JCMTHD = THE METHOD TEMPORARY LIST
C           JCSENS = SEE J1OIMS
C           JCCLNO = A USER ASSIGNED CALL STATEMENT ID. PARAMETER
C
C       REMARKS
C           THE NUMBER REPRESENTED BY JCTCIA MUST, FOR SOME MACHINES,
C           REPLACE THE 1 IN THE NAMED COMMON STATEMENT.
C
C       SUBROUTINE AND FUNCTION SUBPROGRAMS REQUIRED
C           FRMCEL
C           FSTLNK
C           J1OIMS
C           NEWCEL
C           SYSENT
C           SYSEXT
C           TOCELL
C
C       ERROR CODES FOR THIS PROGRAM
C           NONE
C
C       METHOD
C           SELF-EXPLANATORY
C
      SUBROUTINE J1VARI(JCSPUR,JCFIRS,JCMODE,JCMTHD,JCSENS,JCCLNO)
      COMMON/ALLOC/JCCORE(1)
      DIMENSION V(13),D(4),JCSPUR(5),L(5),P(4)
      INTEGER V,D,P
      DATA V/4,1,1,1,21,1,1,2,1,1,3,1,1/,D(1)/0/,P/1,1,1,1/,MH/0/
C                 ENTER
      CALL SYSENT(1,5,1,15,JCCLNO)
C                 MA=SIZE WITHOUT INDICES
      M1=JCCORE(17)+10
      MA=JCCORE(M1)
C                 MB=ADDRESS OF SPUR
      M1=JCCORE(16)+1
      MB=JCCORE(M1)+5*(MA-1)
C                 IN OR OUT
      IF(JCSENS.LT.1)GO TO 3
C                 INPUT
C                 MC=TEMPORARY STORAGE
      MC=0
C                 MD=INPUT DATA SET
      MD=JCCORE(10)
```

```
C                   ME=ADDRESS OF SPUR
      M1=JCCORE(15)+1
      ME=JCCORE(M1)
C                   MF=ADDRESS OF LOAD
      M1=M1+1
      MF=JCCORE(M1)
C                   MG=SCALE FOR UNIT NUMBER
      M1=JCCORE(17)+6
      MG=JCCORE(M1)
C                   READ
    1 READ(MD,101)M1,(D(I),I=2,4)
  101 FORMAT(I1,I19,2I20)
      IF(M1.EQ.0)GO TO 5
C                   GET LIST ENTRY
      M1=MB+5*D(3)*JCMODE
      CALL NEWCEL(JCCORE(M1),M2,1)
C                   LINK
      IF(JCFIRS.EQ.0)JCFIRS=M2
      IF(MC.GT.0)CALL FSTLNK(JCCORE(M1),MC,M2,1)
      MC=M2
C                   STORE DATA
      CALL TOCELL(JCCORE(M1),M2,D,V,1)
      IF(D(3).EQ.0)GO TO 2
      M1=M2+MA
      CALL J1DIMS(M1,D(3),2,JCMODE,1,1)
C                   TEMPORARY LIST
    2 IF(D(4).NE.2)GO TO 1
      L(2)=D(2)/MG
      L(3)=1
      L(4)=1
      M1=5-1/(L(2)+1)
      L(1)=M1
      L(M1)=D(2)-MG*L(2)
      L(2)=L(2)+4
      CALL FETCH(JCCORE(ME),L,M1,P,L,JCCORE(MF),0,1)
      CALL PUSH(JCSPUR,JCMTHD,M1,V(4),1,1)
      IF(MH.EQ.0)MH=JCMTHD
      GO TO 1
C                   OUTPUT
C                   MC=CURRENT LIST ENTRY
    3 MC=JCFIRS
C                   MD AND ME ARE OUTPUT DATA SETS
      MD=JCCORE(11)
      ME=IABS(JCSENS)
C                   GET DATA
    4 IF(MC.LE.1) GO TO 6
      CALL FRMCEL(JCCORE(MB),MC,D,V,1)
C                   WRITE DATA
      M1=MC+MA
      MC=D(1)
      D(1)=1
      WRITE(MD,102)(D(I),I=2,4)
  102 FORMAT(4H C  3I29)
      IF(JCSENS.LT.0)WRITE(ME,101)(D(I),I=1,4)
      JCCORE(13)=6
      IF(D(3).GT.0)CALL J1DIMS(M1,D(3),2,JCMODE,JCSENS,2)
      GO TO 4
C                   UPDATE JCMTHD
    5 JCMTHD=MH
C                   EXIT
    6 CALL SYSEXT
      RETURN
```

```
END
```

```
C
C*****************************************************************
C                          ***************
C                          *    ALPAC    *
C                          *   GENERAL   *
C                          * INFORMATION *
C                          ***************
C
C       SPECIAL INSTRUCTIONS/DEBUG
C           THE FOLLOWING TABLE GIVES THE PACKAGE AND PROGRAM ID.
C           NUMBERS FOR ALPAC.  ALL ARE ON LEVEL 4.
C           PROGRAM          PKG. ID./PROG. ID.
C           HASSAL             1/1
C           J2ACYC             1/2
C           J2AOSA             1/3
C           J2CNZS             1/4
C           J2GOPT             1/5
C           J2MARK             1/6
C           J2MINC             1/7
C           J2NOFZ             1/8
C           J2PATH             1/9
C           J2SWIT             1/10
C           J2TOSA             1/11
C           SPEDUP             2/1
C           J2FOUR             2/2
C           J2GRUP             2/3
C           J2PREC             2/4
C           J2ROWP             2/5
C           J2TADV             2/6
C           BEMADN             3/1
C           J2CPOT             3/2
C           J2ELEM             3/3
C           J2REDU             3/4
C           J2SPRE             3/5
C           J2TOWY             3/6
C
C       CURRENT STATUS
C           OPERATIONAL - ADDITIONAL ROUTINES TO BE PROVIDED.
C
C*****************************************************************
C
```

```
C
C*****************************************************************
C                              **********
C                              * HASSAL *
C                              **********
C
C       PURPOSE
C           HASSAL PERFORMS THE WEIGHTED HUNGARIAN ASSIGNMENT ALGORITHM
C
C       USAGE
C           CALL HASSAL(JCGRUP,JCDFLT,JCCLNO)
C
C       DATA FORMAT
C           N/A
C
C       DESCRIPTION OF PARAMETERS
C           JCGRUP = THE ADDRESS OF A GENDER GROUP.  IF JCGRUP IS 0,
C                    CUDGEL WILL BE USED TO GENERATE A GROUP.
C           JCDFLT = + IF CALL BY DFAULT AND 0 OTHERWISE
C           JCCLNO = A USER ASSIGNED CALL STATEMENT ID. PARAMETER
C
C       REMARKS
C           THE NUMBER REPRESENTED BY JCTCIA MUST, FOR SOME MACHINES,
C           REPLACE THE 1 IN THE NAMED COMMON STATEMENT.
C
C       SUBROUTINE AND FUNCTION SUBPROGRAMS REQUIRED
C           CUDGEL
C           DFAULT
C           FETCH
C           FIND
C           J2ACYC
C           J2AOSA
C           J2CNZS
C           J2MARK
C           J2MINC
C           J2NOFZ
C           J2PATH
C           J2SWIT
C           J2TOSA
C           J7CSUP
C           SIMCIN
C           SIMGEN
C           SIMOPT
C           STORE
C           SYSENT
C           SYSERR
C           SYSEXT
C
C       ERROR CODES FOR THIS PROGRAM
C           CODE FATAL  ERROR(DATA PROVIDED)
C           1    NF     IMPOSSIBLE TO MAKE COMPLETE ASSIGNMENT (ADDRESS
C                       OF SIMBL)
C
C       METHOD
C           HASSAL AUTOMATICALLY PROVIDES A GENDER LIST AND/OR AN SIM
C           IF NECESSARY.
C
C                              **********
C
      SUBROUTINE HASSAL(JCGRUP,JCDFLT,JCCLNO)
      COMMON/ALLOC/JCCORE(1)
      DIMENSION G(4),O(10),C(4),D(4),P(7),N(4),F(4),L(2)
```

```
      INTEGER G,O,C,D,P,F
      DATA G/1,8,1,1/,O/3,12,1,1,13,1,1,5,1,1/,C/1,6,1,1/,D/0,0,0,0/,
     1 P/2,5,1,1,6,1,1/,N/1,10,1,1/,F/1,5,1,1/,L/1,0/
C                     ENTER
      CALL SYSENT(1,4,1,1,JCCLNO)
C                     ACCESS GROUP AND SIM
      MA=JCCFLT
      IF(JCCFLT.EQ.0)CALL OFAULT(JCGRUP,MA,2,1)
C                     MB=ADDRESS OF SPUR FOR ORDINATE
      M1=MA+2
      MB=JCCORE(M1)-5
C                     MC=ADDRESS OF SPUR FOR ELEMENT
      MC=MB-5
C                     MD=ADDRESS OF LOAD FOR ROW - MAIN
      MD=J7OSUP(MA,1,0,0,1)
C                     ME=ADDRESS OF LOAD(8) FOR ROW - MAIN
      ME=MD+7
C                     MH=ADDRESS OF LOAD FOR COLUMN - MAIN
      MH=J7OSUP(MA,1,1,0,3)
C                     MI=ADDRESS OF LOAD(8) FOR COLUMN - MAIN
      MI=MH+7
C                     ZERO THE OUTPUT ASSIGNMENT
      L(2)=1-JCCORE(ME)
    1 M1=1
      CALL FETCH(JCCORE(MB),L,O,O,L,JCCORE(MD),M1,1)
      IF(M1.EQ.-1)GO TO 4
      L(2)=1
      IF((D(2).EQ.0).OR.(D(3).EQ.1))GO TO 1
      IF(C(1).EQ.0)GO TO 3
C                     MULTIPLE OUTPUT
    2 M1=C(2)
      M2=M1+1
      D(2)=JCCORE(M2)
C                     BREAK ASSIGNMENT
    3 CALL SIMFLG(MA,JCCORE(ME),O,O,1,1)
      CALL SIMCIN(MA,JCCORE(ME),O,1,1,D(3),C,1)
      CALL SIMFLG(MA,C(2),1,O,1,2)
      CALL SIMCIN(MA,D(2),1,1,1,D(3),C,2)
      CALL SIMOPT(MA,JCCORE(ME),C(2),-1,1)
C                     ITERATE
      IF(C(1).EQ.0)GO TO 1
      M1=JCCORE(M1)
      IF(M1.GT.0)GO TO 2
      GO TO 1
C                     DUPLICATE MAINS
    4 CALL SIMDUP(MA,1,3,O,1)
      CALL SIMDUP(MA,1,3,1,2)
C                     SUBTRACTION STEP
      M1=O
   41 M2=MD
      M3=ME
      MO=O
    5 L(2)=1-JCCORE(M3)
      M4=M3+1
      M4=JCCORE(M4)
      DO 8 I=1,M4
      CALL FETCH(JCCORE(MB),L,D,P,L,JCCORE(M2),O,2)
      IF(C(1).EQ.0)GO TO 6
      D(1)=O
      GO TO 7
    6 M5=O
      IF(M1.NE.2)M5=J2MINC(MC,D(2),MO,1)
```

```
      D(1)=J2NCFZ(MC,D(2),M5,MO,1)
    7 L(2)=0
      CALL STORE(JCCORE(M8),L,D,N,L,JCCORE(M2),1)
    8 L(2)=1
      M1=M1+1
      GO TO (81,41,9),M1
   81 M2=MH
      M3=MI
      MO=1
      GO TO 5
C                     ACYCLIC ASSIGNMENT
    9 CALL J2ACYC(MA,2,1)
C                     TEST FOR COMPLETE OUTPUT ASSIGNMENT
   10 L(2)=1-JCCORE(ME)
      M1=1
      M2=1
      CALL FIND(JCCORE(M8),L,D(3),F,L,JCCORE(MO),M1,1)
      IF(M1.EQ.0)GO TO 16
      GO TO (11,12),M2
C                     ARBITRARY ASSIGNMENT
   11 CALL J2ACSA(MA,M1,1)
      M2=2
      IF(M1.EQ.0)GO TO 10
      CALL J2ACYC(MA,1,2)
      GO TO 11
C                     MARK ROWS AND COLUMNS
   12 CALL J2MARK(MA,M1,M3,1)
      IF(M1.EQ.0)GO TO 13
      CALL J2SWIT(MA,M1,M3,1,1)
      GO TO 10
C                     SEARCH FOR STEWARD PATH
   13 CALL J2PATH(MA,M1,M3,1)
      IF(M1.EQ.0)GO TO 14
      CALL J2SWIT(MA,M1,M3,2,2)
      GO TO 10
C                     CREATE NEW ZEROS
   14 CALL J2CNZS(JCSPTR,M1,1)
      IF(M1.EQ.0)GO TO 15
      GO TO 12
C                     IMPOSSIBLE TO MAKE COMPLETE OUTPUT ASSIGNMENT
   15 CALL SYSERR(1,MA)
C                     INSTALL OUTPUT ASSIGNMENT IN GENDER
   16 CALL J2TOSA(JCGRUP,MA,1)
C                     EXIT
      CALL SYSEXT
      RETURN
      END
```

```
C
C*******************************************************************
C                              **********
C                              * J2ACYC *
C                              **********
C
C       PURPOSE
C           J2ACYC IS THE RUDD AND WATSON ACYCLIC ASSIGNMENT ALGORITHM.
C
C       USAGE
C           CALL J2ACYC(JCSPTR,JCMODE,JCCLNO)
C
C       DATA FORMAT
C           N/A
C
C       DESCRIPTION OF PARAMETERS
C           JCMODE = 1 FOR COLUMNS ONLY OR 2 FOR BOTH ROWS AND COLUMNS
C           SEE J2MINC
C
C       REMARKS
C           THE NUMBER REPRESENTED BY JCTCIA MUST, FOR SOME MACHINES,
C           REPLACE THE 1 IN THE NAMED COMMON STATEMENT.
C
C       SUBROUTINE AND FUNCTION SUBPROGRAMS REQUIRED
C           FETCH
C           FIND
C           FRMCEL
C           J70SUP
C           LOCATE
C           SIMCIN
C           SIMFLG
C           SIMOPT
C           STORE
C           SYSENT
C           SYSERR
C           SYSEXT
C           TOCELL
C
C       ERROR CODES FOR THIS PROGRAM
C           CODE FATAL  ERROR(DATA PROVIDED)
C           1    NF     NO ELEMENT WITH ZERO COST (-ROW NUMBER OR +
C                       COLUMN NUMBER)
C           2    NF     ELEMENT SHOULD HAVE POSSESSED NON-ZERO FLAG
C                       (-ROW NUMBER OR + COLUMN NUMBER)
C
C       METHOD
C           SELF-EXPLANATORY
C
C                              **********
C
      SUBROUTINE J2ACYC(JCSPTR,JCMODE,JCCLNO)
      COMMON/ALLOC/JCCORE(1)
      DIMENSION P(2),S(2),Z(7),N(4),C(7),D(3),O(4)
      INTEGER P,S,Z,C,D,O
      DATA P/1,0/,S/1,0/,Z/2,10,1,1,5,1,1/,N/1,4,1,1/,C/2,5,1,1,6,1,1/,
     1 D/1,0,0/,O/1,11,1,1/
C                 ENTER
      CALL SYSENT(1,4,1,2,JCCLNO)
C                 MA=ADDRESS OF SPUR FOR ORDINATE
      M1=JCSPTR+2
      MA=JCCORE(M1)-5
C                 MB=ADDRESS OF LOAD FOR ROW
```

```
      MB=J7OSUP(JCSPTR,1,0,0,1)
C                     MC=ADDRESS OF LOAD FOR COLUMN
      MC=J7OSUP(JCSPTR,1,1,0,2)
C                     MD=ADDRESS OF LOAD(8) FOR ROW
      MD=MB+7
C                     ME=ADDRESS OF LOAD(8) FOR COLUMN
      ME=MC+7
C                     MF=ADDRESS OF SPUR FOR ELEMENT
      MF=MA-5
C                     INITIALIZATION
      MP=MB
      MPC=MD
      MS=MC
      MSC=ME
C                     MG=INDICATOR
      MG=1
      IF(JCMODE.EQ.1)GO TO 5
C                     MH=NUMBER OF ER ASSIGNMENTS THIS PASS
      MH=0
C                     FIND PRIMARY WITH ONE ZERO
    1 P(2)=1-JCCORE(MPC)
      N(2)=5-MG
    2 M1=3
      CALL FIND(JCCORE(MA),P,0,Z,P,JCCORE(MP),M1,1)
      IF(M1.EQ.0)GO TO 5
C                     LOCATE ELEMENT WITH ZERO COST
      P(2)=0
      CALL FETCH(JCCORE(MA),P,M1,C(4),P,JCCORE(MP),0,1)
    3 CALL LOCATE(JCCORE(MF),M1,M2,D(2),C,MG,1)
      IF(M2.EQ.0)GO TO 6
C                     GET SECONDARY ENTRY NUMBER
      CALL FRMCEL(JCCORE(MF),M2,S(2),N,1)
      S(2)=S(2)-JCCORE(MSC)
C                     GET SECONDARY FLAG
      CALL FETCH(JCCORE(MA),S,M1,Z(4),S,JCCORE(MS),0,2)
      IF(M1.NE.0)GO TO 7
C                     OUTPUT ASSIGNMENT
      M1=JCCORE(MD)
      M2=JCCORE(ME)
      CALL SIMOPT(JCSPTR,M1,M2,1,1)
      P(2)=0
      CALL FETCH(JCCORE(MA),P,M3,0,P,JCCORE(MB),0,3)
      IF(M3.GT.0)GO TO 4
      CALL SIMCIN(JCSPTR,M1,0,-1,1,D(2),C(4),1)
      CALL SIMFLG(JCSPTR,M1,0,2,0,1)
      GO TO 41
    4 MH=MH+1
   41 CALL SIMCIN(JCSPTR,M2,1,-1,1,D(2),C(4),2)
      CALL SIMFLG(JCSPTR,M2,1,2,0,2)
C                     ITERATE
      P(2)=1
      GO TO 2
C                     LAST PASS
    5 IF((MG.EQ.2).AND.(MH.EQ.0))GO TO 8
      MH=0
      M1=MP
      MP=MS
      MS=M1
      M1=MPC
      MPC=MSC
      MSC=M1
      MG=3-MG
```

```
      GO TO 1
C                     ABORT PRIMARY
    6 M1=JCCORE(MD)*(-1)**MG
      CALL STORE(JCCORE(MA),P,D(2),Z(4),P,JCCORE(MB),1)
      CALL SYSERR(1,M1)
      P(2)=1
      GO TO 2
C                     ABORT SECONDARY
    7 CALL TOCELL(JCCORE(MF),M2,M1,Z(4),1)
      M1=-JCCORE(ME)*(-1)**MG
      CALL SYSERR(2,M1)
      M1=M2
      GO TO 3
C                     EXIT
    8 CALL SYSEXT
      RETURN
      END
```

```
C
C*********************************************************************
C                                **********
C                                * J2AOSA *
C                                **********
C
C        PURPOSE
C            J2AOSA MAKES AN ARBITRARY OUTPUT ASSIGNMENT
C
C        USAGE
C            CALL J2AOSA(JCSPTR,JCSORF,JCCLNO)
C
C        DATA FORMAT
C            N/A
C
C        DESCRIPTION OF PARAMETERS
C            JCSPTR = THE ADDRESS OF THE SIMBL
C            JCSORF = 1 FOR SUCCESS OR 0 FOR FAILURE
C            JCCLNO = A USER ASSIGNED CALL STATEMENT ID. PARAMETER
C
C        REMARKS
C            THE NUMBER REPRESENTED BY JCTCIA MUST, FOR SOME MACHINES,
C            REPLACE THE 1 IN THE NAMED COMMON STATEMENT.
C
C        SUBROUTINE AND FUNCTION SUBPROGRAMS REQUIRED
C            FETCH
C            FIND
C            FRMCEL
C            J7OSUP
C            LNKBWD
C            LOCATE
C            SIMCIN
C            SIMFLG
C            SIMOPT
C            STORE
C            SYSENT
C            SYSERR
C            SYSEXT
C
C        ERROR CODES FOR THIS PROGRAM
C            CODE FATAL  ERROR(DATA PROVIDED)
C            1    NF     ELEMENT IN COLUMN WITH ZERO COST AND ZERO FLAG
C                        CANNOT BE FOUND (COLUMN NUMBER)
C            2    NF     ROW COES NOT HAVE ZERO FLAG (ROW NUMBER)
C            3    NF     NO ROW WITH ZERO FLAG FOUND (COLUMN NUMBER)
C
C        METHOD
C            SELF-EXPLANATORY
C
C                                **********
C
      SUBROUTINE J2AOSA(JCSPTR,JCSORF,JCCLNO)
      COMMON/ALLOC/JCCORE(1).
      DIMENSION V(2),F(4),C(4),E(7),R(4),B(7),D(4),H(2)
      INTEGER V,F,C,E,R,C,B,H
      DATA V/1,0/,F/1,5,1,1/,C/1,10,1,1/,E/2,5,1,1,6,1,1/,R/1,3,1,1/,
     1 B/2,5,1,1,10,1,1/,D/0,0,0,0/,H/1,0/
C                 ENTER
      CALL SYSENT(1,4,1,3,JCCLNO)
C                 MA=ADDRESS OF SPUR FOR ORDINATE
      M1=JCSPTR+2
      MA=JCCORE(M1)-5
```

```
C                   MB=ADDRESS OF LOAD FOR ROW
      MB=J7OSUP(JCSPTR,1,0,0,1)
C                   MC=ADDRESS OF LOAD FOR COLUMN
      MC=J7OSUP(JCSPTR,1,1,0,2)
C                   MD=ADDRESS OF LOAD(8) FOR ROW
      MD=MB+7
C                   ME=ADDRESS OF LOAD(8) FOR COLUMN
      ME=MC+7
C                   MF=ADDRESS OF SPUR FOR ELEMENT
      MF=MA-5
C                   MG=MINIMUM COUNT
      MG=0
C                   MH=COLUMN NUMBER
      MH=0
C                   MI=MINIMUM COUNT
      MI=0
C                   MJ=ROW NUMBER
      MJ=0
C                   FIND A VARIABLE WITH 0 FLAG
      V(2)=1-JCCORE(ME)
    1 JCSORF=1
      CALL FIND(JCCORE(MA),V,D,F,V,JCCORE(MC),JCSORF,1)
      IF(JCSORF.EQ.0)GO TO 2
C                   UPDATE MINIMUM COUNT
      V(2)=0
      CALL FETCH(JCCORE(MA),V,M1,C,V,JCCORE(MC),0,1)
      V(2)=1
      IF((MG*(M1-MG+1)).GT.0)GO TO 1
      MG=M1
      MH=JCCORE(ME)
      GO TO 1
C                   END OF SEARCH FOR VARIABLE
    2 IF(MG.EQ.0)GO TO 8
      V(2)=MH-JCCORE(ME)
      CALL FETCH(JCCORE(MA),V,M1,E(4),V,JCCORE(MC),0,2)
C                   LOCATE ELEMENT
      DO 4 I=1,MG
      IF(M1.EQ.0)GO TO 4
      CALL LOCATE(JCCORE(MF),M1,M2,D,E,2,1)
      IF(M2.EQ.0)GO TO 5
      CALL FRMCEL(JCCORE(MF),M2,H(2),R,1)
      H(2)=H(2)-JCCORE(MD)
      CALL FETCH(JCCORE(MA),H,D(3),B,H,JCCORE(MB),0,3)
      IF(D(3).NE.0)GO TO 6
      IF((MI*(D(4)-MI+1)).GT.0)GO TO 3
      MI=D(4)
      MJ=JCCORE(MD)
    3 MI=LNKBWD(JCCORE(MF),M2,1)
    4 CONTINUE
C                   END OF SEARCH FOR EQUATION
      IF(MI.EQ.0)GO TO 7
C                   OUTPUT ASSIGNMENT
      CALL SIMOPT(JCSPTR,MJ,MH,1,1)
      CALL SIMCIN(JCSPTR,MJ,0,-1,1,D,E(4),1)
      CALL SIMFLG(JCSPTR,MJ,0,2,0,1)
      CALL SIMCIN(JCSPTR,MH,1,-1,1,D,E(4),2)
      CALL SIMFLG(JCSPTR,MH,1,2,0,2)
      JCSORF=1
      GO TO 8
C                   ABORT REMAINDER OF COLUMN
    5 M1=0
      D(4)=MG-I+1
```

```
      D(3)=0
      IF(D(4).EQ.MG)D(1)=2
      V(2)=0
      CALL STORE(JCCORE(MA),V,D(3),B,V,JCCORE(MC),1)
      CALL SYSERR(1,JCCORE(ME))
      GO TO 4
C                     ABORT ROW
    6 CALL SIMFLG(JCSPTR,JCCORE(MD),0,D(3),3)
      D(4)=MG-1
      V(2)=0
      D(3)=0
      IF(D(4).EQ.0)D(3)=2
      CALL STORE(JCCORE(MA),V,D(3),B,V,JCCORE(MC),2)
      CALL SYSERR(2,JCCORE(MD))
      GO TO 4
C                     NO EQUATION
    7 CALL SYSERR(3,JCCORE(ME))
      V(2)=1
      GO TO 1
C                     EXIT
    8 CALL SYSEXT
      RETURN
      END
```

```
C
C*******************************************************************
C                              **********
C                              * J2CNZS *
C                              **********
C
C       PURPOSE
C           J2CNZS CREATES NEW ZERO COST ELEMENTS FROM THE UNASSIGNED
C           BALANCE OF THE SIM.
C
C       USAGE
C           CALL J2CNZS(JCSPTR,JCSORF,JCCLNO)
C
C       DATA FORMAT
C           N/A
C
C       DESCRIPTION OF PARAMETERS
C           JCSPTR = THE ADDRESS OF THE SIMBL
C           JCSORF = 0 OR 1 AS RESPECTIVELY NEW ZERO COST ELEMENTS ARE
C                    NOT OR ARE GENERATED
C           JCCLNO = A USER ASSIGNED CALL STATEMENT ID. PARAMETER
C
C       REMARKS
C           THE NUMBER REPRESENTED BY JCTCIA MUST, FOR SOME MACHINES,
C           REPLACE THE 1 IN THE NAMED COMMON STATEMENT.
C
C       SUBROUTINE AND FUNCTION SUBPROGRAMS REQUIRED
C           FETCH
C           FIND
C           FRMCEL
C           J7OSUP
C           STORE
C           SYSENT
C           SYSEXT
C           TOCELL
C
C       ERROR CODES FOR THIS PROGRAM
C           NONE
C
C       METHOD
C           J2CNZS LOCATES THE MINIMUM UNMARKED ELEMENT.  J2CNZS ADDS
C           THIS MINIMUM TO DOUBLY MARKED ELEMENTS AND SUBTRACTS IT
C           FROM UNMARKED ELEMENTS.  SINGLY MARKED ELEMENTS REMAIN
C           UNCHANGED.  THE ROW AND COLUMN COUNTS ARE ADJUSTED TO
C           REFLECT THE APPEARANCE OF NEW ZERO COSTS.
C
C                              **********
C
      SUBROUTINE J2CNZS(JCSPTR,JCSORF,JCCLNO)
      COMMON/ALLOC/JCCORE(1)
      DIMENSION L(3),M(7),D(4),F(4),C(4),N(7),K(4)
      INTEGER D,F,C
      DATA L/1,0,0/,M/2,5,1,1,13,1,1/,D/0,0,0,0/,F/1,5,1,1/,C/1,6,1,1/,
     1 N/2,4,1,1,1,1,1/,K/1,10,1,1/
C                 ENTER
      CALL SYSENT(1,4,1,4,JCCLNO)
C                 MA=ADDRESS OF SPUR FOR ORDINATE
      M1=JCSPTR+2
      MA=JCCORE(M1)-5
C                 MB=ADDRESS OF LOAD FOR ROW - MAIN
      MB=J7OSUP(JCSPTR,1,0,0,1)
C                 MC=ADDRESS OF LOAD(8) FOR ROW - MAIN
```

```
      MC=MB+7
C                     MD=ADDRESS OF LOAD FOR ROW - WORKING
      MD=J70SUP(JCSPTR,3,0,0,2)
C                     ME=ADDRESS OF LOAD(8) FOR ROW - WORKING
      ME=MD+7
C                     MF=ADDRESS OF LOAD FOR COLUMN - MAIN
      MF=J70SUP(JCSPTR,1,1,0,3)
C                     MG=ADDRESS OF LOAD(8) FOR COLUMN - MAIN
      MG=MF+7
C                     MH=ADDRESS OF LOAD FOR COLUMN - WORKING
      MH=J70SUP(JCSPTR,3,1,0,4)
C                     MI=ADDRESS OF LOAD(8) FOR COLUMN - WORKING
      MI=MH+7
C                     MJ=ADDRESS OF SPUR FOR ELEMENT
      MJ=MA-5
C                     INITIALIZATION
      JCSORF=0
      D(1)=0
      D(2)=0
      N(2)=4
C                     ML=MINIMUM COST
      ML=-1
C                     FIND ROW WITH NO MARK
    1 L(2)=1-JCCORE(ME)
    2 M5=1
      CALL FIND(JCCORE(MA),L,D,M,L,JCCORE(MD),M5,1)
      IF(M5.EQ.0)GO TO 7
C                     LOCATE UNMARKED ELEMENT
      L(2)=0
      CALL FETCH(JCCORE(MA),L,M5,C,L,JCCORE(MD),0,1)
    3 CALL FRMCEL(JCCORE(MJ),M5,L(2),N,1)
      L(2)=L(2)-JCCORE(MI)
      CALL FETCH(JCCORE(MA),L,D(3),M,L,JCCORE(MH),0,2)
      IF((D(3).EQ.1).OR.(D(4).GT.0))GO TO 4
      JCSORF=1
      CALL FRMCEL(JCCORE(MJ),M5,M6,C,2)
      IF((ML*(M6-ML)).LT.0)ML=M6
C                     ITERATE
    4 M5=L(3)
      IF(M5.GT.0)GO TO 3
    5 L(2)=1
      GO TO 2
C                     SUBTRACT ML FROM ALL UNMARKED ELEMENTS
C                     ADD ML TO ALL DOUBLY MARKED ELEMENTS
C                     LEAVE SINGLY MARKED ELEMENTS UNCHANGED
    7 L(2)=1-JCCORE(MC)
      N(2)=4
    8 M1=1
C                     MM=NUMBER OF ZERO'S IN ROW
      CALL FETCH(JCCORE(MA),L,MM,K,L,JCCORE(MB),M1,3)
      IF(M1.EQ.-1)GO TO 14
C                     GET ROW FLAG AND MARK
      L(2)=JCCORE(MC)-JCCORE(ME)
      CALL FETCH(JCCORE(MA),L,D,M,L,JCCORE(MD),0,4)
      IF(D(1).EQ.1)GO TO 13
      M1=0
      IF(D(2).GT.0)M1=1
C                     ACCESS ELEMENT
      L(2)=0
      CALL FETCH(JCCORE(MA),L,M2,C,L,JCCORE(MB),D,5)
C                     GET COLUMN NUMBER AND FORWARD (ROW) LINK
    9 CALL FRMCEL(JCCORE(MJ),M2,D,N,3)
```

```
C                   GET COLUMN MARK AND FLAG
      L(2)=D(1)-JCCORE(MI)
      CALL FETCH(JCCORE(MA),L,D(3),M,L,JCCORE(MH),0,6)
      IF(D(3).EQ.1)GO TO 12
      M3=M1
      IF(D(4).GT.0)M3=M3+1
      M4=1
      IF(M3.EQ.1)GO TO 12
C                   ADJUST ELEMENT COST
      CALL FRMCEL(JCCORE(MJ),M2,M4,C,4)
      M4=M4-ML*(-1)**(M3/2)
      CALL TOCELL(JCCORE(MJ),M2,M4,C,1)
      IF(M4.GT.0)GO TO 12
C                   INCREMENT COLUMN COUNT (AND ROW COUNTER MM)
      L(2)=JCCORE(MI)-JCCORE(MG)
      CALL FETCH(JCCORE(MA),L,M4,K,L,JCCORE(MF),0,7)
      M4=M4+1
      L(2)=0
      CALL STORE(JCCORE(MA),L,M4,K,L,JCCORE(MF),1)
      MM=MM+1
C                   ITERATE
   12 M2=D(2)
      IF(M2.GT.0)GO TO 9
      L(2)=0
      CALL STORE(JCCORE(MA),L,MM,K,L,JCCORE(MB),2)
   13 L(2)=1
      GO TO 8
C                   EXIT
   14 CALL SYSEXT
      RETURN
      END
```

```
C
C*******************************************************************
C                              **********
C                              * J2GOPT *
C                              **********
C
C       PURPOSE
C           J2GOPT ESTABLISHES GROUP BODY OUTPUT VARIABLE LIST ENTRIES
C
C       USAGE
C           J2GOPT(JCFUNC,JCVARI,JCFLAG,JCMODE,JCCLNO)
C
C       DATA FORMAT
C           N/A
C
C       DESCRIPTION OF PARAMETERS
C           JCFUNC = THE FUNCTION AUXILIARY BLOCK OR 0 IF UNDIMENSIONED
C           JCVARI= THE VARIABLE AUXILIARY BLOCK OR THE VARIABLE NAME
C                   IF UNDIMENSIONED
C           JCFLAG = THE ROW FLAG
C           JCMODE = 0 IF THE VARIABLE IS UNDIMENSIONED AND 1 OTHERWISE
C           JCCLNO = A USER ASSIGNED CALL STATEMENT ID. PARAMETER
C
C       REMARKS
C           THE NUMBER REPRESENTED BY JCTCIA MUST, FOR SOME MACHINES,
C           REPLACE THE 1 IN THE NAMED COMMON STATEMENT.
C
C       SUBROUTINE AND FUNCTION SUBPROGRAMS REQUIRED
C           FETCH
C           J7COMV
C           J7ICOM
C           NEWCEL
C           RETURN
C           SYSENT
C           SYSEXT
C           TOCELL
C
C       ERROR CODES FOR THIS PROGRAM
C           NONE
C
C       METHOD
C           SELF-EXPLANATORY
C
C                              **********
C
      FUNCTION J2GOPT(JCFUNC,JCVARI,JCFLAG,JCMODE,JCCLNO)
      COMMON/ALLOC/JCCORE(1)
      DIMENSION L(6),D(2),F(7),N(4),X(4),G(10)
      INTEGER D,F,X,G
      DATA L/5,0,0,0,1,0/,F/2,5,1,1,1,1,1/,N/1,8,1,1/,X/1,1,1,1/,G/2,
     1 21,1,1,2,1,1,21,1,1/
C                     ENTER
      CALL SYSENT(1,4,1,5,JCCLNO)
C                     MA=DISPLACEMENT FOR OUTPUT
      M1=JCCORE(17)+10
      MA=JCCORE(M1)
C                     MB=ADDRESS OF SPUR FOR OUTPUT
      MB=JCCORE(16)+1
      MB=JCCORE(MB)+5*MA-5
C                     MC=UNIT SCALE
      M1=M1-4
      MC=JCCORE(M1)
```

```
C                   MD=ADDRESS OF SPUR FOR SECEDE
      M1=JCCORE(15)+1
      MD=JCCORE(M1)
C                   ME=ADDRESS OF LOAD FOR SECEDE (1)
      M1=M1+1
      ME=JCCORE(M1)
      M1=ME+28
      JCCORE(M1)=-1
C                   MF=ADDRESS OF LOAD FOR SECEDE (2)
      MF=M1+2
C                   ADJUST CURRENT VALUE POINTER IN GIMBL
      M1=JCCORE(16)+8
      MG=JCCORE(M1)
      JCCORE(M1)=JCFUNC+3
C                   MH=LENGTH OF A SCALAR VARIABLE
      M1=JCCORE(17)+3
      MH=JCCORE(M1)
C                   GET J2GOPT
      M1=JCVARI+2
      M1=MB+5*JCCORE(M1)*JCMODE
      CALL NEWCEL(JCCORE(M1),J2GOPT,1)
      IF(M1.EQ.MB)GO TO 71
      IF(JCFUNC.EQ.0)GO TO 5
C                   L
      M1=JCFUNC+1
      L(2)=JCCORE(M1)/MC
      L(3)=JCFLAG+6
      L(4)=JCCORE(M1)-MC*L(2)
C                   ACCESS FUNCTION
      CALL FETCH(JCCORE(MD),L,L(6),F(4),L,JCCORE(ME),0,1)
C                   ACCESS CONSTITUENT
    1 CALL FETCH(JCCORE(MD),L(5),D,F,L,JCCORE(ME),0,2)
      IF(D(1).NE.1)GO TO 3
C                   GET NAME
      L(6)=0
      CALL FETCH(JCCORE(MD),L,D,N,L,JCCORE(ME),0,3)
      D(1)=D(1)+MC*L(2)
      CALL J7COMV(MD,D(1),MF,1)
      M1=JCVARI+1
      IF(D(1).NE.JCCORE(M1))GO TO 3
C                   COMPUTE INDICES
      M1=M1+1
      CALL J7ICOM(MD,MH,MB,J2GOPT,MA,1,JCCORE(M1),ME,M2,1)
      M2=J2GOPT+MA-1
      M3=0
      M4=JCCORE(M1)
      DO 2 I=1,M4
      IF(M3.EQ.1)GO TO 2
      M1=M1+1
      M2=M2+1
      IF(JCCORE(M1).NE.JCCORE(M2))M3=1
    2 CONTINUE
      IF(M3.EQ.0)GO TO 4
C                   ITERATE
      IF(D(2).GT.0)D(2)=1
    3 L(6)=D(2)
      IF(L(6).GT.0)GO TO 1
C                   ERROR
      M1=JCFUNC+1
      CALL SYSERR(1,JCCORE(M1))
      M1=JCVARI+2
      M1=MB+JCCORE(M1)*5
```

```
      CALL RETURN(JCCORE(M1),J2GOPT,J2GOPT,1)
      J2GOPT=0
      GO TO 8
C                     TRANSFER INDICES FROM SECEDE
    4 L(6)=1-M4
      M1=J2GOPT+MA
      X(4)=M4
      CALL FETCH(JCCORE(MD),L(5),JCCORE(M1),X,L,JCCORE(ME),0,4)
      GO TO 7
C                     TRANSFER INDICES FROM SIM
    5 M1=JCVARI+2
      M2=JCCORE(M1)
      M3=J2GOPT+MA-1
      DO 6 I=1,M2
      M1=M1+1
      M3=M3+1
    6 JCCORE(M3)=JCCORE(M1)
C                     INSTALL NAME AND DIMENSIONALITY
    7 M1=JCVARI+1
      CALL TOCELL(JCCORE(M8),J2GOPT,JCCORE(M1),G,1)
      GO TO 8
   71 CALL TOCELL(JCCORE(M8),J2GOPT,JCVARI,G(7),2)
C                     EXIT
    8 CALL SYSEXT
      RETURN
      END
```

```
C
C*******************************************************************
C                              **********
C                              * J2MARK *
C                              **********
C
C        PURPOSE
C            J2MARK PERFORMS MARKING OF ROWS AND COLUMNS WITHOUT A FULL
C            OUTPUT ASSIGNMENT.
C
C        USAGE
C            CALL J2MARK(JCSPTR,JCRNUM,JCCNUM,JCCLNO)
C
C        DATA FORMAT
C            N/A
C
C        DESCRIPTION OF PARAMETERS
C            JCSPTR = THE ADDRESS OF THE SIMBL
C            JCRNUM = 0 OR THE ROW NUMBER OF AN ELEMENT FOUND TO BE
C                     TRIPLY MARKED
C            JCCNUM = ANALOGOUS TO JCRNUM, BUT FOR THE COLUMN
C            JCCLNO = A USER ASSIGNED CALL STATEMENT ID. PARAMETER
C
C        REMARKS
C            THE NUMBER REPRESENTED BY JCTCIA MUST, FOR SOME MACHINES,
C            REPLACE THE 1 IN THE NAMED COMMON STATEMENT.
C
C        SUBROUTINE AND FUNCTION SUBPROGRAMS REQUIRED
C            FETCH
C            FIND
C            FRHCEL
C            J7OSUP
C            J7OTPT
C            LOCATE
C            SIMINT
C            SYSENT
C            SYSEXT
C
C        ERROR CODES FOR THIS PROGRAM
C            NONE
C
C        METHOD
C            SELF-EXPLANATORY
C
C                              **********
C
      SUBROUTINE J2MARK(JCSPTR,JCRNUM,JCCNUM,JCCLNO)
      COMMON/ALLOC/JCCORE(1)
      DIMENSION P(2),S(2),F(4),C(4),N(7),D(8),M(10),L(4)
      INTEGER P,S,F,C,O
      DATA P/1,0/,S/1,0/,F/1,5,1,1/,C/1,6,1,1/,N/2,4,1,1,5,1,1/,O/0,0,
     1 0,0,0,0,0,0/,M/3,11,1,1,12,I,1,13,1,1/,L/1,1,1,1/
C                     ENTER
      CALL SYSENT(1,4,1,6,JCCLNO)
C                     MA=ADDRESS OF SPUR FOR ORDINATE
      M1=JCSPTR+2
      MA=JCCORE(M1)-5
C                     MB=ADDRESS OF LOAD FOR ROW - MAIN
      MB=J7OSUP(JCSPTR,1,0,0,1)
C                     MC=ADDRESS OF LOAD FOR COLUMN - MAIN
      MC=J7OSUP(JCSPTR,1,1,0,2)
C                     MD=ADDRESS OF LOAD FOR ROW - WORKING
```

```
      MD=J7OSUP(JCSPTR,3,0,0,3)
C                     ME=ADDRESS OF LOAD FOR COLUMN - WORKING
      ME=J7OSUP(JCSPTR,3,1,0,4)
C                     MF=ADDRESS OF LOAD(8) FOR ROW - MAIN
      MF=MB+7
C                     MG=ADDRESS OF LOAD(8) FOR COLUMN - MAIN
      MG=MC+7
C                     MH=ADDRESS OF LOAD(8) FOR ROW - WORKING
      MH=MD+7
C                     MI=ADDRESS OF LOAD(8) FOR COLUMN - WORKING
      MI=ME+7
C                     MJ=ADDRESS OF SPUR FOR ELEMENT
      MJ=MA-5
C                     MK=INDICATOR
      MK=1
C                     INITIALIZATION
      JCRNUM=0
      JCCNUM=0
C                     FIND PRIMARY WITH ZERO FLAG
    1 P(2)=1-JCCORE(MF)
      N(2)=5-MK
      L(2)=MK
    2 M1=1
      CALL FIND(JCCORE(MA),P,O,F,P,JCCORE(MB),M1,1)
      IF(M1.EQ.0)GO TO 12
C                     GET POINTER
      P(2)=0
      CALL FETCH(JCCORE(MA),P,M1,C,P,JCCORE(MB),O,1)
C                     LOCATE ELEMENT WITH ZERO COST
    3 CALL LOCATE(JCCORE(MJ),M1,M2,O,C,MK,1)
      IF(M2.EQ.0)GO TO 11
C                     GET SECONDARY NUMBER AND STATUS FLAG
      CALL FRMCEL(JCCORE(MJ),M2,D(2),N,1)
      IF(D(3).EQ.1)GO TO 10
C                     GET SECONDARY MARK
      S(2)=D(2)-JCCORE(MI)
      CALL FETCH(JCCORE(MA),S,D(2),M,S,JCCORE(ME),O,2)
      IF((D(4)*(MK-1)).EQ.0)GO TO 8
C                     GET OUTPUT OF SECONDARY
    4 S(2)=JCCORE(MI)-JCCORE(MG)
      CALL FETCH(JCCORE(MA),S,D(5),M,S,JCCORE(MC),O,3)
      IF(D(7).EQ.0)GO TO 8
      IF(D(6).EQ.0)GO TO 6
C                     ACCESS OUTPUT ASSIGNMENT (AND MARK OF OUTPUT)
      M1=D(7)+1
    5 D(7)=JCCORE(M1)
    6 P(2)=D(7)-JCCORE(MH)
      CALL FETCH(JCCORE(MA),P,D(8),M(7),P,JCCORE(MD),O,4)
      IF(D(8).GT.0)GO TO 7
C                     ITERATE
      IF(D(6).EQ.0)GO TO 8
      M3=M1-1
      M1=JCCORE(M3)
      IF(M1.GT.0)GO TO 5
      GO TO 8
C                     TRIPLE MARKED
    7 JCCNUM=JCCORE(MI)
      JCRNUM=JCCORE(MH)
      IF(MK.EQ.1)GO TO 8
      JCCNUM=JCCORE(MH)
      JCRNUM=JCCORE(MI)
C                     MARK SECONDARY
```

```
    8 IF(D(4).GT.0)GO TO 9
      M1=2-MK
      CALL SIMINT(JCSPTR,1,3,M1,1)
      CALL J7OTPT(JCSPTR,JCCORE(MI),M1,2,JCCORE(MF),1)
      CALL SIMINT(JCSPTR,1,3,M1,2)
C                    ITERATE
    9 IF(JCRNUM.GT.0)GO TO 13
   10 CALL FRMCEL(JCCORE(MJ),M2,M1,L,2)
      IF(M1.GT.0)GO TO 3
   11 P(2)=1
      GO TO 2
C                    SWITCH
   12 IF(MK.EQ.2)GO TO 13
      M1=MB
      MB=MC
      MC=M1
      M1=MD
      MD=ME
      ME=M1
      M1=MF
      MF=MG
      MG=M1
      M1=MH
      MH=MI
      MI=M1
      MK=2
      GO TO 1
C                    EXIT
   13 CALL SYSEXT
      RETURN
      END
```

```
C
C*******************************************************************
C                              **********
C                              * J2MINC *
C                              **********
C
C       PURPOSE
C           J2MINC FINDS THE MINIMUM COST ELEMENT OF A ROW OR COLUMN.
C
C       USAGE
C           J2MINC(JCSPUR,JCFIRS,JCRORC,JCCLNO)
C
C       DATA FORMAT
C           N/A
C
C       DESCRIPTION OF PARAMETERS
C           JCSPUR = THE ADDRESS OF THE SPUR FOR ELEMENTS
C           JCFIRS = THE ADDRESS OF THE FIRST ELEMENT
C           JCRORC = 0 FOR ROW, 1 FOR COLUMN
C           JCCLNO = A USER ASSIGNED CALL STATEMENT ID. PARAMETER
C
C       REMARKS
C           THE NUMBER REPRESENTED BY JCTCIA MUST, FOR SOME MACHINES,
C           REPLACE THE 1 IN THE NAMED COMMON STATEMENT.
C
C       SUBROUTINE AND FUNCTION SUBPROGRAMS REQUIRED
C           FRMCEL
C           SYSENT
C           SYSEXT
C
C       ERROR CODES FOR THIS PROGRAM
C           NONE
C
C       METHOD
C           SELF-EXPLANATORY
C
C                              **********
C
      FUNCTION J2MINC(JCSPUR,JCFIRS,JCRORC,JCCLNO)
      COMMON/ALLOC/JCCORE(1)
      DIMENSION C(10),D(3)
      INTEGER C,D
      DATA C/3,1,1,1,5,1,1,6,1,1/
C                     ENTER
      CALL SYSENT(1,4,1,7,JCCLNO)
C                     INITIALIZATION
      J2MINC=-1
      C(2)=1+JCRORC
      M1=JCFIRS
C                     EXAMINE ELEMENT
    1 CALL FRMCEL(JCCORE(JCSPUR),M1,D,C,1)
      IF(D(2).NE.0)GO TO 2
      IF((J2MINC*(D(3)-J2MINC)).LT.0)J2MINC=D(3)
    2 M1=D(1)
      IF(M1.GT.0)GO TO 1
C                     EXIT
      CALL SYSEXT
      RETURN
      END
```

```
C
C*******************************************************************
C                              **********
C                              * J2NOFZ *
C                              **********
C
C       PURPOSE
C           J2NOFZ PERFORMS THE SUBTRACTION AND COMPUTES THE NUMBER OF
C           ZERO'S.
C
C       USAGE
C           J2NOFZ(JCSPUR,JCFIRS,JCMINC,JCRORC,JCCLNO)
C
C       DATA FORMAT
C           N/A
C
C       DESCRIPTION OF PARAMETERS
C           JCMINC = MINIMUM COST FOR ROW OR COLUMN
C           SEE J2MINC FOR ALL OTHER PARAMETERS
C
C       REMARKS
C           THE NUMBER REPRESENTED BY JCTCIA MUST, FOR SOME MACHINES,
C           REPLACE THE 1 IN THE NAMED COMMON STATEMENT.
C
C       SUBROUTINE AND FUNCTION SUBPROGRAMS REQUIRED
C           FRMCEL
C           SYSENT
C           SYSEXT
C           TOCELL
C
C       ERROR CODES FOR THIS PROGRAM
C           NONE
C
C       METHOD
C           SELF-EXPLANATORY
C
C                              **********
C
      FUNCTION J2NOFZ(JCSPUR,JCFIRS,JCMINC,JCRORC,JCCLNO)
      COMMON/ALLOC/JCCORE(1)
      DIMENSION C(10),D(3)
      INTEGER C,D
      DATA C/3,1,1,1,5,1,1,6,1,1/
C                     ENTER
      CALL SYSENT(1,4,1,8,JCCLNO)
C                     INITIALIZATION
      J2NOFZ=0
      C(2)=1+JCRORC
      M1=JCFIRS
C                     EXAMINE ELEMENT
    1 CALL FRMCEL(JCCORE(JCSPUR),M1,D,C,1)
      IF(D(2).NE.0)GO TO 2
      D(3)=D(3)-JCMINC
      IF(D(3).EQ.0)J2NOFZ=J2NOFZ+1
      CALL TOCELL(JCCORE(JCSPUR),M1,D(3),C(7),1)
    2 M1=D(1)
      IF(M1.GT.0)GO TO 1
C                     EXIT
      CALL SYSEXT
      RETURN
      END
```

```
C
C*****************************************************************
C                              **********
C                              * J2SWIT *
C                              **********
C
C       PURPOSE
C           J2SWIT PERFORMS THE OUTPUT ASSIGNMENT ADJUSTMENTS FOR A
C           STEWARD PATH.
C
C       USAGE
C           CALL J2SWIT(JCSPTR,JCRNUM,JCCNUM,JCMODE,JCCLNO)
C
C       DATA FORMAT
C           N/A
C
C       DESCRIPTION OF PARAMETERS
C           JCMODE = 1 OR 2 AS RESPECTIVELY THE TRIPLE MARK WAS
C                    DISCOVERED BY J2MARK OR BY J2PATH
C           FOR THE OTHER PASSED PARAMETERS SEE J2MARK.
C
C       REMARKS
C           THE NUMBER REPRESENTED BY JCTCIA MUST, FOR SOME MACHINES,
C           REPLACE THE 1 IN THE NAMED COMMON STATEMENT.
C
C       SUBROUTINE AND FUNCTION SUBPROGRAMS REQUIRED
C           FETCH
C           FIND
C           J7OSUP
C           J7OTPT
C           LNKFWD
C           LOCATE
C           SIMCIN
C           SIMFLG
C           SIMINT
C           SIMOPT
C           SYSENT
C           SYSERR
C           SYSEXT
C
C       ERROR CODES FOR THIS PROGRAM
C           CODE FATAL  ERROR(DATA PROVIDED)
C           1    NF     NO OUTPUT FOR MARK (+ ROW NO. OR - COLUMN NO.)
C           2    NF     NO MARK FOR OUTPUT (+ ROW NO. OR - COLUMN NO.)
C
C       METHOD
C           J2SWIT ELIMINATES THE APPROPRIATE ROW AND COLUMN MARKS.
C
C                              **********
C
      SUBROUTINE J2SWIT(JCSPTR,JCRNUM,JCCNUM,JCMODE,JCCLNO)
      COMMON/ALLOC/JCCORE(1)
      DIMENSION L(2),M(7),K(4),D(3),R(2),C(2),E(10),F(7)
      INTEGER C,D,R,E,F
      DATA L/1,0/,M/2,12,1,1,13,1,1/,K/1,6,1,1/,D/0,0,0/,E/3,12,1,1,13,
     1 1,1,13,1,1/,F/2,2,1,1,2,1,1/
C                 ENTER
      CALL SYSENT(1,4,1,10,JCCLNO)
C                 MA=ADDRESS OF SPUR FOR ORDINATE
      M1=JCSPTR+2
      MA=JCCORE(M1)-5
C                 MB=ADDRESS OF LOAD FOR ROW - MAIN
```

```
      MB=J7OSUP(JCSPTR,1,0,0,1)
C                     MC=ADDRESS OF LOAD FOR COLUMN - MAIN
      MC=J7OSUP(JCSPTR,1,1,0,2)
C                     MD=ADDRESS OF LOAD FOR ROW - WORKING
      MD=J7OSUP(JCSPTR,3,0,0,3)
C                     ME=ADDRESS OF LOAD FOR COLUMN - WORKING
      ME=J7OSUP(JCSPTR,3,1,0,4)
C                     MF=ADDRESS OF LOAD(8) FOR ROW - MAIN
      MF=MB+7
C                     MG=ADDRESS OF LOAD(8) FOR COLUMN - MAIN
      MG=MC+7
C                     MH=ADDRESS OF LOAD(8) FOR ROW - WORKING
      MH=MD+7
C                     MI=ADDRESS OF LOAD(8) FOR COLUMN - WORKING
      MI=ME+7
C                     MJ=INDICATOR
      MJ=1
C                     MK=INDICATOR
      MK=1
C                     INITIALIZATION
      E(1)=JCMODE+1
      ML=JCRNUM
      MM=JCCNUM
      MN=0
      F(1)=JCMODE
C                     BREAK OUTPUT ASSIGNMENT
    1 CALL SIMFLG(JCSPTR,ML,0,0,1,1)
      CALL SIMCIN(JCSPTR,ML,0,1,1,0,K,1)
      CALL SIMFLG(JCSPTR,MM,1,0,1,2)
      CALL SIMCIN(JCSPTR,MM,1,1,1,0,K,2)
      CALL SIMOPT(JCSPTR,ML,MM,-1,1)
      IF(MK.EQ.3)GO TO 6
C                     GET MARKS
    2 M1=ML
      M2=MD
      M3=MH
    3 L(2)=M1-JCCORE(M3)
      CALL FETCH(JCCORE(MA),L,D(2),M,L,JCCORE(M2),0,1)
      IF(D(3).EQ.0)GO TO 12
      IF(D(2).EQ.0)GO TO 4
      M1=D(3)+1
      D(3)=JCCORE(M1)
    4 C(MJ)=D(3)
      IF(M2.EQ.MD)R(MJ)=C(MJ)
      GO TO (5,7,11,13),MK
    5 MK=2
    6 M1=MM
      M2=ME
      M3=MI
      GO TO 3
C                     GET OUTPUT
    7 IF(JCMODE.EQ.1)GO TO 16
      MJ=2
      MK=3
      M1=C(1)
      M2=MB
      M3=MF
    8 L(2)=M1-JCCORE(M3)
      CALL FETCH(JCCORE(MA),L,D(2),M,L,JCCORE(M2),0,2)
      IF(D(3).EQ.0)GO TO 24
      MN=0
      IF(D(2).EQ.0)GO TO 9
```

```
      MN=D(3)+1
      D(3)=JCCORE(MN)
    9 IF(MK.EQ.4)GO TO 10
      MO=D(3)
      ML=C(1)
      MM=MO
      GO TO 1
   10 ML=D(3)
      MM=R(1)
      GO TO 1
   11 M1=R(1)
      M2=MC
      M3=MG
      MK=4
      GO TO 8
   12 IF(MN.EQ.0)GO TO 25
      MN=MN-1
      MN=JCCORE(MN)
      IF(MN.EQ.0)GO TO 25
      MN=MN+1
      D(3)=JCCORE(MN)
      GO TO 9
C                     MAKE OUTPUT ASSIGNMENTS
   13 MM=R(2)
   14 CALL SIMOPT(JCSPTR,ML,MM,1,2)
      CALL SIMCIN(JCSPTR,ML,0,-1,1,0,K,3)
      CALL SIMFLG(JCSPTR,ML,0,2,0,3)
      CALL SIMCIN(JCSPTR,MM,1,-1,1,0,K,4)
      CALL SIMFLG(JCSPTR,MM,1,2,0,4)
      GO TO (18,17,16,15),MK
   15 MK=3
      ML=C(2)
      MM=MO
      GO TO 14
   16 MK=2
      ML=JCRNUM
      MM=R(1)
      GO TO 14
   17 MK=1
      ML=C(1)
      MM=JCCNUM
      GO TO 14
C                     ELIMINATE MARKS
   18 MK=0
      M2=MO
      M3=MH
      D(1)=1
      D(2)=R(1)
      D(3)=R(2)
   19 L(2)=1-JCCORE(M3)
      CALL SIMINT(JCSPTR,1,3,MK,1)
   20 M1=-1
      CALL FIND(JCCORE(MA),L,D,E,L,JCCORE(M2),M1,1)
      IF(M1.EQ.0)GO TO 23
      M4=0
      IF(M1.NE.1)GO TO 22
      L(2)=0
      CALL FETCH(JCCORE(MA),L,M4,M(4),L,JCCORE(M2),0,3)
      L(2)=1
   21 IF(M4.EQ.0)GO TO 20
      M1=-1
      CALL LOCATE(JCCORE(MA),M4,M5,D(2),F,M1,1)
```

```
      IF(M5.EQ.0)GO TO 20
      M1=M1+1
   22 CALL J7OTPT(JCSPTR,JCCORE(M3),MK,-1,O(M1),1)
      L(2)=1
      IF(M4.EQ.0)GO TO 20
      M4=LNKFWD(JCCORE(MA),M4,1)
      GO TO 21
   23 CALL SIMINT(JCSPTR,1,3,MK,2)
      IF(MK.EQ.1)GO TO 26
      MK=1
      M2=ME
      M3=MI
      O(2)=C(1)
      O(3)=C(2)
      GO TO 19
C                     NO OUTPUT
   24 M5=(-1)**MK*JCCORE(M3)
      CALL SYSERR(1,M5)
      GO TO 16
C                     NO MARK
   25 M5=(-1)**MK*JCCORE(M3)
      CALL SYSERR(2,M5)
      GO TO 16
C                     EXIT
   26 CALL SYSEXT
      RETURN
      END
```

```
C
C*****************************************************************
C                              **********
C                              * J2TOSA *
C                              **********
C
C       PURPOSE
C           J2TOSA TRANSFERS AN OUTPUT SET ASSIGNMENT FROM AN SIM TO A
C           GENDER GROUP.
C
C       USAGE
C           CALL J2TOSA(JCGRUP,JCSPTR,JCCLNO)
C
C       DATA FORMAT
C           N/A
C
C       DESCRIPTION OF PARAMETERS
C           JCGRUP = THE ADDRESS OF THE GENDER GROUP
C           JCSPTR = THE ADDRESS OF THE SIMBL
C           JCCLNO = A USER ASSIGNED CALL STATEMENT ID. PARAMETER
C
C       REMARKS
C           THE NUMBER REPRESENTED BY JCTCIA MUST, FOR SOME MACHINES,
C           REPLACE THE 1 IN THE NAMED COMMON STATEMENT.
C
C       SUBROUTINE AND FUNCTION SUBPROGRAMS REQUIRED
C           FETCH
C           FIND
C           FRMCEL
C           FSTLNK
C           J2COPT
C           J2TADV
C           SEARCH
C           TOCELL
C           SYSENT
C           SYSEXT
C
C       ERROR CODES FOR THIS PROGRAM
C           CODE FATAL  ERROR(DATA PROVIDED)
C           1    NF     NOT ENOUGH GROUP BODIES (NUMBER OF ROWS)
C           2    NF     NO OUTPUT ASSIGNMENT (ROW NUMBER)
C           3    NF     TOO MANY OUTPUTS (ROW NUMBER)
C           4    NF     INCOMPLETE OUTPUT ASSIGNMENT (ROW NUMBER)
C
C       METHOD
C           J2TOSA MUST NOT BE EMPLOYED IF REARRANCEMENT OF THE BODY OF
C           JCGRUP HAS OCCURRED SINCE SIM GENERATION.
C
C                              **********
C
      SUBROUTINE J2TOSA(JCGRUP,JCSPTR,JCCLNO)
      COMMON/ALLOC/JCCORE(1)
      DIMENSION R(2),C(2),K(4),G(7),F(4),H(19),V(10),P(4),D(11),S(4),
     1 O(4)
      INTEGER R,C,G,F,H,V,P,D,S,O
      DATA R/1,0/,C/1,0/,K/1,4,1,1/,G/2,3,1,1,3,1,1/,F/1,3,2,2/,H/6,7,1,
     1 1,8,1,1,9,1,1,11,1,1,12,1,1,13,1,1/,V/3,5,1,1,7,1,1,9,1,1/,P/1,
     2 18,1,1/,S/1,7,2,2/,O/1,20,1,1/
C                 ENTER
      CALL SYSENT(1,4,1,11,JCCLNO)
C                 MA=ADDRESS OF SPUR FOR ORDINATE
      M1=JCSPTR+2
```

```
      MA=JCCORE(M1)-5
C                     MB=ADDRESS OF LOAD FOR ROW
      MB=J7OSUP(JCSPTR,1,0,0,1)
C                     R
      M1=MB+7
      R(2)=1-JCCORE(M1)
C                     MC=NUMBER OF ROWS
      M1=M1+1
      MC=JCCORE(M1)
C                     MD=ADDRESS OF LOAD FOR COLUMN
      MD=J7OSUP(JCSPTR,1,1,0,2)
C                     ME=ADDRESS OF LOAD(8) FOR COLUMN
      ME=MD+7
C                     MF=ADDRESS OF SPUR FOR GROUP BODY
      M1=JCCORE(16)+1
      M2=JCCORE(17)+9
      MF=JCCORE(M1)+5*JCCORE(M2)-5
C                     MG=TRACE KEY
      CALL FRMCEL(JCCORE(MF),JCGRUP,MG,K,1)
C                     MH=TEMPORARY LIST POINTER
      MH=0
C                     MI=LEVEL
      MI=0
C                     MJ=LEVEL OF PROTECTED GROUP
      MJ=0
C                     MK=CURRENT LIST ENTRY
      MK=JCGRUP
C                     ML=OVER-RIDE INDICATOR
      ML=0
C                     DO LOOP
      DO 10 I=1,MC
      IF(ML.EQ.1)GO TO 3
C                     LIST ADVANCE
    1 M1=MK
      M2=-1
      D(1)=1
      D(2)=2
      CALL SEARCH(JCCORE(16),M1,MK,D,G,MG,M2,MI,MH,1)
      IF(M2.EQ.0)GO TO 11
    2 IF((MJ*(MJ-MI)).LT.0)GO TO 1
      MJ=0
      IF(M2.EQ.2)GO TO 3
C                     GROUP
      CALL FRMCEL(JCCORE(MF),MK,M1,F,2)
      IF(M1.EQ.1)MJ=MI
      GO TO 1
C                     GROUP BODY
C                     SIM ADVANCE
    3 CALL FETCH(JCCORE(MA),R,D(3),H,R,JCCORE(MB),0,1)
      IF(D(6).GT.0)CALL SYSERR(4,JCCORE(ME))
C                     PREPARE FOR J2GOPT CALL
      ML=0
      M2=0
      IF(D(4).LT.3)GO TO 4
      ML=1
      GO TO 10
    4 M1=0
      IF(D(3).GT.0)M1=D(5)
      CALL FRMCEL(JCCORE(MF),MK,D(6),0,3)
      C(2)=D(8)
      IF(D(7).EQ.0)GO TO 6
    5 M3=D(8)+1
```

```
      C(2)=JCCORE(M3)
    6 IF(C(2).EQ.0)GO TO 12
      C(2)=C(2)-JCCORE(ME)
      D(6)=D(6)-1
C                     ACCESS OUTPUT
      CALL FETCH(JCCORE(MA),C,D(9),V,C,JCCORE(MD),0,2)
C                     CALL J2GOPT
      M3=M2
      M2=J2GOPT(M1,D(11),D(4),D(10),1)
C                     LINK
      IF(M3.GT.0)GO TO 7
      M3=M2
      CALL TOCELL(JCCORE(MF),MK,M3,P,1)
      GO TO 8
    7 CALL FSTLNK(JCCORE(MF),M3,M2,1)
C                     ASSIGNMENT FLAG
      M3=0
      IF(D(9).EQ.1)M3=1
      CALL TOCELL(JCCORE(MF),M2,M3,G(4),2)
C                     ITERATE
    8 IF(D(7).EQ.0)GO TO 9
      M3=D(8)
      D(8)=JCCORE(M3)
      IF(D(8).GT.0)GO TO 5
C                     END OF LOOP
    9 IF(D(6).NE.0)GO TO 13
   10 R(2)=1
      C(2)=1-JCCORE(ME)
      D(1)=0
      MM=0
  101 M1=1
      CALL FIND(JCCORE(MA),C,D,H(16),C,JCCORE(MD),M1,1)
      IF(M1.EQ.0)GO TO 102
      C(2)=0
      CALL FETCH(JCCORE(MA),C,D(2),V,C,JCCORE(MD),0,3)
      M1=J2TADV(D(4),D(2),D(3),1)
      CALL FSTLNK(JCCORE(MF),M1,MM,2)
      MM=M1
      C(2)=1
      GO TO 101
C                     EXIT
  102 M1=JCCORE(17)+8
      M2=JCCORE(M1)
      M1=JCCORE(M1+1)
      M1=MF+5*(M2-M1)
      CALL TOCELL(JCCORE(M1),JCGRUP,MM,S,3)
      CALL SYSEXT
      RETURN
C                     NOT ENOUGH GROUP BODIES
   11 CALL SYSERR(1,MC)
      GO TO 10
C                     NO OUTPUT ASSIGNMENT
   12 CALL SYSERR(2,I)
      GO TO 8
C                     MISMATCH OF NUMBER OF OUTPUTS
   13 CALL SYSERR(3,JCCORE(ME))
      GO TO 10
      END
```

```
C
C*****************************************************************
C                              **********
C                              * SPEDUP *
C                              **********
C
C       PURPOSE
C           SPEDUP PERFORMS THE SARGENT AND WESTERBERG SPEED-UP I
C           ALGORITHM.
C
C       USAGE
C           CALL SPEDUP(JCGRUP,JCMODE,JCCLNO)
C
C       DATA FORMAT
C           N/A
C
C       USAGE
C           JCMODE = 0 OR 1 AS RESPECTIVELY THE SIM IS OR IS NOT TO BE
C                    RELEASED AT THE CONCLUSION OF SPEDUP
C           SEE HASSAL FOR ALL OTHER PASSED PARAMETERS.
C
C       REMARKS
C           THE NUMBER REPRESENTED BY JCTCIA MUST, FOR SOME MACHINES,
C           REPLACE THE 1 IN THE NAMED COMMON STATEMENT.
C           SPEDUP UTILIZES CUDGEL, SIMGEN AND HASSAL TO AUTOMATICALLY
C           PROVIDE ANY LIST STRUCTURE OR FEATURE REQUIRED FOR THE
C           SPEED-UP I ALGORITHM.
C
C       SUBROUTINE AND FUNCTION SUBPROGRAMS REQUIRED
C           BSTLNK
C           CORN
C           DFAULT
C           FETCH
C           FIND
C           FRMCEL
C           FSTLNK
C           J2FOUR
C           J2GRUP
C           J2PREC
C           J2ROWP
C           J7OSUP
C           LNKBWD
C           LNKFWD
C           LOCATE
C           POPUP
C           PUSH
C           RETURN
C           SETLNK
C           STORE
C           SYSENT
C           SYSEXT
C           TOCELL
C
C       ERROR CODES FOR THIS PROGRAM
C           NONE
C
C       METHOD
C           SELF-EXPLANATORY
C
C                              **********
C
      SUBROUTINE SPEDUP(JCGRUP,JCMODE,JCCLNO)
```

```
      COMMON/ALLOC/JCCORE(1)
      DIMENSION S(4),T(5),R(2),F(4),P(4),E(10),C(4),V(10),O(4),G(7),B(1)
     1 ,L(4),Z(1),O(3)
      INTEGER S,T,R,F,P,E,C,V,O,G,B,Z,O
      DATA S/1,8,1,1/,T/0,0,1,C,2/,R/1,0/,F/1,12,1,1/,P/1,6,1,1/,E/3,3,
     1 1,1,21,1,1,22,1,1/,C/1,0/,V/3,4,1,1,5,1,1,3,1,1/,O/1,13,1,1/,G/2,
     2 3,1,1,3,1,1/,B/0/,L/1,1,1,1/,Z/-1/
C                     ENTER
      CALL SYSENT(1,4,2,1,JCCLNO)
C                     DEFAULT
      CALL DFAULT(JCGRUP,MB,3,1)
C                     MC=ADDRESS OF SPUR FOR ORDINATE
      M1=MB+2
      MC=JCCORE(M1)-5
C                     MD=ADDRESS OF SPUR FOR ELEMENT
      MD=MC-5
C                     ME=ADDRESS OF LOAD FOR ROW
      ME=J70SUP(MB,1,0,0,1)
      MF=ME+7
      MG=MF+1
      MG=JCCORE(MG)
C                     MH=ADDRESS OF LOAD FOR COLUMN
      MH=J70SUP(MB,1,1,0,2)
      MI=MH+7
C                     T
      T(1)=0
      T(2)=0
      M2=JCCORE(17)+11
      T(4)=JCCORE(M2)
C                     REPLACE OUTPUT WITH POINTER (ROWS ONLY)
      MAA=0
      CALL J2ROWP(JCGRUP,MC,ME,T,MAA,1)
C                     MJ=LIST 4 HEADCELL
      MJ=0
C                     MK=LIST 3 HEADCELL
      MK=0
C                     ML=LIST 3 TAILCELL
      ML=0
C                     MM=GROUPING LIST HEADCELL
      MM=0
C                     START
      DO 99 I=1,MG
C                     STEP II
      R(2)=I-JCCORE(MF)
C                     STEP III
      CALL FETCH(JCCORE(MC),R,M1,F,R,JCCORE(ME),0,1)
C                     B
      IF(M1.EQ.1)GO TO 99
C                     C
      CALL J2FOUR(T,MJ,MC,0,ME,1,1)
C                     A
      CALL J2GRUP(T,MM,MJ,1)
C                     STEP IV
    4 IF(MJ.EQ.0)GO TO 99
      CALL FRMCEL(T,MJ,O,E,2)
C                     STEP V
    5 IF(O(1).EQ.1)GO TO 4
C                     STEP VI
    6 MN=MJ
      IF(O(1).EQ.0)GO TO 62
      MN=O(3)
      MO=MJ
```

```
   61 CALL FRMCEL(T,MN,O(3),E(7),3)
   62 IF(O(3).GT.0)GO TO 63
      IF(O(1).EQ.0)GO TO 9
      MN=LNKFWD(T,MN,1)
      CALL TOCELL(T,MO,MN,E(7),2)
      IF(MN.GT.0)GO TO 61
      GO TO 9
   63 MP=O(3)
      O(3)=LNKFWD(JCCORE(MO),MP,2)
      CALL TOCELL(T,MN,O(3),E(7),3)
C                    STEP VII
    7 CALL FRMCEL(JCCORE(MO),MP,C(2),V,4)
      IF(C(3).GE.3)GO TO 62
      C(2)=C(2)-JCCORE(MI)
      CALL FETCH(JCCORE(MC),C,R(2),O,C,JCCORE(MH),0,2)
C                    A
      IF(R(2).GT.0)GO TO 71
      C(2)=0
      CALL FETCH(JCCORE(MC),C,M1,F,C,JCCORE(MH),0,21)
      IF(M1.GT.0)GO TO 62
      M1=1
      CALL STORE(JCCORE(MC),C,M1,F,C,JCCORE(MH),10)
      CALL J2FOUR(T,MJ,0,0,MH,3,2)
      GO TO 62
C                    B
C                    2
   71 IF(R(2).EQ.C(4))GO TO 4
      R(2)=R(2)-JCCORE(MF)
      CALL FETCH(JCCORE(MC),R,M1,F,R,JCCORE(ME),0,3)
      IF(M1.EQ.1)GO TO 62
C                    1
      R(2)=0
      CALL J2FOUR(T,M1,MC,R(2),ME,2,3)
      CALL J2GRUP(T,MM,M1,2)
C                    3
      E(1)=2
      CALL FRMCEL(T,M1,O,E,5)
      CALL LOCATE(T,MJ,M2,O,E,1,1)
      E(1)=3
C                    B
      IF(M2.GT.0)GO TO 8
C                    A
      CALL SETLNK(T,M1,MJ,1)
      MJ=M1
      GO TO 4
C                    STEP VIII
C                    A
    8 MQ=MJ
      MJ=LNKFWD(T,M2,3)
      CALL FSTLNK(T,M2,O,1)
      MR=M1
C                    B
      O(1)=1
      O(2)=2
      M1=MQ
   81 M2=-1
      CALL LOCATE(T,M1,M3,O,G,M2,2)
      IF(M3.EQ.0)GO TO 84
      M1=LNKFWD(T,M3,4)
      IF(M2.EQ.2)GO TO 82
      CALL SETLNK(T,M3,MJ,2)
      MJ=M3
```

```
      M2=LNKBWD(T,MJ,1)
      CALL SETLNK(T,M2,M1,3)
      GO TO 81
   82 CALL FRMCEL(T,M3,M2,E(4),6)
      M4=LNKBWD(T,M3,2)
      CALL SETLNK(T,M4,M2,3)
      CALL RETURN(T,M3,M3,1)
      CALL LOCATE(T,M2,M4,B,L,1,3)
      CALL SETLNK(T,M4,M1,4)
      CALL LOCATE(T,MM,M2,M3,E(4),1,4)
      IF(M2.EQ.0)GO TO 83
      IF(M2.EQ.MM)MM=LNKFWD(T,MM,5)
      CALL POPUP(T,M2,M2,M4,Z,1,1)
   83 IF(M1.GT.0)GO TO 81
C                    D
   84 CALL SETLNK(T,MR,MJ,5)
      MJ=MR
      CALL PUSH(T,MM,MJ,E(4),1,1)
C                    C
      D(1)=2
      D(2)=MQ
      D(3)=MQ
      CALL TOCELL(T,MJ,D,E,4)
C                    E
      GO TO 4
C                    STEP IX
    9 CALL FRMCEL(T,MJ,D,E,7)
      IF(D(1).EQ.1)GO TO 93
      IF(D(1).EQ.0)GO TO 92
C                    A
   91 M1=D(2)
      CALL FRMCEL(T,M1,D(2),E(4),8)
   92 R(2)=D(2)-JCCORE(MF)
      D(2)=1
      CALL STORE(JCCORE(MC),R,D(2),F,R,JCCORE(ME),1)
      IF(D(1).NE.2)GO TO 93
      D(2)=LNKFWD(T,M1,6)
      IF(D(2).GT.0)GO TO 91
C                    B
   93 CALL SETLNK(T,ML,MJ,6)
      IF(MK.EQ.0)MK=MJ
      ML=MJ
      MJ=LNKFWD(T,MJ,7)
      CALL FSTLNK(T,ML,0,2)
      IF(MJ.GT.0)CALL BSTLNK(T,MJ,0,1)
C                    C
      IF((MJ*(-1)**D(1)).LT.0)GO TO 9
C                    D
      GO TO 4
C                    END OF LOOP
   99 CONTINUE
C                    PRECEDENCE ORDER GROUP
      CALL J2PREC(JCGRUP,MC,ME,MH,T,MK,MAA,1)
C                    RETURN SIM AND T ALLOCATION
      IF(JCMODE.EQ.1)GO TO 100
      CALL CORN(JCCORE(MC),1)
      D(1)=0
      M1=JCCORE(16)+1
      M2=JCCORE(17)+8
      MA=JCCORE(M1)+5*JCCORE(M2)-5
      CALL TOCELL(JCCORE(MA),JCGRUP,D,S,5)
  100 CALL CORN(T,2)
```

```
C                    EXIT
      CALL SYSEXT
      RETURN
      END
```

```
C
C*****************************************************************
C                              **********
C                              * J2FOUR *
C                              **********
C
C       PURPOSE
C           J2FOUR ACQUIRES LIST ENTRIES FOR LIST FOUR.
C
C       USAGE
C           CALL J2FOUR(JCSPUR,JCHDCL,JCSPTR,JCDISP,JCLOAD,JCMODE,
C           JCCLNO)
C
C       DATA FORMAT
C           N/A
C
C       DESCRIPTION OF PARAMETERS
C           JCSPUR = THE SPUR VECTOR FOR LIST FOUR
C           JCHDCL = THE ADDRESS OF THE HEADCELL FOR LIST FOUR
C           JCSPTR = THE ADDRESS OF THE SPUR VECTOR FOR ROW ORDINATE
C           JCDISP = THE DISPLACEMENT TO BE APPLIED TO THE LOAD VECTOR
C                    TO REACH THE DESIRED ROW
C           JCLOAD = THE ADDRESS OF THE LOAD VECTOR FOR ROW ORDINATE IF
C                    JCMODE = 1 OR 2, OR FOR COLUMN ORDINATE IF JCMODE
C                    = 3
C           JCMODE = AN INTEGER VALUED AT 1, 2 OR 3.  IF EQUAL TO 1 OR
C                    2 THE NEW LIST ENTRY REPRESENTS AN SIM ROW WHICH
C                    IS OR IS NOT THREADED ONTO LIST FOUR RESPECTIVELY.
C                    IF EQUAL TO 3, THE NEW LIST ENTRY REPRESENTS AN
C                    SIM COLUMN AND IS THREADED ONTO LIST FOUR.  THE
C                    PASSED PARAMETERS JCSPTR AND JCDISP ARE IGNORED IF
C                    JCMODE = 3.
C           JCCLNO = A USER ASSIGNED CALL STATEMENT ID. PARAMETER
C
C       REMARKS
C           THE NUMBER REPRESENTED BY JCTCIA MUST, FOR SOME MACHINES,
C           REPLACE THE 1 IN THE NAMED COMMON STATEMENT.
C
C       SUBROUTINE AND FUNCTION SUBPROGRAMS REQUIRED
C           FETCH
C           NEWCEL
C           SETLNK
C           SYSENT
C           SYSEXT
C           TOCELL
C
C       ERROR CODES FOR THIS PROGRAM
C           NONE
C
C       METHOD
C           SELF-EXPLANATORY
C
C                              **********
C
      SUBROUTINE J2FOUR(JCSPUR,JCHDCL,JCSPTR,JCDISP,JCLOAD,JCMODE,
     1 JCCLNO)
      COMMON/ALLOC/JCCORE(1)
      DIMENSION JCSPUR(5),D(3),L(10),R(2),P(4)
      INTEGER D,R,P
      DATA L/3,3,1,1,21,1,1,22,1,1/,R/1,0/,P/1,6,1,1/
C                 ENTER
      CALL SYSENT(1,4,2,2,JCCLNO)
```

```
C                   D(1)=FLAG
      D(1)=JCMODE/3
C                   D(3)=POINTER TO TYPE 2 LIST
      D(3)=0
      IF(JCMODE.GT.2)GO TO 1
      R(2)=JCOISP
      CALL FETCH(JCCORE(JCSPTR),R,D(3),P,R,JCCORE(JCLOAD),0,1)
C                   D(2)=CODE NUMBER
    1 M1=JCLOAD+7
      D(2)=JCCORE(M1)
C                   GET NEW LIST ENTRY
      CALL NEWCEL(JCSPUR,M1,1)
      CALL TOCELL(JCSPUR,M1,0,L,1)
      IF(JCMODE.EQ.2)GO TO 2
C                   THREAD ONTO LIST 4
      CALL SETLNK(JCSPUR,M1,JCHOCL,1)
C                   SET JCHOCL
    2 JCHOCL=M1
C                   EXIT
      CALL SYSEXT
      RETURN
      END
```

```
C
C*******************************************************************
C                              **********
C                              * J2GRUP *
C                              **********
C
C       PURPOSE
C           J2GRUP REPLACES A LIST FOUR ENTRY BY A GROUPING IF POSSIBLE
C
C       USAGE
C           CALL J2GRUP(JCSPUR,JCGRUP,JCFOUR,JCCLNO)
C
C       DATA FORMAT
C           N/A
C
C       DESCRIPTION OF PARAMETERS
C           JCSPUR = THE SPUR FOR THE TEMPORARY LISTS
C           JCGRUP = THE ADDRESS OF THE GROUPING LIST HEADCELL
C           JCFOUR = THE ADDRESS OF THE LIST FOUR ENTRY
C           JCCLNO = A USER ASSIGNED CALL STATEMENT ID. PARAMETER
C
C       REMARKS
C           THE NUMBER REPRESENTED BY JCTCIA MUST, FOR SOME MACHINES,
C           REPLACE THE 1 IN THE NAMED COMMON STATEMENT.
C
C       SUBROUTINE AND FUNCTION SUBPROGRAMS REQUIRED
C           COPY
C           FRMCEL
C           LNKFWD
C           LOCATE
C           SYSENT
C           SYSEXT
C
C       ERROR CODES FOR THIS PROGRAM
C           NONE
C
C       METHOD
C           SELF-EXPLANATORY
C
C                              **********
C
      SUBROUTINE J2GRUP(JCSPUR,JCGRUP,JCFOUR,JCCLNO)
      COMMON/ALLOC/JCCORE(1)
      DIMENSION JCSPUR(5),T(10)
      INTEGER T
      DATA T/3,3,1,1,21,1,1,22,1,1/
C                   ENTER
      CALL SYSENT(I,4,2,3,JCCLNO)
C                   GET DATA
      IF(JCGRUP.EQ.0)GO TO 3
      CALL FRMCEL(JCSPUR,JCFOUR,MA,T(4),1)
C                   SEARCH
      MB=JCGRUP
    1 CALL FRMCEL(JCSPUR,MB,MC,T(4),2)
      CALL FRMCEL(JCSPUR,MC,MD,T(4),3)
      CALL LOCATE(JCSPUR,MD,M1,MA,T(4),1,1)
      IF(M1.GT.0)GO TO 2
      MB=LNKFWD(JCSPUR,MB,1)
      IF(MB.GT.0)GO TO 1
      GO TO 3
C                   GROUPING FOUND
    2 CALL COPY(JCSPUR,MC,JCFOUR,T,1)
```

```
C                       EXIT
    3 CALL SYSEXT
      RETURN
      END
```

```
C
C*****************************************************************
C                              **********
C                              * J2PREC *
C                              **********
C
C       PURPOSE
C           J2PREC REARRANGES A GROUP BODY ACCORDING TO THE PRECEDENCE
C           ORDER INDICATED BY LIST THREE.
C
C       USAGE
C           CALL J2PREC(JCGRUP,JCSPUR,JCLODR,JCLODC,JCTSPR,JCTHRE,
C           JCTEMP,JCCLNO)
C
C       DATA FORMAT
C           N/A
C       DESCRIPTION OF PARAMETERS
C           JCLODR = THE ADDRESS OF THE LOAD FOR ROWS
C           JCLODC = THE ADDRESS OF THE LOAD FOR COLUMNS
C           JCTHRE = THE ADDRESS OF THE LIST THREE HEADCELL
C           FOR ALL OTHER PASSED PARAMETERS SEE J2ROWP.
C
C       REMARKS
C           THE NUMBER REPRESENTED BY JCTCIA MUST, FOR SOME MACHINES,
C           REPLACE THE 1 IN THE NAMED COMMON STATEMENT.
C
C       SUBROUTINE AND FUNCTION SUBPROGRAMS REQUIRED
C           BREAK
C           COUPLE
C           FETCH
C           FRMCEL
C           FSTLNK
C           J2TADV
C           LOCATE
C           NEWCEL
C           POPUP
C           PUSH
C           RETURN
C           SYSENT
C           SYSEXT
C           TOCELL
C
C       ERROR CODES FOR THIS PROGRAM
C           NONE
C
C       METHOD
C           SELF-EXPLANATORY
C
C                              **********
C
      SUBROUTINE J2PREC(JCGRUP,JCSPUR,JCLODR,JCLODC,JCTSPR,JCTHRE,
     1 JCTEMP,JCCLNO)
      COMMON/ALLOC/JCCORE(1)
      DIMENSION JCTSPR(5),D(5),G(7),R(2),A(4),C(4),T(7),L(10),B(4),F(7),
     1 V(2),P(10)
      INTEGER D,G,R,A,C,T,B,F,V,P
      DATA G/2,3,1,1,21,1,1/,R/1,0/,A/1,13,1,1/,C/1,5,1,2/,T/2,21,1,1,
     1 22,1,1/,L/3,3,1,1,8,4,4,7,2,2/,B/1,7,1,1/,F/2,3,1,1,3,1,1/,V/1,0/
     2 ,P/3,9,1,1,5,1,1,7,1,1/
C                    ENTER
      CALL SYSENT(1,4,2,4,JCCLNO)
C                    MA=ADDRESS OF SPUR FOR GENDER GROUP
```

```
      M1=JCCORE(16)+1
      M2=JCCORE(17)+8
      MA=JCCORE(M1)+5*JCCORE(M2)-5
C                     MB=CURRENT GROUP
      MB=JCGRUP
C                     MC=GROUP BODY HEADCELL
      MC=0
C                     MD=PREVIOUS GROUP BODY ENTRY
      MD=0
C                     ME=CURRENT GROUP BODY ENTRY
C                     MF=CURRENT LIST THREE ENTRY
      MF=JCTHRE
C                     MG=GROUP LIST HEADCELL
      MG=0
C                     MH=DECISION VARIABLE LIST
      MH=0
C                     NEXT ENTRY ON LIST THREE
    1 IF(MF.EQ.0)GO TO 11
      CALL POPUP(JCTSPR,MF,MF,D,G,1,1)
C                     MODE
      M1=D(1)+1
      GO TO (2,12,9),M1
C                     GET POINTER
    2 M1=JCLOOR+7
      R(2)=D(2)-JCCORE(M1)
      CALL FETCH(JCCORE(JCSPUR),R,ME,A,R,JCCORE(JCLOOR),0,1)
C                     BREAK LINKS
      CALL FRMCEL(JCCORE(MA),ME,D,C,1)
      M1=D(1)
      IF(M1.EQ.0)GO TO 4
      DO 3 I=1,M1
    3 CALL BREAK(JCCORE(16),ME,0,1,0,1)
    4 M1=D(2)
      IF(M1.EQ.0)GO TO 6
      DO 5 I=1,M1
    5 CALL BREAK(JCCORE(16),0,ME,1,0,2)
C                     SET LINKS
    6 IF(MD.GT.0)GO TO 7
      MC=ME
      CALL TOCELL(JCCORE(MA),MB,MC,8,1)
      GO TO 8
    7 CALL COUPLE(JCCORE(16),MD,ME,0,0,0,1)
    8 MD=ME
      IF(M1.GE.0)GO TO 1
      MB=-M1
      MC=0
      MD=0
      MF=D(2)
      GO TO 1
C                     GROUPING
    9 D(3)=MB
      D(4)=MF
      CALL PUSH(JCTSPR,MG,D(3),T,1,1)
      CALL NEWCEL(JCCORE(MA),M1,1)
      M3=M1+JCCORE(M2)-1
      D(3)=1
      D(4)=1
      D(5)=MH
      CALL TOCELL(JCCORE(MA),M1,D(3),L,2)
      ME=M1
      M1=-M1
      MH=0
```

```
      GO TO 6
C                     END OF GROUPING
   11 IF(MG.EQ.0)GO TO 13
      MD=MB
      CALL POPUP(JCTSPR,MG,MG,D(3),T,1,2)
      MB=D(3)
      MF=D(4)
      GO TO 1
C                     DECISION VARIABLE
   12 IF(MF.EQ.0)GO TO 11
      M1=-1
      D(3)=0
      D(4)=2
      CALL LOCATE(JCTSPR,MF,M3,D(3),F,M1,1)
      IF(M3.EQ.0)GO TO 11
      IF(M1.EQ.1)GO TO 1
      M1=JCLOCC+7
      V(2)=D(2)-JCCORE(M1)
      CALL FETCH(JCCORE(JCSPUR),V,D(3),P,V,JCCORE(JCLOOC),0,2)
      M1=J2TAOV(D(3),D(4),D(5),1)
      CALL FSTLNK(JCCORE(MA),M1,MH,1)
      MH=M1
      GO TO 1
C                     RELEASE GROUPS ON TEMPORARY LIST
   13 IF(JCTEMP.EQ.0)GO TO 14
      CALL POPUP(JCTSPR,JCTEMP,JCTEMP,M1,G(4),1,3)
      CALL RETURN(JCCORE(MA),M1,M1,1)
      GO TO 13
C                     EXIT
   14 CALL SYSEXT
      RETURN
      END
```

```
C
C*****************************************************************
C                              **********
C                              * J2ROWP *
C                              **********
C
C       PURPOSE
C           J2ROWP REPLACES THE ROW OUTPUT WITH A POINTER TO A GROUP
C           BODY ENTRY.
C
C       USAGE
C           CALL J2ROWP(JCGRUP,JCSPUR,JCLOAD,JCTSPR,JCTEMP,JCCLNO)
C
C       DATA FORMAT
C           N/A
C
C       DESCRIPTION OF PARAMETERS
C           JCGRUP = THE ADDRESS OF A GENDER GROUP
C           JCSPUR = THE ADDRESS OF THE SPUR FOR SIM ORDINATES
C           JCLOAD = THE ADDRESS OF THE LOAD FOR ROWS
C           JCTSPR = THE TEMPORARY LIST SPUR
C           JCTEMP = A LIST REFERRING TO INNER UNPROTECTED GROUPS
C           JCCLNO = A USER ASSIGNED CALL STATEMENT ID. PARAMETER
C
C       REMARKS
C           THE NUMBER REPRESENTED BY JCTCIA MUST, FOR SOME MACHINES,
C           REPLACE THE 1 IN THE NAMED COMMON STATEMENT.
C
C       SUBROUTINE AND FUNCTION SUBPROGRAMS REQUIRED
C           FRMCEL
C           NEXT
C           PUSH
C           STORE
C           SYSENT
C           SYSERR
C           SYSEXT
C
C       ERROR CODES FOR THIS PROGRAM
C           CODE FATAL  ERROR(DATA PROVIDED)
C           1    NF     END OF GROUP BODY (CURRENT ROW NUMBER)
C
C       METHOD
C           SELF-EXPLANATORY
C
C                              **********
C
      SUBROUTINE J2ROWP(JCGRUP,JCSPUR,JCLOAD,JCTSPR,JCTEMP,JCCLNO)
      COMMON/ALLOC/JCCORE(1)
      DIMENSION R(2),K(4),D(2),F(4),G(4),A(7),JCTSPR(5)
      INTEGER R,D,F,G,A
      DATA R/1,0/,K/1,4,1,1/,F/1,3,1,2/,G/1,21,1,1/,A/2,13,1,1,12,1,1/
C                 ENTER
      CALL SYSENT(1,4,2,5,JCCLNO)
C                 MA=ADDRESS OF SPUR FOR GENDER (GROUP)
      M1=JCCORE(16)+1
      M2=JCCORE(17)+8
      MA=JCCORE(M1)+5*JCCORE(M2)-5
C                 MB=KEY
      CALL FRMCEL(JCCORE(MA),JCGRUP,MB,K,1)
C                 MC=LEVEL
      MC=0
C                 MD=INDICATOR
```

```
      MD=1
C                     ME=CURRENT ENTRY
      ME=JCGRUP
C                     MF=TEMPORARY LIST HEADCELL
      MF=0
C                     R(2)
      M1=JCLOAD+7
      R(2)=1-JCCORE(M1)
C                     MG=LIMIT FOR DO LOOP
      M1=M1+1
      MG=JCCORE(M1)
C                     DO LOOP
      DO 5 I=1,MG
C                     ADVANCE
    1 M1=ME
      ME=NEXT(JCCORE(16),M1,MB,1,MC,MF,R,0,1)
      IF(ME.GT.0)GO TO 2
      IF(MF.GT.0)GO TO 1
      CALL SYSERR(1,I)
      ME=M1
      GO TO 5
C                     TEST BY-PASS
    2 IF((MC*(MC-MD-1)).GE.0)GO TO 1
C                     FLAGS
      CALL FRMCEL(JCCORE(MA),ME,D,F,2)
      MD=MC+1
      IF(D(1).NE.1)GO TO 4
C                     GROUP
      IF(D(2).EQ.1)GO TO 3
      CALL PUSH(JCTSPR,JCTEMP,ME,G,1,1)
      GO TO 1
C                     RECORD ADDRESS
    3 MD=MC
    4 D(1)=ME
      D(2)=0
      CALL STORE(JCCORE(JCSPUR),R, D,A,R,JCCORE(JCLOAD),1)
C                     END OF LOOP
    5 R(2)=1
C                     EXIT
      CALL SYSEXT
      RETURN
      END
```

```
C
C*****************************************************************
C                              **********
C                              * J2TADV *
C                              **********
C
C       PURPOSE
C           J2TADV PREPARES A TEAR OR DECISION VARIABLE LIST ENTRY FOR
C           A GROUP.
C
C       USAGE
C           J2TADV(JCVARI,JCFLAG,JCMODE,JCCLNO)
C
C       DATA FORMAT
C           N/A
C
C       DESCRIPTION OF PARAMETERS
C           JCVARI = A VARIABLE CODE NAME OR THE ADDRESS OF AN SIM
C                    AUXILIARY DATA BLOCK IF JCMODE=1
C           JCFLAG = 0 FOR ALGORITHMIC SELECTION AND 1 FOR PRECHOSEN
C           JCMODE = 0 FOR UNDIMENSIONED AND 1 OTHERWISE
C           JCCLNO = A USER ASSIGNED CALL STATEMENT IO. PARAMETER
C
C       REMARKS
C           THE NUMBER REPRESENTED BY JCTCIA MUST, FOR SOME MACHINES,
C           REPLACE THE 1 IN THE NAMED COMMON STATEMENT.
C
C       SUBROUTINE AND FUNCTION SUBPROGRAMS REQUIRED
C           NEWCEL
C           SYSENT
C           SYSEXT
C           TOCELL
C
C       ERROR CODES FOR THIS PROGRAM
C           NONE
C
C       METHOD
C           SELF-EXPLANATORY
C
C                              **********
C
      FUNCTION J2TADV(JCVARI,JCFLAG,JCMODE,JCCLNO)
      COMMON/ALLOC/JCCORE(1)
      DIMENSION D(3),V(10)
      INTEGER D,V
      DATA V/3,2,1,1,3,1,1,21,1,1/
C                    ENTER
      CALL SYSENT(1,4,2,6,JCCLNO)
C                    MA=DISPLACEMENT FOR OUTPUT
      M1=JCCORE(17)+10
      MA=JCCORE(M1)
C                    MB=ADDRESS OF SPUR FOR OUTPUT
      MB=JCCORE(16)+1
      MB=JCCORE(MB)+5*MA-5
      IF(JCMODE.EQ.0)GO TO 1
      M1=JCVARI+1
      MB=MB+5*JCCORE(M1)
C                    GET LIST ENTRY
    1 CALL NEWCEL(JCCORE(MB),J2TADV,1)
C                    STORE DATA
      D(1)=0
      D(2)=JCFLAG
```

```
      D(3)=JCVARI
      IF(JCMODE.EQ.0)GO TO 3
      D(1)=JCCORE(M1)
      D(3)=JCCORE(JCVARI)
      M2=JCTADV+MA
      M3=M2+JCCORE(M1)-1
      DO 2 I=M2,M3
      M1=M1+1
    2 JCCORE(I)=JCCORE(M1)
    3 CALL TOCELL(JCCORE(MB),J2TADV,0,V,1)
C                    EXIT
      CALL SYSEXT
      RETURN
      END
```

```
C
C*******************************************************************
C                              **********
C                              * BEMOAN *
C                              **********
C
C       PURPOSE
C           BEMOAN PERFORMS THE BARKLEY AND MOTARD MINIMUM TEAR
C           ALGORITHM.
C
C       USAGE
C           CALL BEMOAN(JCGRUP,JCMODE,JCCLNO)
C
C       DATA FORMAT
C           N/A
C
C       DESCRIPTION OF PARAMETERS
C           SEE SPEDUP
C
C       REMARKS
C           THE NUMBER REPRESENTED BY JCTCIA MUST, FOR SOME MACHINES,
C           REPLACE THE 1 IN THE NAMED COMMON STATEMENT.
C           MEMOAN EMPLOYS DFAULT AT OPTION LEVEL THREE.
C
C       SUBROUTINE AND FUNCTION SUBPROGRAMS REQUIRED
C           DFAULT
C           FETCH
C           FRMCEL
C           FSTLNK
C           J2CPOT
C           J2SPRE
C           J2TADV
C           J2TOWY
C           J7OSUP
C           RETURN
C           SIMRTV
C           SPEDUP
C           STORE
C           SYSENT
C           SYSEXT
C           TOCELL
C
C       ERROR CODES FOR THIS PROGRAM
C           NONE
C
C       METHOD
C           SELF-EXPLANATORY
C
C                              **********
C
      SUBROUTINE BEMOAN(JCGRUP,JCMODE,JCCLNO)
      COMMON/ALLOC/JCCORE(1)
      DIMENSION L(4),D(3),F(2),Z(10),G(4),T(7),N(7)
      INTEGER D,F,Z,G,T
      DATA L/3,5,1,1/,F/1,0/,Z/3,14,1,1,3,1,1,4,1,1/,G/1,7,3,3/,
     1 T/2,1,1,1,2,1,1/,N/2,7,1,1,9,1,1/
C                    ENTER
      CALL SYSENT(1,4,3,1,JCCLNO)
C                    MA=ADDRESS OF SPUR FOR GROUP
      M1=JCCORE(16)+1
      M1=JCCORE(M1)
      M2=JCCORE(17)+8
```

```
      MA=M1+5*JCCORE(M2)-5
C                     MB=ADDRESS OF SPUR FOR TEAR VARIABLE
      M2=M2+2
      MB=M1+5*JCCORE(M2)-5
C                     MC=ADDRESS OF SIMBL
      CALL DFAULT(JCGRUP,MC,0,3,1)
C                     MD=ADDRESS OF SPUR FOR ORDINATE
      M1=MC+2
      MD=JCCORE(M1)-5
C                     ME=ADDRESS OF LOAD FOR COLUMN - WORKING
      ME=J7OSUP(MC,5,1,0,1)
      MF=ME+7
C                     ESTABLISH WORKING ORDINATE
      MG=J7OSUP(MC,1,1,0,2)
      MH=MG+7
      M1=MG+8
      DO 1 I=1,3
    1 D(I)=0
      M2=JCCORE(M1)
      M1=M1+13
      F(2)=JCCORE(M1)
      M1=1
      DO 2 I=1,M2
      CALL STORE(JCCORE(MD),L(M1),D,Z,F,JCCORE(ME),1)
    2 M1=3
      L(1)=1
C                     RELEASE CURRENT TEAR VARIABLES
       CALL FRMCEL(JCCORE(MA),JCGRUP,M1,G,1)
      IF(M1.EQ.0)GO TO 4
    3 CALL FRMCEL(JCCORE(MB),M1,D(2),T,2)
      M2=MB+10*D(3)
      CALL RETURN(JCCORE(M2),M1,M1,1)
      M1=D(2)
      IF(M1.GT.0)GO TO 3
      CALL TOCELL(JCCORE(MA),JCGRUP,D,G,1)
C                     FIND INTERVAL WITH SINGLE PRECURSOR
    4 M1=1
      MI=0
    5 CALL J2SPRE(MC,ME,M2,M3,M1,1)
      IF(M2.EQ.0)GO TO 8
      IF(M2.EQ.M3)GO TO 7
C                     ASSIGN INTERVAL TO PRECURSOR
      D(1)=1
      D(2)=M3
      Z(1)=2
    6 L(2)=M2-JCCORE(MF)
      CALL STORE(JCCORE(MD),L,D,Z,F,JCCORE(ME),2)
      M1=M2+1
      GO TO 5
C                     MAKE INTERVAL A TEAR VARIABLE
    7 Z(1)=1
      L(2)=M2-JCCORE(MH)
      CALL FETCH(JCCORE(MD),L,D,N,L,JCCORE(MG),0,1)
      M1=J2TACV(D(2),0,D(1),1)
      CALL FSTLNK(JCCORE(MB),M1,MI,1)
      MI=M1
      D(1)=2
      GO TO 6
C                     FIND INTERVAL APPEARING IN THE MOST TWO-WAY EDGES
    8 M2=J2TOWY(MC,ME,1)
      IF(M2.GT.0)GO TO 7
C                     FIND INTERVAL APPEARING AS THE PRECURSOR TO THE
```

```
C                    LARGEST NUMBER OF INTERVALS
      M2=J2CPOT(MC,ME,1)
      IF(M2.GT.0)GO TO 7
C                    UPDATE JCGRUP
      CALL TOCELL(JCCORE(MA),JCGRUP,MI,G,2)
      Z(1)=3
C                    ADJUST SIM
      CALL SIMRTV(JCGRUP,1)
C                    PRECEDENCE ORDER
      CALL SPEOUP(JCGRUP,JCMODE,JCCLNO)
C                    EXIT
      CALL SYSEXT
      RETURN
      END
```

```
C
C*******************************************************************
C                              **********
C                              * J2CPOT *
C                              **********
C
C       PURPOSE
C           J2CPOT FINDS THE INTERVAL APPEARING IN THE MAXIMUM NUMBER
C           OF OTHER INTERVALS.
C
C       USAGE
C           J2CPOT(JCSPTR,JCLOOW,JCCLNO)
C
C       DATA FORMAT
C           N/A
C
C       DESCRIPTION OF PARAMETERS
C           SEE J2REDU
C
C       REMARKS
C           THE NUMBER REPRESENTED BY JCTCIA MUST, FOR SOME MACHINES,
C           REPLACE THE 1 IN THE NAMED COMMON STATEMENT.
C
C       SUBROUTINE AND FUNCTION SUBPROGRAMS REQUIRED
C           CORN
C           FETCH
C           FIND
C           FRMCEL
C           J2REDU
C           J7OSUP
C           LNKBWD
C           LNKFWD
C           LOCATE
C           POPUP
C           PUSH
C           SYSENT
C           SYSEXT
C
C       ERROR CODES FOR THIS PROGRAM
C           NONE
C
C       METHOD
C           SELF-EXPLANATORY
C
C                              **********
C
      FUNCTION J2CPOT(JCSPTR,JCLOOW,JCCLNO)
      COMMON/ALLOC/JCCORE(1)
      DIMENSION L(2),F(4),O(3),P(7),R(4),C(4),O(4),T(5),N(4)
      INTEGER F,O,P,R,C,O,T
      DATA L/1,0/,F/1,14,1,1/,O(1)/0/,P/2,13,1,1,6,1,1/,R/1,3,1,1/,C/1,
     1 4,1,1/,O/1,13,1,1/,T/0,0,1,3,3/,N/1,1,3,3/
C                     ENTER
      CALL SYSENT(1,4,3,2,JCCLNO)
C                     MA=ADDRESS OF LOAD(8) FOR COLUMN - WORKING
      MA=JCLOOW+7
C                     MB=ADDRESS OF LOAD FOR COLUMN - MAIN
      MB=J7OSUP(JCSPTR,1,1,0,1)
      MC=MB+7
C                     MD=ADDRESS OF LOAD FOR ROW - MAIN
      MD=J7OSUP(JCSPTR,1,0,0,2)
      ME=MD+7
```

```
C                   MF=ADDRESS OF SPUR FOR ELEMENT
      M1=JCSPTR+2
      JCSPRO=JCCORE(M1)-5
      MF=JCSPRO-5
C                   L
      L(2)=1-JCCORE(MA)
C                   INITIALIZE J2CPOT
      J2CPOT=0
C                   MG=CURRENT COUNT
      MG=0
C                   MH=MAXIMUM COUNT
      MH=0
C                   MI=TEMPORARY LIST HEADCELL
      MI=0
C                   MJ=CURRENT COLUMN
      MJ=0
C                   T
      T(1)=0
      T(2)=0
C                   FIND INTERVAL
    1 M1=1
      CALL FIND(JCCORE(JCSPRO),L,D,F,L,JCCORE(JCLODW),M1,1)
      IF(M1.EQ.0)GO TO 8
C                   ACCESS COLUMN
      L(2)=JCCORE(MA)-JCCORE(MC)
      MJ=JCCORE(MA)
      CALL FETCH(JCCORE(JCSPRO),L,D(2),P,L,JCCORE(MB),0,1)
C                   COLUMN ELEMENT
    2 CALL FRMCEL(JCCORE(MF),D(3),M1,R,1)
      IF(M1.EQ.D(2)) GO TO 6
C                   ACCESS ROW
      L(2)=M1-JCCORE(ME)
      CALL FETCH(JCCORE(JCSPRO),L,M1,P(4),L,JCCORE(MO),0,2)
C                   ROW ELEMENT
    3 CALL FRMCEL(JCCORE(MF),M1,L(2),C,2)
      L(2)=L(2)-JCCORE(MC)
      CALL FETCH(JCCORE(JCSPRO),L,M2,O,L,JCCORE(MB),0,3)
      IF(M2.NE.JCCORE(ME))GO TO 5
C                   REDUCTION
      M3=J2REDU(JCSPRO,JCLODW,JCCORE(MC),1)
C                   SEARCH T LIST
      IF(MI.EQ.0)GO TO 4
      CALL LOCATE(T,MI,M4,M3,N,1,1)
      IF(M4.NE.0)GO TO 5
C                   ADD TO T LIST
    4 CALL PUSH(T,MI,M3,N,1,1)
      MG=MG+1
C                   ADVANCE - ROW
    5 M1=LNKFWD(JCCORE(MF),M1,1)
      IF(M1.GT.0)GO TO 3
C                   ADVANCE - COLUMN
    6 D(3)=LNKBWD(JCCORE(MF),D(3),1)
      IF(D(3).GT.0)GO TO 2
C                   UPDATE
      IF(MG.LE.MH)GO TO 7
      MH=MG
      J2CPOT=MJ
C                   REINITIALIZE
    7 MG=0
      L(2)=MJ+1-JCCORE(MA)
      M5=0
      IF(MI.GT.0)CALL POPUP(T,MI,M5,M4,N,1,1)
```

```
      GO TO 1
    8 CALL CORN(T,1)
C                    EXIT
      CALL SYSEXT
      RETURN
      END
```

```
C
C*********************************************************************
C                                 **********
C                                 * J2ELEM *
C                                 **********
C
C       PURPOSE
C           J2ELEM USES THE OUTPUT OF A COLUMN TO FIND A PRECURSOR TO
C           THE COLUMN.
C
C       USAGE
C           J2ELEM(JCSPTR,JCLODW,JCNORA,JCMODE,JCROWN,JCCLNO)
C
C       DATA FORMAT
C           N/A
C
C       DESCRIPTION OF PARAMETERS
C           JCSPTR = THE POINTER TO THE SIMBL
C           JCNORA = A COLUMN NUMBER IF JCMODE = 0, OR THE ADDRESS OF
C                    AN ELEMENT IF JCMODE = 1
C           JCMODE = 0 TO ACCESS THE FIRST ROW ELEMENT AND 1 TO ACCESS
C                    THE NEXT ROW ELEMENT GIVEN THAT A ROW HAS ALREADY
C                    BEEN ACCESSED
C           JCROWN = THE ROW NUMBER
C           FOR ALL OTHER PASSED PARAMETERS SEE J2REDU
C
C       REMARKS
C           THE NUMBER REPRESENTED BY JCTCIA MUST, FOR SOME MACHINES,
C           REPLACE THE 1 IN THE NAMED COMMON STATEMENT.
C
C       SUBROUTINE AND FUNCTION SUBPROGRAMS REQUIRED
C           FETCH
C           FRMCEL
C           J2REDU
C           J70SUP
C           LNKFWD
C           SYSENT
C           SYSEXT
C
C       ERROR CODES FOR THIS PROGRAM
C           NONE
C
C       METHOD
C           JCMODE IS AUTOMATICALLY CHANGED FROM 0 TO 1.  J2ELEM
C           RETURNED AS 0 INDICATES FAILURE.
C
C                                 **********
C
      FUNCTION J2ELEM(JCSPTR,JCLODW,JCNORA,JCMODE,JCROWN,JCCLNO)
      COMMON/ALLOC/JCCORE(1)
      DIMENSION L(2),O(4),P(4),E(4)
      INTEGER O,P,E
      DATA L/1,0/,O/1,13,1,1/,P/1,6,1,1/,E/1,4,1,1/
C                 ENTER
      CALL SYSENT(1,4,3,3,JCCLNO)
C                 MA=ADDRESS OF LOAD FOR COLUMN - MAIN
      MA=J70SUP(JCSPTR,1,1,0,1)
      MB=MA+7
C                 MC=ADDRESS OF LOAD FOR ROW - MAIN
      MC=J70SUP(JCSPTR,1,0,0,2)
      MD=MC+7
C                 INITIALIZATION OF J2ELEM
```

```
      J2ELEM=0
C                    ME=ADDRESS OF SPUR FOR ELEMENT
      M1=JCSPTR+2
      JCSPRO=JCCORE(M1)-5
      ME=JCSPRO-5
C                    MODE
      IF(JCNORA.EQ.0)GO TO 4
      IF(JCMODE.EQ.1)GO TO 2
C                    GET OUTPUT OF COLUMN
      L(2)=JCNORA-JCCORE(MB)
      CALL FETCH(JCCORE(JCSPRO),L,M1,0,L,JCCORE(MA),0,1)
      IF(M1.EQ.0)GO TO 4
C                    ACCESS ROW
      L(2)=M1-JCCORE(MO)
      CALL FETCH(JCCORE(JCSPRO),L,JCNORA,P,L,JCCORE(MC),0,2)
C                    ACCESS ELEMENT
    1 IF(JCNORA.EQ.0)GO TO 4
      CALL FRMCEL(JCCORE(ME),JCNORA,M2,E,1)
C                    REDUCE M2
      M3=J2REDU(JCSPRO,JCLODW,M2,1)
      JCMODE=1
      IF(M3.GT.0)GO TO 3
C                    ADVANCE
    2 JCNORA=LNKFWD(JCCORE(ME),JCNORA,1)
      GO TO 1
C                    SUCCESS
    3 JCROWN=JCCORE(MO)
      J2ELEM=M3
C                    EXIT
    4 CALL SYSEXT
      RETURN
      END
```

```
C
C*****************************************************************
C                              **********
C                              * J2PATH *
C                              **********
C
C       PURPOSE
C           J2PATH DEVELOPS STEWARD PATHS.
C
C       USAGE
C           CALL J2PATH(JCSPTR,JCRNUM,JCCNUM,JCCLNO)
C
C       DATA FORMAT
C           N/A
C
C       DESCRIPTION OF PARAMETERS
C           SEE J2MARK
C
C       REMARKS
C           THE NUMBER REPRESENTED BY JCTCIA MUST, FOR SOME MACHINES,
C           REPLACE THE 1 IN THE NAMED COMMON STATEMENT.
C
C       SUBROUTINE AND FUNCTION SUBPROGRAMS REQUIRED
C           SEE J2MARK
C
C       ERROR CODES FOR THIS PROGRAM
C           NONE
C
C       METHOD
C           SELF-EXPLANATORY
C
C                              **********
C
      SUBROUTINE J2PATH(JCSPTR,JCRNUM,JCCNUM,JCCLNO)
      COMMON/ALLOC/JCCORE(1)
      DIMENSION F(7),O(7),R(4),C(2),Z(4),N(10),L(4),D(4)
      INTEGER F,O,R,C,Z,D
      DATA F/2,12,1,1,5,1,1/,O/2,12,1,1,13,1,1/,R/1,0,0/,C/1,0/,Z/1,6,1,
     1 1/,N/3,3,1,1,4,1,1,3,1,1/,L/1,1,1,1/
C                     ENTER
      CALL SYSENT(1,4,1,9,JCCLNO)
C                     MA=ADDRESS OF SPUR FOR ORDINATE
      M1=JCSPTR+2
      MA=JCCORE(M1)-5
C                     M8=ADDRESS OF LOAD FOR ROW - MAIN
      M8=J70SUP(JCSPTR,1,0,0,1)
C                     MC=ADDRESS OF LOAD FOR COLUMN - MAIN
      MC=J70SUP(JCSPTR,1,1,0,2)
C                     MD=ADDRESS OF LOAD FOR ROW - WORKING
      MD=J70SUP(JCSPTR,3,0,0,3)
C                     ME=ADDRESS OF LOAD FOR COLUMN - WORKING
      ME=J70SUP(JCSPTR,3,1,0,4)
C                     MF=ADDRESS OF LOAD(8) FOR ROW - MAIN
      MF=M8+7
C                     MG=ADDRESS OF LOAD(8) FOR COLUMN - MAIN
      MG=MC+7
C                     MH=ADDRESS OF LOAD(8) FOR ROW - WORKING
      MH=MD+7
C                     MI=ADDRESS OF LOAD(8) FOR COLUMN - WORKING
      MI=ME+7
C                     MJ=MARK
      MJ=0
```

```
C                    INITIALIZATION
      JCRNUM=0
      JCCNUM=0
C                    MK=INDICATOR
      MK=0
C                    ML=INDICATOR
      ML=0
C                    MM=ADDRESS OF SPUR FOR ELEMENT
      MM=MA-5
C                    FIND ROW WITH OUTPUT ASSIGNMENT
      D(1)=1
      D(2)=2
      R(2)=1-JCCORE(MF)
    1 M1=-1
      CALL FIND(JCCORE(MA),R,D,F,R,JCCORE(MB),M1,1)
      IF(M1.EQ.0)GO TO 14
C                    GET ROW MARK
      MO=JCCORE(MF)
      R(2)=JCCORE(MF)-JCCORE(MH)
      CALL FETCH(JCCORE(MA),R,M2,D(4),R,JCCORE(MO),0,1)
      MK=0
      IF(M2.GT.0)MK=2
C                    GET OUTPUT
      R(2)=0
      CALL FETCH(JCCORE(MA),R,D(3),0,R,JCCORE(MB),0,2)
      ML=0
      MN=D(4)
      IF(MN.EQ.0)GO TO 11
      IF(D(3).NE.1)GO TO 3
C                    MULTIPLE OUTPUT
    2 ML=JCCORE(MN)
      M2=MN+1
      MN=JCCORE(M2)
C                    GET COLUMN MARK
    3 C(2)=MN-JCCORE(MI)
      IF(MK.EQ.2)GO TO 4
      CALL FETCH(JCCORE(MA),C,M2,D(4),C,JCCORE(ME),0,3)
      IF(M2.GT.0)MK=1
      IF(MK.EQ.0)GO TO 11
C                    INITIALIZATION
    4 M1=MB
      M2=MC
      M3=MD
      M4=ME
      M5=MF
      M6=MG
      M7=MH
      M8=MI
      N(2)=3
      C(2)=MN-JCCORE(MG)
      IF(MK.EQ.2)GO TO 5
      M1=MC
      M2=MB
      M3=ME
      M4=MO
      M5=MG
      M6=MF
      M7=MI
      M8=MH
      N(2)=4
      C(2)=0
C                    FIND ELEMENT WITH ZERO COST
```

```
    5 CALL FETCH(JCCORE(MA),C,M9,Z,C,JCCORE(M2),0,4)
    6 L(2)=MK
      D(1)=0
      CALL LOCATE(JCCORE(MM),M9,M11,D,Z,MK,1)
      IF(M11.EQ.0)GO TO 7
C                     CHECK OUTPUT
      CALL FRMCEL(JCCORE(MM),M11,R(2),N,1)
      C(2)=R(3)-JCCORE(MG)
      CALL FETCH(JCCORE(MA),C,M12,D(4),C,JCCORE(MC),0,41)
      IF(M12.EQ.R(4))GO TO 61
C                     CHECK MARKING
      R(2)=R(2)-JCCORE(M7)
      CALL FETCH(JCCORE(MA),R,M12,D(4),R,JCCORE(M3),0,5)
      IF(M12.EQ.0)GO TO 8
   61 CALL FRMCEL(JCCORE(MM),M11,M9,L,2)
      IF(M9.GT.0)GO TO 6
C                     NO UNMARKED ELEMENT
    7 R(2)=1
      GO TO 1
C                     MARK
    8 M9=2-MK
      CALL SIMINT(JCSPTR,1,3,M9,1)
      CALL J70TPT(JCSPTR,JCCORE(M7),M9,1,JCCORE(M8),1)
      CALL SIMINT(JCSPTR,1,3,M9,2)
C                     OUTPUT
      M9=JCCORE(ME)
      R(2)=JCCORE(M7)-JCCORE(M5)
      CALL FETCH(JCCORE(MA),R,D(3),0,R,JCCORE(M1),0,6)
      IF(D(4).EQ.0)GO TO 11
      M10=0
      IF(D(3).EQ.0)GO TO 10
      M10=D(4)
    9 M11=M10+1
      M10=JCCORE(M10)
      D(4)=JCCORE(M11)
C                     MARKS
   10 C(2)=D(4)-JCCORE(M8)
      CALL FETCH(JCCORE(MA),C,M12,D(4),C,JCCORE(M4),0,7)
      IF(M12.GT.0)GO TO 13
C                     ITERATE
      IF(M10.GT.0)GO TO 9
   11 IF(ML.EQ.0)GO TO 12
      MN=ML
      GO TO 2
   12 R(2)=MO-JCCORE(MF)+1
      GO TO 1
C                     TRIPLE MARK
   13 JCRNUM=JCCORE(M5)
      JCCNUM=JCCORE(M8)
      IF(MK.EQ.2)GO TO 14
      JCRNUM=JCCNUM
      JCCNUM=JCCORE(M5)
C                     EXIT
   14 CALL SYSEXT
      RETURN
      END
```

```
C
C*******************************************************************
C                              **********
C                              * J2REDU *
C                              **********
C
C       PURPOSE
C           J2REDU REDUCES A PRECURSOR TO AN INTERVAL.
C
C       USAGE
C           J2REDU(JCSPRO,JCLODW,JCCNUM,JCCLNO)
C
C       DATA FORMAT
C           N/A
C
C       DESCRIPTION OF PARAMETERS
C           JCSPRO = THE ADDRESS OF THE SPUR FOR ORDINATE
C           JCLODW = THE ADDRESS OF THE LOAD FOR THE WORKING ORDINATE
C           JCCNUM = THE COLUMN NUMBER OF THE VARIABLE OF INTEREST
C           JCCLNO = A USER ASSIGNED CALL STATEMENT ID. PARAMETER
C
C       REMARKS
C           THE NUMBER REPRESENTED BY JCTCIA MUST, FOR SOME MACHINES,
C           REPLACE THE 1 IN THE NAMED COMMON STATEMENT.
C
C       SUBROUTINE AND FUNCTION SUBPROGRAMS REQUIRED
C           FETCH
C           SYSENT
C           SYSEXT
C
C       ERROR CODES FOR THIS PROGRAM
C           NONE
C
C       METHOD
C           SELF-EXPLANATORY
C
C                              **********
C
      FUNCTION J2REDU(JCSPRO,JCLODW,JCCNUM,JCCLNO)
      COMMON/ALLOC/JCCORE(1)
      DIMENSION L(2),P(7),D(2)
      INTEGER P,D
      DATA L/1,0/,P/2,14,1,1,3,1,1/
C                     ENTER
      CALL SYSENT(1,4,3,4,JCCLNO)
C                     MA=ADDRESS OF LOAD(8)
      MA=JCLODW+7
C                     L
      L(2)=JCCNUM
C                     INITIALIZE J2REDU
      J2REDU=JCCNUM
C                     ACCESS FILE ENTRY
    1 L(2)=L(2)-JCCORE(MA)
      M1=1
      CALL FETCH(JCCORE(JCSPRO),L,D,P,L,JCCORE(JCLODW),M1,1)
C                     MODE
      IF(M1.EQ.-1)GO TO 3
      IF(D(1).EQ.2)GO TO 2
      IF(D(2).EQ.0)GO TO 3
C                     ITERATE
      J2REDU=D(2)
      L(2)=D(2)
      GO TO 1
C     ----------END OF CHAIN AT TEAR VARIABLE
    2 J2REDU=0
C                     EXIT
    3 CALL SYSEXT
      RETURN
      END
```

```
C
C*******************************************************************
C                              **********
C                              * J2SPRE *
C                              **********
C
C       PURPOSE
C           J2SPRE FINDS AN INTERVAL WITH A SINGLE PRECURSOR.
C
C       USAGE
C           CALL J2SPRE(JCSPTR,JCLOCW,JCINTE,JCPREC,JCCNUM,JCCLNO)
C
C       DATA FORMAT
C           N/A
C
C       DESCRIPTION OF PARAMETERS
C           JCSPTR = THE POINTER TO THE SINBL
C           JCINTE = THE INTERVAL
C           JCPREC = THE SINGLE PRECURSOR OF JCINTE
C           JCCNUM = THE COLUMN NUMBER AT WHICH THE SEARCH IS TO
C                    COMMENCE
C           SEE J2REDU FOR ALL OTHER PASSED PARAMETERS
C
C       REMARKS
C           THE NUMBER REPRESENTED BY JCTCIA MUST, FOR SOME MACHINES,
C           REPLACE THE 1 IN THE NAMED COMMON STATEMENT.
C
C       SUBRCUTINE AND FUNCTION SUBPROGRAMS REQUIRED
C           FETCH
C           FIND
C           FRMCEL
C           J2ELEM
C           SYSENT
C           SYSEXT
C
C       ERROR CODES FOR THIS PROGRAM
C           NONE
C
C       METHOD
C           J2SPRE WILL PERFORM A SECOND SEARCH FROM THE BEGINNING OF
C           THE ORDINATE IF ON THE INITIAL PASS THE SEARCH FAILS.  IF
C           THE SECOND SEARCH ALSO FAILS, J2SPRE IS RETURNED AS ZERO.
C
```

```
C                                  **********
C
      SUBROUTINE J2SPRE(JCSPTR,JCLODW,JCINTE,JCPREC,JCCNUM,JCCLNO)
      COMMON/ALLOC/JCCORE(1)
      DIMENSION L(2),F(4),D(1),O(4),C(4)
      INTEGER F,D,O,C
      DATA L/1,0/,F/1,14,1,1/,D/0/,O/1,13,1,1/,C/1,4,1,1/
C                     ENTER
      CALL SYSENT(1,4,3,5,JCCLNO)
C                     MA=ADDRESS OF LOAD(8) FOR COLUMN - WORKING
      MA=JCLODW+7
C                     L
      L(2)=JCCNUM-JCCORE(MA)
C                     MB=CURRENT COLUMN
      MB=0
C                     MC=INDICATOR
      MC=0
C                     MD=COUNTER
      MD=0
C                     ME=PRECURSOR
      ME=0
C                     MF=ADDRESS OF LOAD FOR COLUMN - MAIN
      MF=J70SUP(JCSPTR,1,1,0,1)
      MG=MF+7
C                     JCSPRO
      M1=JCSPTR+2
      JCSPRO=JCCORE(M1)-5
C                     JCSPRS
      JCSPRS=JCSPRO-5
C                     INITIALIZATION
      JCINTE=0
      JCPREC=0
C                     FIND COLUMN WITH ZERO FLAG
    1 M1=1
      CALL FIND(JCCORE(JCSPRO),L,D,F,L,JCCORE(JCLODW),M1,1)
      IF(M1.EQ.0)GO TO 5
C                     J2ELEM
      MB=JCCORE(MA)
      M1=MB
      M2=0
    2 M3=J2ELEM(JCSPTR,JCLODW,M1,M2,M4,1)
      IF(M3.GT.0)GO TO 4
      GO TO 6
C                     ITERATE
    3 L(2)=MB+1-JCCORE(MA)
      MD=0
      ME=0
      GO TO 1
C                     TEST
    4 L(2)=M3-JCCORE(MG)
      CALL FETCH(JCCORE(JCSPRO),L,M5,O,L,JCCORE(MF),O,1)
      IF(M4.EQ.M5)GO TO 51
      IF((M5.EQ.0).OR.(M3.EQ.ME))GO TO 2
      MD=MD+1
      IF(MD.GT.1)GO TO 3
      IF(ME.EQ.0)ME=M3
      GO TO 2
C                     SEARCH FAILURE
    5 MC=MC+1
      IF(MC.GT.1)GO TO 7
      L(2)=1-JCCORE(MA)
      GO TO 1
```

```
C                    TEST FOR LEGITIMATE SELF-LOOP
   51 CALL FRMCEL(JCCORE(JCSPRS),M1,M5,C,1)
      IF(MB.EQ.M5)GO TO 2
      ME=MB
C                    SUCCESS
    6 JCINTE=MB
      JCPREC=ME
C                    EXIT
    7 CALL SYSEXT
      RETURN
      END
```

```
C
C*******************************************************************
C                                **********
C                                * J2TOWY *
C                                **********
C
C       PURPOSE
C           J2TOWY ENUMERATES TWO WAY EDGES AND SELECTS THE MAXIMUM.
C
C       USAGE
C           J2TOWY(JCSPTR,JCLODW,JCCLNO)
C
C       DATA FORMAT
C           N/A
C
C       DESCRIPTION OF PARAMETERS
C           SEE J2SPRE
C
C       REMARKS
C           THE NUMBER REPRESENTED BY JCTCIA MUST, FOR SOME MACHINES,
C           REPLACE THE 1 IN THE NAMED COMMON STATEMENT.
C
C       SUBROUTINE AND FUNCTION SUBPROGRAMS REQUIRED
C           FETCH
C           FIND
C           J2ELEM
C           STORE
C           SYSENT
C           SYSEXT
C
C       ERROR CODES FOR THIS PROGRAM
C           NONE
C
C       METHOD
C           SELF-EXPLANATORY
C
C                              **********
C
      FUNCTION J2TOWY(JCSPTR,JCLODW,JCCLNO)
      COMMON/ALLOC/JCCORE(1)
      DIMENSION L(2),C(4),D(1),F(4)
      INTEGER C,D,F
      DATA L/1,0/,C/1,4,1,1/,D/0/,F/1,14,1,1/
C                     ENTER
      CALL SYSENT(1,4,3,6,JCCLNO)
C                     JCSPRO
      M1=JCSPTR+2
      JCSPRO=JCCORE(M1)-5
C                     MA=ADDRESS OF LOAD(8) FOR COLUMN - WORKING
      MA=JCLODW+7
C                     MB=NUMBER OF COLUMNS
      MB=MA+1
      MB=JCCORE(MB)
C                     ZERO COUNTS
      L(2)=1-JCCORE(MA)
      DO 1 I=1,MB
      CALL STORE(JCCORE(JCSPRO),L,D,C,L,JCCORE(JCLODW),1)
    1 L(2)=0
C                     MC=PRIMARY
      MC=0
C                     MD=SECONDARY
      MD=0
```

```
C                       L
      L(2)=1-JCCORE(MA)
C                       ME=CURRENT MAXIMUM COUNT
      ME=0
C                       INITIALIZE J2TOWY
      J2TOWY=0
C                       FIND COLUMN WITH ZERO FLAG
    2 M1=1
      CALL FIND(JCCORE(JCSPRO),L,0,F,L,JCCORE(JCLODW),M1,1)
      IF(M1.EQ.0)GO TO 6
C                       PRIMARY
      MC=JCCORE(MA)
      L(2)=0
      CALL FETCH(JCCORE(JCSPRO),L,MF,C,L,JCCORE(JCLODW),0,1)
      IF(JCCORE(MA).EQ.M8)GO TO 5
      M1=MC
      M2=0
    3 M3=J2ELEM(JCSPTR,JCLODW,M1,M2,M4,1)
      IF(M3.EQ.0)GO TO 5
      IF(M3.LE.MC)GO TO 3
C                       SECONDARY
      MD=M3
      M5=0
    4 M6=J2ELEM(JCSPTR,JCLODW,M3,M5,M7,2)
      IF(M6.EQ.0)GO TO 3
      IF(M6.NE.MC)GO TO 4
C                       INCREMENT SECONDARY
      L(2)=MD-JCCORE(MA)
      CALL FETCH(JCCORE(JCSPRO),L,M8,C,L,JCCORE(JCLODW),0,2)
      M8=M8+1
      L(2)=0
      CALL STORE(JCCORE(JCSPRO),L,M8,C,L,JCCORE(JCLODW),2)
C                       INCREMENT PRIMARY
      MF=MF+1
      GO TO 3
C                       END PRIMARY
    5 L(2)=MC-JCCORE(MA)+1
      IF(MF.LE.ME)GO TO 2
      ME=MF
      J2TOWY=MC
      GO TO 2
C                       EXIT
    6 CALL SYSEXT
      RETURN
      END
```

```
C
C*****************************************************************
C                          **************
C                          *   SUPPAC     *
C                          *   GENERAL    *
C                          * INFORMATION  *
C                          **************
C
C       SPECIAL INSTRUCTIONS/LIST TYPE
C           ALL LIST TYPES EMPLOYED WITH SUPPAC MUST BE FORWARD-
C           BACKWARD.  SIMPAC USES LIST TYPES 0 (A REMOTE APPLICATION)
C           AND 4.  GALAP USES LIST TYPE 6.  NETPAC MAY BE USED WITH
C           ANY LIST TYPE.
C
C           SPECIAL INSTRUCTIONS/LEND
C           THE FOLLOWING DATA STRUCTURES MUST BE PERMITTED BY THE
C           LENDS.
C           SIMPAC - LIST TYPE 3
C               SEE LEND REQUIREMENTS FOR REMOTE.
C               (5 - STATUS FLAG, 6 - POINTER, 7 - DIMENSION FLAG, 8 -
C                   TYPE FLAG, 9 - CODE NAME, 10 - INCIDENCE COUNT,
C                   11 - NUMBER OF OUTPUTS, 12 - MULTIPLE OUTPUT FLAG,
C                   13 - OUTPUT)
C           SIMPAC - LIST TYPE 4
C               (1 - HORIZONTAL LINK, 2 - VERTICAL LINK, 3 - ROW
C                   NUMBER, 4 - COLUMN NUMBER, 5 - STATUS FLAG, 6 -
C                   ASSIGNMENT COST, 7 - SENSITIVITY)
C           NETPAC
C               (1 - LINK, 2 - LINK, 3 - ID. FLAG, 4 - KEY, 4 - KEY,
C                   5 - PATH COUNT, 5 - PATH COUNT, 5 - WORKING COUNT)
C           GALAP
C               SEE LEND REQUIREMENTS FOR NETPAC - INNER KEY NOT REQ.
C               (1, 2, 3, 4, 5, 5, 5, 6 - WORKING LEND, 7 - NID CODE,
C                   8 - VALUE OF CONSTANT, 9 - VARIABLE OF INTEREST
C                   FLAG, 9 - TYPE FLAG)
C
C       SPECIAL INSTRUCTIONS/DEBUG
C           THE FOLLOWING TABLE GIVES THE PACKAGE AND PROGRAM ID.
C           NUMBERS FOR SUPPAC.  ALL ARE ON LEVEL 3.
C           PROGRAM          PKG. ID./PROG. ID.
C           CUDGEL           1/1
C           SIMPAC
C           SIMCIN           2/1
C           SIMCPY           2/2
C           SIMDUP           2/3
C           SIMEIN           2/4
C           SIMFLG           2/5
C           SIMGEN           2/6
C           SIMINT           2/7
C           SIMOPT           2/8
C           SIMRTV           2/9
C           J7BODY           2/10
C           J7COMV           2/11
C           J71COM           2/12
C           J7LODV           2/13
C           J7OSUP           2/14
C           J7GTPT           2/15
C           J7VARI           2/16
C           NETPAC
C           BREAK            3/1
C           COUPLE           3/2
C           LNKBNT           3/3
```

```
C            LNKFNT          3/4
C            NEXT            3/5
C            REMOVE          3/6
C            SEARCH          3/7
C            J7LINK          3/8
C            DATIVE          4/1
C            GALAP
C            JORTRN          5/1
C            JOSVIF          5/2
C            MATH            5/3
C            JORLSE          5/5
C            JOCPND          5/6
C            JOMTCH          5/7
C            JONTPS          5/8
C            JOPSTN          5/9
C            JOCPCO          5/10
C            JOORDR          5/11
C            JORULS          5/12
C            JORULT          5/13
C            SOLVE           5/14
C
C       SURRENT STATUS
C           OPERATIONAL
C
C*****************************************************************
C
```

```
C
C*****************************************************************
C                              **********
C                              * CUDGEL *
C                              **********
C
C       PURPOSE
C           CUDGEL IS THE CRUDE GENDER LIST GENERATOR.
C
C       USAGE
C           CALL CUDGEL(JCUNIT,JCGRUP,JCCLNO)
C
C       DATA FORMAT
C           N/A
C
C       DESCRIPTION OF PARAMETERS
C           JCUNIT = UNIT NUMBER OR 0 FOR ALL UNITS IN SECEDE
C           JCGRUP = ADDRESS OF NEW GROUP IF JCUNIT = 0, OR THE
C                    CONTENTS OF JCCORE(16) OTHERWISE.
C           JCCLNO = A USER ASSIGNED CALL STATEMENT ID. PARAMETER
C
C       REMARKS
C           THE NUMBER REPRESENTED BY JCTCIA MUST BE SUPPLIED IN THE
C           COMMON STATEMENT FOR SOME MACHINES.
C
C       SUBROUTINE AND FUNCTION SUBPROGRAMS REQUIRED
C           FETCH
C           FSTLNK
C           J7ICOM
C           LNKFWD
C           NEWCEL
C           SETLNK
C           STORE
C           SYSENT
C           SYSERR
C           SYSEXT
C           TOCELL
C
C       ERROR CODES FOR THIS SUBPROGRAM
C           CODE FATAL  ERROR(DATA PROVIDED)
C           1    NF     JCCORE(16) NOT EMPTY (JCCORE(16))
C           2    F      NO UNIT (JCUNIT)
C           3    F      NO GROUP BODY (ADDRESS OF GROUP)
C
C       METHOD
C           CUDGEL GENERATES A GENDER LIST WITHOUT PARALLELISM.  EACH
C           UNIT OF SECEDE IS INCORPORATED INTO THE GENDER LIST.
C
C                              **********
C
      SUBROUTINE CUDGEL(JCUNIT,JCGRUP,JCCLNO)
      COMMON/ALLOC/JCCORE(1)
      DIMENSION P(4),S(4),N(4),T(10),F(4),V(4),B(4),G(16),O(7),L(9)
      INTEGER P,S,T,F,V,B,G,O
      DATA P/1,1,1,1/,S/1,1,2,3/,N/1,1,4,0/,T/3,21,1,1,2,1,1,3,1,1/,
     1 F/1,3,1,1/,V/1,7,2,2/,B/1,7,1,1/,G/5,20,1,1,19,1,1,17,1,1,3,1,2,
     2 5,1,2/
C                 ENTER
      CALL SYSENT(1,3,1,1,JCCLNO)
C                 BEGIN
      B(2)=7
      M1=JCCORE(15)+1
```

```
      MAA=JCCCRE(M1)
      M1=M1+1
      MAB=JCCCRE(M1)
      M1=MAB+28
      JCCCRE(M1)=-1
      JCGRUP=0
      ML=0
      MA=1
      MD=JCCORE(16)
      IF(MD.EQ.0)GO TO 1
      M1=MD+1
      ME=JCCORE(M1)
      IF(JCUNIT.EQ.0)CALL SYSERR(1,MD)
      JCGRUP=MD
      GO TO 2
    1 CALL J1GMBL(MD,ME,MX,1)
      JCGRUP=0
    2 M1=JCCCRE(17)+8
      MB=ME+5*(JCCORE(M1)+1)
      M1=M1+1
      MC=ME+5*(JCCORE(M1)-1)
      MCC=JCCCRE(M1)
      IF(JCUNIT.GT.0)MA=JCUNIT
C                     GROUP
    3 CALL NEWCEL(JCCORE(MB),MF,1)
      CALL TCCELL(JCCORE(MB),MF,P,T(7),1)
      IF(JCUNIT.EQ.0)JCGRUP=MF
      JCCCRE(MD)=MF
C                     ACCESS RECORD 1
      L(6)=1
      M1=JCCCRE(17)+6
      MG=1
      MH=JCCORE(M1)
      MK=JCCORE(17)+10
      MK=JCCORE(MK)
C                     ACCESS DECISION VARIABLE FILE
      MJ=0
    5 L(1)=4
      L(2)=1
      L(3)=1
      L(4)=1
      MI=1
      S(3)=1
      O(4)=1
C                     INSPECT DECISION VARIABLE
   51 CALL FETCH(JCCORE(MAA),L(MI),O,S,L,JCCORE(MAB),MG,1)
      IF(MG.EQ.-1)GO TO 8
      M1=D(2)/MH
      IF((M1.NE.0).AND.(M1.NE.MA))GO TO 71
C                     ADD DECISION VARIABLE TO LIST
      M1=MK+2*D(3)
      M2=ME+5*(M1-1)
      CALL NEWCEL(JCCORE(M2),M3,2)
      CALL TCCELL(JCCORE(M2),M3,D(2),T,2)
      IF(D(3).EQ.0)GO TO 52
      N(4)=3+D(3)
      M1=M3+MK
      L(7)=0
      CALL FETCH(JCCORE(MAA),L(6),JCCORE(M1),N,L,JCCORE(MAB),O,2)
C                     SET LINKS
   52 IF(MJ.NE.0)GO TO 6
      CALL TCCELL(JCCORE(MB),MF,M3,V,3)
```

```
      GO TO 7
    6 CALL FSTLNK(JCCORE(M2),MJ,M3,1)
    7 MJ=M3
   71 M1=6
      L(7)=D(1)
      IF(L(7).GT.0)GO TO 51
C                     ACCESS UNIT
    8 L(5)=1
      S(3)=2
      D(4)=2
      D(6)=1
      L(3)=5
      L(7)=0
      L(2)=MA+4
      L(1)=3
      M1=1
      CALL FETCH(JCCORE(MAA),L,D,P,L,JCCORE(MAB),M1,3)
      IF(M1.EQ.-1)GO TO 17
    9 L(3)=L(3)+1
      D(5)=L(3)-6+(L(3)/7)+(L(3)/8)*2
      L(4)=1
      L(1)=4
      M1=1
      CALL FETCH(JCCORE(MAA),L,D,P,L,JCCORE(MAB),M1,4)
      IF(M1.EQ.-1)GO TO 16
C                     ACCESS ER,FUNCTION OR CONSTRAINT
   10 L(1)=5
      M1=1
      CALL FETCH(JCCORE(MAA),L,D,S,L,JCCORE(MAB),M1,5)
      IF(M1.EQ.-1)GO TO 16
      D(3)=MA*MH+L(4)
C                     ACQUIRE BROUP BODY ENTRY
   11 M2=ML
      ML=1
      CALL J7ICOM(MAA,3,ME,ML,MCC,3,D(2),MAB,M5,1)
      IF(M5.EQ.-1)GO TO 15
C                     SET LINKS
   12 IF(M2.GT.0)GO TO 13
      D(7)=0
      CALL TOCELL(JCCORE(MB),MF,ML,8,4)
      GO TO 14
   13 D(7)=1
      CALL SETLNK(JCCORE(MC),M2,ML,1)
C                     STORE CATA
   14 CALL TOCELL(JCCORE(MC),ML,D,G,5)
      IF(D(2).EQ.0)GO TO 15
      MR=-3
      GO TO 11
C                     NEXT ER, FUNCTION OR CONSTRAINT
   15 L(4)=L(4)+1
      GO TO 10
C                     AOVANCE TO NEXT FILE
   16 IF(L(3).LT.8)GO TO 9
C                     ADVANCE TO NEXT UNIT
      IF(MA.EQ.JCUNIT)GO TO 18
      MA=MA+1
      IF(MG.GT.0)GO TO 5
      GO TO 8
C                     LAST UNIT
   17 IF(MA.EQ.JCUNIT)CALL SYSERR(-2,MA)
   18 B(2)=5
      D(1)=0
```

```
      IF(ML.EQ.0)CALL SYSERR(-3,MF)
      IF(ML.GT.0)CALL TOCELL(JCCORE(MC),ML,D,B,8)
C                 EXIT
      CALL SYSEXT
      RETURN
      END
```

```
C
C*****************************************************************
C                          **********
C                          * SIMCIN *
C                          **********
C
C       PURPOSE
C           SIMCIN ADJUSTS THE COUNT ORDINATES TO REFLECT THE REMOVAL
C           OR REINSTATEMENT OF A ROW OR COLUMN.
C
C       USAGE
C           CALL SIMCIN(JCSPTR,JCENUM,JCRORC,JCSENS,JCMODE,LION,LOCAL,
C           JCCLNO)
C
C       DATA FORMAT
C           N/A
C
C       DESCRIPTION OF PARAMETERS
C           JCSPTR = THE ADDRESS OF THE SIMBL
C           JCENUM = THE ORDINATE ENTRY NUMBER OF THE SUBJECT ROW OR
C                    COLUMN
C           JCRORC = 0 FOR ROW AND 1 FOR COLUMN
C           JCSENS = +1 FOR REINSTATEMENT OR -1 FOR REMOVAL
C           JCMODE = 0, 1 OR -1.  WHEN EQUAL TO 0, SIMCIN OPERATES ON
C                    ELEMENTS WITH A STATUS FLAG OF 0.  WHEN NOT EQUAL
C                    TO 0, SIMCIN OPERATES ON ELEMENTS WITH 0 STATUS
C                    FLAG AND DATA ITEMS WHICH MATCH THOSE IN LION.
C                    WHEN POSITIVE, JCMODE REQUIRES ALL DATA ITEMS TO
C                    MATCH LION.  WHEN NEGATIVE, JCMODE REQUIRES ONLY
C                    ONE DATA TYPE TO MATCH.
C           LION   = A VECTOR OF DATA ITEM VALUES
C           LOCAL  = SEE COPY
C           JCCLNO = A USER ASSIGNED CALL STATEMENT ID. PARAMETER
C
C       REMARKS
C           THE NUMBER REPRESENTED BY JCTCIA MUST, FOR SOME MACHINES,
C           REPLACE THE 1 IN THE NAMED COMMON STATEMENT.
C
C       SUBROUTINE AND FUNCTION SUBPROGRAMS REQUIRED
C           FETCH
C           FRMCEL
C           J5COMP
C           J7OSUP
C           STORE
C           SYSENT
C           SYSERR
C           SYSEXT
C
C       ERROR CODES FOR THIS PROGRAM
C           CODE FATAL  ERROR(DATA PROVIDED)
C           1    NF     COUNT ALREADY ZERO (ENTRY NUMBER)
C           2    NF     NEGATIVE COUNT FOR JCENUM (JCENUM)
C
C       METHOD
C           SIMCIN BREAKS NOR SETS ANY LINKS.  REINSTATEMENT AND
C           REMOVAL ARE ONLY APPARENT.  SIMCIN OPERATES ONLY ON
C           ELEMENTS WITH A STATUS FLAG OF 0.
C
C                          **********
C
      SUBROUTINE SIMCIN(JCSPTR,JCENUM,JCRORC,JCSENS,JCMODE,LION,LOCAL,
     1 JCCLNO)
```

```
      COMMON/ALLOC/JCCORE(1)
      DIMENSION L(2),D(3),P(10),C(4),LION(1),LOCAL(1)
      INTEGER D,P,C
      DATA L/1,0/,P/3,5,1,1,6,1,1,0,1,1/,C/1,10,1,1/
C                     ENTER
      CALL SYSENT(1,3,2,1,JCCLNO)
      P(5)=6
      P(8)=10
      MA=JCSPTR+2
      MA=JCCORE(MA)-5
      MB=MA-5
      MC=J70SUP(JCSPTR,1,JCRORC,0,1)
      M2=MC+7
      L(2)=JCENUM-JCCORE(M2)
C                     RETRIEVE POINTER AND STATUS FLAG
      CALL FETCH(JCCORE(MA),L,D,P,L,JCCORE(MC),0,1)
      IF(D(1).NE.0)GO TO 4
C                     TRACE CHAIN
      M1=1-JCRORC
      MF=J70SUP(JCSPTR,1,M1,0,2)
      MD=D(3)
      ME=MF+7
      P(1)=3
      P(5)=JCRORC+1
      P(8)=4-JCRORC
C                     EXAMINE ELEMENT
    1 IF(D(2).EQ.0)GO TO 3
      M3=D(2)
      CALL FRMCEL(JCCORE(MB),M3,D,P,1)
      IF(D(1).NE.0)GO TO 1
      IF(JCMODE.EQ.0)GO TO 11
      M1=JCMODE
      CALL J5COMP(JCCORE(MB),M3,LION,LOCAL,M1,M2,1)
      IF(M2.EQ.0)GO TO 1
   11 MD=MD+JCSENS
      L(2)=D(3)-JCCORE(ME)
      D(1)=0
      CALL FETCH(JCCORE(MA),L,D,C,L,JCCORE(MF),0,2)
      D(1)=D(1)+JCSENS
      L(2)=0
      IF(D(1).GE.0)GO TO 2
      D(1)=0
      CALL SYSERR(1,D(3))
    2 CALL STORE(JCCORE(MA),L,D,C,L,JCCORE(MF),1)
      GO TO 1
C                     SET CHAIN COUNT
    3 IF(MD.GE.0)GO TO 31
      MD=0
      CALL SYSERR(2,JCENUM)
   31 L(2)=0
      CALL STORE(JCCORE(MA),L,MD,C,L,JCCORE(MC),2)
C                     EXIT
    4 CALL SYSEXT
      RETURN
      END
```

```
C
C*******************************************************************
C                              **********
C                              * SIMCPY *
C                              **********
C
C       PURPOSE
C           SIMCPY TRANSFERS DATA FROM ONE ORDINATE TO ANOTHER PARALLEL
C           ORDINATE.
C
C       USAGE
C           CALL SIMCPY(JCSPTR,JCSNUM,JCRNUM,JCRORC,LOCALS,LOCALR,
C           JCFINE,JCCLNO)
C
C       DATA FORMAT
C           N/A
C
C       DESCRIPTION OF PARAMETERS
C           JCSPTR = THE ADDRESS OF THE SIMBL
C           JCSNUM = THE NUMBER OF THE SOURCE ORDINATE
C           JCRNUM = THE NUMBER OF THE RECEIVER ORDINATE
C           JCRORC = 0 FOR ROW, 1 FOR COLUMN AND 2 FOR BOTH ROW AND
C                    COLUMN
C           LOCALS = THE LOCAL VECTOR FOR THE SOURCE
C           LOCALR = THE LOCAL VECTOR FOR THE RECEIVER
C           JCFINE = THE FINE VECTOR FOR THE RECEIVER
C           JCCLNO = A USER ASSIGNED CALL STATEMENT ID. PARAMETER
C
C       REMARKS
C           THE NUMBER REPRESENTED BY JCTCIA MUST, FOR SOME MACHINES,
C           REPLACE THE 1 IN THE NAMED COMMON STATEMENT.
C           SIMCPY CAN TRANSFER UP TO 15 DATA ITEMS AT A TIME.  THE
C           LOCAL VECTORS SHOULD NOT SPECIFY MORE THAN 15 DATA ITEMS.
C
C       SUBROUTINE AND FUNCTION SUBPROGRAMS REQUIRED
C           FETCH
C           J7OSUP
C           STORE
C           SYSENT
C           SYSEXT
C
C       ERROR CODES FOR THIS PROGRAM
C           NONE
C
C       METHOD
C           SELF-EXPLANATORY
C
C                              **********
C
      SUBROUTINE SIMCPY(JCSPTR,JCSNUM,JCRNUM,JCRORC,LOCALS,LOCALR,
     1 JCFINE,JCCLNO)
      COMMON/ALLOC/JCCORE(1)
      DIMENSION S(2),R(4),D(15),LOCALS(1),LOCALR(1),JCFINE(1)
      INTEGER S,R,D
      DATA S/1,0/,R/3,0,1,0/
C                 ENTER
      CALL SYSENT(1,3,2,2,JCCLNO)
C                 MA=ADDRESS OF SPUR FOR ORDINATE
      M1=JCSPTR+2
      MA=JCCORE(M1)-5
C                 MB=ROW/COLUMN INDICATOR
      MB=JCRORC-2*(JCRORC/2)
```

```
C                   MC=ADDRESS OF LOAD FOR SOURCE
    1 MC=J70SUP(JCSPTR,JCSNUM,MB,0,1)
C                   S
      M1=MC+7
      S(2)=1-JCCORE(M1)
C                   MD=ADDRESS OF LOAD FOR RECEIVER
      MD=J70SUP(JCSPTR,JCRNUM,MB,0,2)
C                   R
      M1=MD+14
      IF(JCCORE(M1).NE.-1)GO TO 2
      ME=1
      R(2)=JCRNUM
      GO TO 3
    2 ME=3
      M1=MD+7
      R(4)=1-JCCORE(M1)
C                   MF=LIMIT
    3 M1=MC+8
      MF=JCCORE(M1)
C                   DATA TRANSFER
      DO 4 I=1,MF
      CALL FETCH(JCCORE(MA),S,0,LOCALS,S,JCCORE(MC),0,1)
      CALL STORE(JCCORE(MA),R(ME),0,LOCALR,JCFINE,JCCORE(MD),1)
      ME=3
      S(2)=1
    4 R(4)=1
C                   CHECK MB
      MB=MB+1
      IF(MB.LT.JCRORC)GO TO 1
C                   EXIT
      CALL SYSEXT
      RETURN
      END
```

```
C
C*****************************************************************
C                              **********
C                              * SIMDUP *
C                              **********
C
C       PURPOSE
C           SIMDUP DUPLICATES THE CONTENTS OF ONE ORDINATE IN A SECOND
C           PARALLEL ORDINATE.
C
C       USAGE
C           CALL SIMDUP(JCSPTR,JCMSTR,JCDUPL,JCRORC,JCCLNO)
C
C       DATA FORMAT
C           N/A
C
C       DESCRIPTION OF PARAMETERS
C           JCSPTR = THE ADDRESS OF THE SIMBL
C           JCMSTR = CATALOGUE ENTRY NUMBER OF THE MASTER ORDINATE
C           JCDUPL = CATALOGUE ENTRY NUMBER OF THE ORDINATE TO RECEIVE
C                    THE CONTENTS OF THE MASTER
C           JCRORC = 0 FOR VERTICAL AND 1 FOR HORIZONTAL ORDINATES
C           JCCLNO = A USER ASSIGNED CALL STATEMENT ID. PARAMETER
C
C       REMARKS
C           THE NUMBER REPRESENTED BY JCTCIA MUST, FOR SOME MACHINES,
C           REPLACE THE 1 IN THE NAMED COMMON STATEMENT.
C
C       SUBROUTINE AND FUNCTION SUBPROGRAMS REQUIRED
C           FETCH
C           J7OSUP
C           NEWCEL
C           RETURN
C           STORE
C           SYSENT
C           SYSEXT
C           TOCELL
C
C       ERROR CODES FOR THIS PROGRAM
C           NONE
C
C       METHOD
C           SIMDUP DUPLICATES THE CONTENTS OF EACH MASTER ORDINATE
C           SEGMENT IN THE CORRESPONDING DUPLICATE ORDINATE SEGMENT.
C           DATA TRANSFER IS WORD FOR WORD.
C
C                              **********
C
      SUBROUTINE SIMDUP(JCSPTR,JCMSTR,JCDUPL,JCRORC,JCCLNO)
      COMMON/ALLOC/JCCORE(1)
      DIMENSION W(4),F(4),L(4)
      INTEGER W,F
      DATA W/1,1,1,1/,L/3,0,1,1/
C                 ENTER
      CALL SYSENT(1,3,2,3,JCCLNO)
C                 SETTING UP PARAMETERS
      MA=JCSPTR+2
      MA=JCCORE(MA)-5
      MB=J7OSUP(JCSPTR,JCMSTR,JCRORC,0,1)
      MC=J7OSUP(JCSPTR,JCDUPL,JCRORC,0,2)
C                 ACCESS MASTER ORDINATE
      L(2)=JCMSTR
```

```
      CALL FETCH(JCCORE(MA),L,M1,W,W,JCCORE(MB),0,1)
C                   ADJUST PRE-FILE PARAMETERS
      MD=MB+8
      ME=MB+9
      MH=MB+28
      JCCCRE(MH)=-1
      F(1)=JCCORE(MD)
      F(2)=JCCORE(ME)
      JCCCRE(MD)=JCCORE(MD)*JCCORE(ME)
      JCCORE(ME)=1
      F(3)=JCCORE(MD)
      F(4)=1
C                   ACCESS DUPLICATE ORDINATE
      L(2)=JCCUPL
      CALL STORE(JCCORE(MA),L,M1,W,F(3),JCCORE(MC),1)
C                   ADJUST PRE-FILE PARAMETERS
      MF=MC+8
      MG=MC+9
      MI=MC+28
      JCCORE(MI)=-1
      JCCORE(MF)=JCCORE(MF)*JCCORE(MG)
      JCCORE(MG)=1
C                   SECURE ORDINATE SEGMENT SIZE BLOCK OF CORE
      CALL NEWCEL(JCCORE(MA),MJ,1)
C                   PREPARE FOR TRANSFER
      M1=MA+3
      M2=MB+21
      M2=JCCORE(M1)-JCCORE(M2)-3
      M3=JCCORE(MD)-1
      L(4)=1
C                   FETCH DATA
    1 IF(M2.GT.M3)M2=M3
      M3=M3-M2
      W(4)=M2
      CALL FETCH(JCCORE(MA),L(3),JCCORE(MJ),W,F,JCCORE(MB),0,2)
C                   STORE DATA
      CALL STORE(JCCORE(MA),L(3),JCCORE(MJ),W,F,JCCORE(MC),2)
C                   ADVANCE TO NEXT ORDINATE SEGMENT
      L(4)=M2
      M2=JCCCRE(M1)
      IF(M3.GT.0)GO TO 1
C                   SET PRE-FILE PARAMETERS IN DUPLICATE ORDINATE
      M1=MC+10
      M2=MC+15
      CALL TOCELL(JCCORE(MA),JCCORE(M1),F,JCCORE(M2),1)
      M2=MC+22
      CALL TOCELL(JCCORE(MA),JCCORE(M1),F(2),JCCORE(M2),2)
C                   RE-SET PRE-FILE PARAMETERS IN LOAD VECTORS
      JCCORE(MD)=F(1)
      JCCCRE(ME)=F(2)
      JCCORE(MF)=F(1)
      JCCCRE(MG)=F(2)
      JCCORE(MH)=0
      JCCORE(MI)=0
      W(4)=1
      CALL FETCH(JCCORE(MA),L,M1,W,F,JCCORE(MC),0,3)
      L(2)=JCMSTR
      CALL FETCH(JCCORE(MA),L,M1,W,F,JCCORE(MB),0,4)
C                   RETURN TEMPORARY STORAGE
      CALL RETURN(JCCORE(MA),MJ,MJ,1)
C                   EXIT
      CALL SYSEXT
```

```
RETURN
END
```

```
C
C*******************************************************************
C                              **********
C                              * SIMEIN *
C                              **********
C
C
C       PURPOSE
C           SIMEIN SETS ALL NON-UNITY ROW, COLUMN AND ELEMENT STATUS
C           FLAGS TO ZERO.
C
C       USAGE
C           CALL SIMEIN(JCSPTR,JCCLNO)
C
C       DATA FORMAT
C           N/A
C
C       DESCRIPTION OF PARAMETERS
C           JCSPTR = ADDRESS OF THE SIMBL
C           JCCLNO = A USER ASSIGNED CALL STATEMENT ID. PARAMETER
C
C       REMARKS
C           THE NUMBER REPRESENTED BY JCTCIA MUST, FOR SOME MACHINES,
C           REPLACE THE 1 IN THE NAMED COMMON STATEMENT.
C
C       SUBROUTINE AND FUNCTION SUBPROGRAMS REQUIRED
C           FETCH
C           FRMCEL
C           STORE
C           SYSENT
C           SYSEXT
C           TOCELL
C
C       ERROR CODES FOR THIS PROGRAM
C           NONE
C
C       METHOD
C           SIMEIN REINITIALIZES THE ROW AND COLUMN STATUS FLAGS FIRST.
C           EACH ROW IS TRACED.  ALL ELEMENTS ENCOUNTERED WITH A STATUS
C           FLAG GREATER THAN UNITY ARE FITTED WITH ZERO FLAGS.
C
C                              **********
C
      SUBROUTINE SIMEIN(JCSPTR,JCCLNO)
      COMMON/ALLOC/JCCORE(1)
      DIMENSION P(7),E(7),D(2),L(4)
      INTEGER P,E,D
      DATA P/2,6,1,1,5,1,1/,E/2,1,1,1,5,1,1/,L/3,1,1,0/
C                     ENTER
      CALL SYSENT(1,3,2,4,JCCLNO)
C                     SETTING UP PARAMETERS
      MA=JCSPTR+2
      MA=JCCORE(MA)-5
      MB=MA-5
      M1=1
      MC=J70SUP(JCSPTR,1,1,0,1)
      M2=MC+7
      M3=MC+8
      L(4)=0
C                     RESET COLUMN STATUS FLAGS
    1 CALL FETCH(JCCORE(MA),L(M1),D,P(4),L,JCCORE(MC),0,1)
      IF(D(1).GT.1)CALL STORE(JCCORE(MA),L(3),L(4),P(4),L,JCCORE(MC),1)
      M1=2
```

```
      IF(JCCORE(M2).LT.JCCORE(M3))GO TO 1
C                     RESET PARAMETERS
      M1=1
      MC=J7OSUP(JCSPTR,1,0,0,2)
      M2=MC+7
      M3=MC+8
C                     RESET ROW STATUS FLAG
    2 CALL FETCH(JCCORE(MA),L(M1),D,P,L,JCCORE(MC),0,2)
      IF(D(2).GT.1)CALL STORE(JCCORE(MA),L(3),L(4),P(4),L,JCCORE(MC),2)
      M1=2
C                     RESET ROW ELEMENT STATUS FLAG
    3 IF(D(1).EQ.0)GO TO 4
      M4=D(1)
      CALL FRMCEL(JCCORE(MB),M4,D,E,1)
      IF(D(2).GT.1)CALL TOCELL(JCCORE(MB),M4,L(4),E(4),1)
      GO TO 3
C                     LAST ROW
    4 IF(JCCORE(M2).LT.JCCORE(M3))GO TO 2
C                     EXIT
      CALL SYSEXT
      RETURN
      END
```

```
C
C*****************************************************************
C                              **********
C                              * SIMFLG *
C                              **********
C
C       PURPOSE
C           SIMFLG SETS THE ORDINATE AND ELEMENT STATUS FLAGS TO A
C           SPECIFIED VALUE FOR A ROW OR A COLUMN.
C
C       USAGE
C           CALL SIMFLG(JCSPTR,JCENUM,JCRORC,JCFLAG,JCMODE,JCCLNO)
C
C       DATA FORMAT
C           N/A
C
C       DESCRIPTION OF PARAMETERS
C           JCSPTR = THE POINTER TO THE SIMBL
C           JCENUM = THE ORDINATE ENTRY NUMBER
C           JCRORC = 0 FOR ROW AND 1 FOR COLUMN
C           JCFLAG = THE NEW FLAG VALUE
C           JCMODE = 0 OR 1 AS RESPECTIVELY AN ELEMENT STATUS FLAG IS
C                    ALWAYS RESET OR ONLY RESET IF THE STATUS FLAG
C                    ORTHOGONAL TO JCRORC IS UNEQUAL TO THE CURRENT
C                    ELEMENT STATUS FLAG
C           JCCLNO = A USER ASSIGNED CALL STATEMENT ID. PARAMETER
C
C       REMARKS
C           THE NUMBER REPRESENTED BY JCTCIA MUST, FOR SOME MACHINES,
C           REPLACE THE 1 IN THE NAMED COMMON STATEMENT.
C
C       SUBROUTINE AND FUNCTION SUBPROGRAMS REQUIRED
C           FETCH
C           FRMCEL
C           J70SUP
C           STORE
C           SYSENT
C           SYSERR
C           SYSEXT
C           TOCELL
C
C       ERROR CODES FOR THIS PROGRAM
C           CODE FATAL  ERROR(DATA PROVIDED)
C           1    NF     ENTRY STATUS FLAG IS UNITY (ENTRY NUMBER)
C
C       METHOD
C           NO UNITY STATUS FLAG IS ALTERED.
C
C                              **********
C
      SUBROUTINE SIMFLG(JCSPTR,JCENUM,JCRORC,JCFLAG,JCMODE,JCCLNO)
      COMMON/ALLOC/JCCORE(1)
      DIMENSION L(2),D(3),P(10)
      INTEGER D,P
      DATA L/1,0/,P/2,6,1,1,5,1,1,3,1,1/
C                 ENTER
      CALL SYSENT(1,3,2,5,JCCLNO)
C                 INITIALIZE SPUR POINTER
      MA=JCSPTR+2
      MA=JCCORE(MA)-5
C                 INITIALIZE L VECTOR
      MB=J70SUP(JCSPTR,1,JCRORC,0,1)
      M2=MB+7
```

```
      L(2)=JCENUM-JCCORE(M2)
      M2=1-JCRORC
      MC=J7OSUP(JCSPTR,1,M2,0,2)
      MD=MC+7
      ME=MA-5
C                    SET ORDINATE ENTRY FLAG
      P(1)=2
      P(2)=6
      CALL FETCH(JCCORE(MA),L,O,P,L,JCCORE(MB),0,1)
      L(2)=0
      IF(O(2).NE.1)GO TO 1
      CALL SYSERR(1,JCENUM)
      GO TO 2
    1 CALL STORE(JCCORE(MA),L,JCFLAG,P(4),L,JCCORE(MB),1)
    2 IF(O(1).GT.0)GO TO 3
      GO TO 5
C                    SET ELEMENT STATUS FLAGS
    3 P(2)=JCRORC+1
      P(1)=2+JCMODE
      P(8)=4-JCRORC
    4 M1=O(1)
      CALL FRMCEL(JCCORE(ME),M1,O,P,1)
      IF(JCMODE.EQ.0)GO TO 41
      L(2)=O(3)-JCCORE(MD)
      CALL FETCH(JCCORE(MA),L,M2,P(4),L,JCCORE(MC),0,2)
      IF(M2.EQ.O(2))O(2)=1
   41 IF(O(2).NE.1)CALL TOCELL(JCCORE(ME),M1,JCFLAG,P(4),1)
      IF(O(1).GT.0)GO TO 4
C                      EXIT
    5 CALL SYSEXT
      RETURN
      END
```

```
C
C*****************************************************************
C                              **********
C                              * SIMGEN *
C                              **********
C
C       PURPOSE
C           SIMGEN GENERATES AN SIM FOR A SPECIFIED GENDER GROUP.
C
C       USAGE
C           CALL SIMGEN(JCGRUP,JCSPTR,JCCLNO)
C
C       DATA FORMAT
C           N/A
C
C       DESCRIPTION OF PARAMETERS
C           JCGRUP = THE ADDRESS OF THE GENDER GROUP
C           JCSPTR = THE ADDRESS OF THE SIMBL
C           JCCLNO = A USER SPECIFIED CALL STATEMENT ID. NUMBER
C
C       REMARKS
C           THE NUMBER REPRESENTED BY JCTCIA MUST, FOR SOME MACHINES,
C           REPLACE THE 1 IN THE NAMED COMMON STATEMENT.
C
C       SUBRCUTINE AND FUNCTION SUBPROGRAMS REQUIRED
C           FETCH
C           FRMCEL
C           J7BODY
C           J71COM
C           J70SUP
C           J7VARI
C           NEWCEL
C           NEXT
C           POPUP
C           TOCELL
C           SYSENT
C           SYSERR
C
C       ERROR CODES FOR THIS PROGRAM
C           CODE FATAL  ERROR(DATA PROVIDED)
C           1    F      NO GROUP BODY - NOT PROTECTED (ADDRESS OF GROUP)
C           2    NF     NO GROUP BODY - PROTECTED (ADDRESS OF GROUP)
C           3    NF     JCGRUP NOT FOUND BY POST SET-UP TRACE (JCGRUP)
C
C       METHOD
C           SIMGEN INCLUDES IN THE SIM EACH ER, FUNCTION, CONSTRAINT
C           AND PROTECTED GROUP ENCOUNTERED IN THE GROUP BODY OF
C           JCGRUP.  ALL GROUP BODY INSERTS ARE IGNORED, BUT OTHERWISE
C           THERE IS A ONE TO ONE CORRESPONDENCE OF SIM ROWS TO GROUP
C           BODY ENTRIES.  DIMENSIONED ENTRIES ARE PLACED IN THE SIM
C           WITH INDICES VALUED AT THEIR MINIMUMS.  GROUP BODY ENTRIES
C           WITH PROTECTED GROUPS ARE INSPECTED OVER THE ENTIRE INDEX
C           RANGE DURING SIM CONSTRUCTION.  ALL ENTRIES ARE INSPECTED
C           OVER THE ENTIRE INDEX RANGE DURING THE POST SET-UP TRACE.
C
C                              **********
C
      SUBROUTINE SIMGEN(JCGRUP,JCSPTR,JCCLNO)
      COMMON/ALLOC/JCCORE(1)
      DIMENSION G(10),F(4),S(4),T(4),D(8),L(5),O(2),R(7)
      INTEGER G,F,S,T,D,O,R
      DATA G/3,3,2,2,7,1,4,4,1,1/,F/1,3,1,1/,S/1,8,1,1/,T/1,6,0,0/,
     1 D/0,0,1,0,3,0,0,0/,L/5,0,0,0,1/,O/1,0/,R/2,7,1,1,9,1,1/
```

```
C                   ENTER
      CALL SYSENT(1,3,2,6,JCCLNO)
C                   PARAMETERS
      M1=JCCORE(16)+1
      M1=JCCORE(M1)
      M2=JCCORE(17)
      R(4)=JCCORE(M2)
C                   MA ANC MB ARE ADDRESSES OF SPUR'S FOR TEMPORARY LISTS
      M2=M2+7
      MA=M1+5*JCCORE(M2)
      MB=MA+30
      T(3)=M1+1
      T(4)=T(3)
C                   MC=ADDRESS OF SPUR FOR GROUP
      M2=M2+1
      M3=JCCORE(M2)-1
      MC=M1+5*M3
      MD=M3-1
C                   ME=ADDRESS OF SPUR FOR GROUP BODY
      M2=M2+1
      M3=M2+5
      M3=JCCORE(M3)
      ME=M1+5*(JCCORE(M2)+M3-1)
C                   TEST JCPOFP FOR SIM
      CALL FRMCEL(JCCORE(MC),JCGRUP,JCSPTR,S,1)
      IF(JCSPTR.GT.0)GO TO 36
C                   SET UP SPUR BLOCK
      IF(M3.EC.0)M3=1
      M1=M3+5
      D(4)=M1*5
      CALL NEWCEL(D,M5,1)
C                   MF=ADDRESS OF SPUR BLOCK
      MF=M5+15
C                   MG = ADDRESS OF SPUR FOR ORDINATE
      MG=MF-5
C                   INITIALIZE SPUR BLOCK
      M4=M2+6
      M2=M5
      M5=M4+2
      DO 1 I=1,M1
      JCCORE(M2)=D(1)
      M2=M2+1
      DO 1 J=2,5
      JCCORE(M2)=(2/J)*(I/M1)*D(2)+(3/J)*(J/3)+(4/J)*(J/4)*(I+(1/I)*32+
     1 (2/I)*(I/2)*(JCCORE(M5)+1)+(3/I)*(I/3)*JCCORE(M4)-3)+(J/5)*(3+
     2 (2/I)*(I/2))
    1 M2=M2+1
C                   SET UP SIMBL
      M1=MF+10
      CALL NEWCEL(JCCORE(M1),JCSPTR,2)
      CALL TOCELL(JCCORE(MC),JCGRUP,JCSPTR,S,1)
      M1=JCSPTR
      DO 2 I=1,2
      JCCORE(M1)=0
    2 M1=M1+1
      JCCORE(M1)=MF
C                   MH=ADDRESS OF GROUP INDEX ADDRESS STORAGE BLOCK
      M1=JCCORE(16)+6
      MH=JCCORE(M1)-1
C                   PHASE I - ORDINATES
C                   MI=LEVEL
      MI=0
```

```
C                   MJ=DEPTH
      MJ=0
C                   MK=ADDRESS OF CURRENT LIST ENTRY
      MK=JCGRUP
C                   ML=OVER-RIDE INDICATOR
      ML=0
C                   MM=TEMPORARY LIST HEADCELL
      MM=0
C                   GROUP
    3 CALL FRMCEL(JCCORE(MC),MK,D,G,2)
      IF((D(1)+D(2)+ML).EQ.0)CALL SYSERR(-1,MK)
      IF((D(1)+ML).GT.D(2))CALL SYSERR(2,MK)
      IF(ML.GT.0)ML=ML+1
      IF(D(5).GT.0)MJ=D(5)
      IF((ML.GT.0).OR.((ML+D(1)).EQ.0))GO TO 4
C                   PROTECTED GROUP
C                   MN=ROW NUMBER
      CALL J7BODY(MC,MK,JCSPTR,MN,1,1)
      GO TO 6
C                   NOT PROTECTED GROUP OR AN INNER PROTECTED GROUP
    4 IF(D(5).EQ.0)GO TO 6
      M1=5
    5 M2=MK+MD
      CALL J7ICOM(MC,M2,MF,0,0,M1,MJ,D,M3,1)
      IF(M3.EQ.-1)GO TO 8
C                   NEXT LIST ENTRY
    6 M1=MK
      M2=MI
      D(1)=MJ
      D(2)=MK
      MK=NEXT(JCCORE(16),M1,D(6),1,MI,MM,0,4,1)
      IF(MI.GE.M2)GO TO 10
C                   COMPLETE CURRENT GROUP
      M1=MJ-D(1)
      MJ=D(1)
      M2=1
    7 M3=5-M2
      IF(D(M3).GT.0)CALL J7VARI(ME,D(M3),JCSPTR,MN,M2,0,0,1)
      M2=M2+1
      IF(M2.EQ.2)GO TO 7
      IF((ML*M1*MJ).EQ.0)GO TO 9
      M1=-5
      M4=MK
      MK=C(2)
      GO TO 5
    8 MK=M4
    9 IF(ML.GT.0)ML=ML-1
      IF(MI.EQ.0)GO TO 11
      IF(MK.EQ.0)GO TO 6
C                   TEST GROUP BODY LIST ENTRY FLAG
   10 CALL FRMCEL(JCCORE(ME),MK,M1,F,3)
      IF(M1.EQ.1)GO TO 3
      IF(M1.NE.2)GO TO 6
C                   FUNCTION, ER OR CONSTRAINT
      M1=2
      IF(ML.GT.0)M1=3
      CALL J7BODY(ME,MK,JCSPTR,MN,M1,2)
      GO TO 6
C                   PHASE II - ELEMENTS
C                   BEGIN WITH LAST ENTRY ON VERTICAL ORDINATE
C                   MD=ADDRESS OF LOAD VECTOR FOR ROW ORDINATE
   11 MD=J7OSUP(JCSPTR,1,0,0,1)
```

```
      KEY=D(6)
      M1=MO+7
      D(2)=MN-JCCORE(M1)
      G(6)=4
C                     ACCESS ROW ORDINATE ENTRY
   12 CALL FETCH(JCCORE(MG),0,M1,S,F,JCCORE(MO),0,1)
      D(2)=0
      IF(M1.GT.2)GO TO 13
C                     FUNCTION, ER OR CONSTRAINT
      CALL J78ODY(MC,0,JCSPTR,MN,5,3)
      GO TO 22
C                     GROUP ON SIM
   13 CALL FETCH(JCCORE(MG),I,0,R,F,JCCORE(MO),0,2)
      MK=C(2)
      MJ=0
C                     PREPARE FOR TRACE
   14 IF(MM.EQ.0)GO TO 15
      CALL POPUP(JCCORE(MB),MM,MM,M1,T,1,1)
      IF(M1.GT.0)CALL RETURN(JCCORE(MA),M1,0,1)
      GO TO 14
C                     DIMENSIONED GROUP
   15 IF(C(1).EQ.0)GO TO 16
      MK=JCCORE(MK)
C                     GROUP
   16 CALL FRMCEL(JCCORE(MC),MK,D(2),G(4),4)
      IF(C(2).EQ.0)GO TO 18
C                     INDICES
      MJ=D(2)
      M1=5
   17 M2=MK+MC
      CALL J7ICOM(MC,M2,MF,0,0,M1,MJ,D,M3,2)
      IF(M3.EQ.-1)GO TO 19
C                     NEXT LIST ENTRY
   18 M1=MK
      M2=MI
      D(2)=MK
      D(1)=MJ
      MK=NEXT(JCCORE(16),M1,D(3),1,MI,MM,0,2,2)
      IF(MI.GE.M2)GO TO 21
C                     DECREASE IN LEVEL
      M1=MJ-C(1)
      MJ=C(1)
      IF((MJ*M1).EQ.0)GO TO 20
      M1=-5
      M4=MK
      MK=C(2)
      GO TO 17
   19 MK=M4
   20 IF(MI.EQ.0)GO TO 22
      IF(MK.EQ.0)GO TO 18
C                     TEST FLAG
   21 CALL FRMCEL(JCCORE(ME),MK,M1,F,5)
      IF(M1.EQ.1)GO TO 16
      IF(M1.NE.2)GO TO 18
C                     FUNCTION, ER OR CONSTRAINT
      CALL J78ODY(ME,MK,JCSPTR,MN,4,4)
      GO TO 18
C                     PROCEED TO NEXT ROW
   22 MN=MN-1
      D(2)=-1
      IF(MN.GT.0)GO TO 12
C                     PHASE III - POST SET-UP INITIALIZATION
```

```
C                    ACCESS GENDER GROUP
      M1=JCCORE(16)
      MK=JCCORE(M1)
      M1=M1+2
      MJ=0
      G(6)=2
      D(8)=JCCORE(M1)+1
      IF(KEY.EQ.D(8))C(8)=D(8)+1
C                    DESTROY REMNANTS OF TEMPORARY LIST
      MB=MB-10
   23 IF(MM.EQ.0)GO TO 24
      CALL POPUP(JCCORE(MB),MM,MM,M1,T,1,2)
      IF(M1.GT.0)CALL RETURN(JCCORE(MA),M1,0,2)
      GO TO 23
C                    NEXT LIST ENTRY
   24 M1=MK
      M2=MI
      D(4)=MJ
      D(7)=MK
      MK=NEXT(JCCORE(16),M1,D(8),1,MI,MM,D(4),4,3)
      IF(MI.GE.M2)GO TO 29
C                    DECREASE IN LEVEL
   25 M1=MJ-D(4)
      MJ=D(4)
      M2=6
   26 IF(C(M2).GT.0)CALL J7VARI(ME,D(M3),JCSPTR,0,6,D,0,2)
      M2=M2-1
      IF(M2.EQ.5)GO TO 26
      IF(M1.EQ.0)GO TO 28
      M1=-5
      M4=MK
      MK=C(7)
      GO TO 31
   27 MK=M4
   28 IF(ML.GT.0)GO TO 33
      IF((MK+MM).EQ.0)GO TO 34
C                    TEST FLAG
   29 CALL FRMCEL(JCCORE(ME),MK,M1,F,6)
      IF(M1.EQ.1)GO TO 30
      IF(M1.NE.2)GO TO 24
C                    FUNCTION, ER OR CONSTRAINT
      CALL J7BODY(ME,MK,JCSPTR,0,6,5)
      GO TO 24
C                    GROUP
   30 IF(MK.EQ.JCGRUP)GO TO 32
      CALL FRMCEL(JCCORE(MC),MK,D(5),G(4),7)
      IF(C(7).EQ.0)GO TO 24
C                    INDICES
      MJ=D(7)
      M1=5
   31 M2=MK+MD
      CALL J7ICOM(MC,M2,MF,0,0,M1,MJ,0,M3,3)
      IF(M3.EQ.-1)GO TO 27
      IF(ML.GT.0)GO TO 25
C                    JCPOFP FOUND
   32 MB=MB+10
      T(4)=T(4)+6
      ML=1
C                    REMOVE HEADCELL
   33 IF(MM.EQ.0)GO TO 35
      CALL PCPUP(JCCORE(MB),MM,MM,D,T,1,3)
      M1=0
```

```
      IF(D(1).GT.0)CALL RETURN(JCCORE(MA),D(1),M1,3)
      GO TO 25
C                     ERROR
   34 CALL SYSERR(3,JCGRUP)
C                     SET KEY
   35 M1=JCCORE(16)+2
      JCCORE(M1)=D(3)
C                     EXIT
   36 CALL SYSEXT
      RETURN
      END
```

```
C
C                                  **********
C                                  * SIMINT *
C                                  **********
C
C        PURPOSE
C            SIMINT INTERCHANGES TWO PARALLEL ORDINATE VECTORS.
C
C        USAGE
C            CALL SIMINT(JCSPTR,JCONOO,JCONOT,JCRORC,JCCLNO)
C
C        DATA FORMAT
C            N/A
C
C        DESCRIPTION OF PARAMETERS
C            JCSPTR = THE POINTER TO THE SIMBL
C            JCONOO = THE NUMBER OF ONE ON THE ORDINATES
C            JCONOT = THE NUMBER OF THE OTHER ORDINATE
C            JCRORC = 0 FOR ROW OR 1 FOR COLUMN
C            JCCLNO = A USER ASSIGNED CALL STATEMENT ID. PARAMETER
C
C        REMARKS
C            THE NUMBER REPRESENTED BY JCTCIA MUST, FOR SOME MACHINES,
C            REPLACE THE 1 IN THE NAMED COMMON STATEMENT.
C
C        SUBROUTINE AND FUNCTION SUBPROGRAMS REQUIRED
C            J70SUP
C            SYSENT
C            SYSEXT
C
C        ERROR CODES FOR THIS PROGRAM
C            NONE
C
C        METHOD
C            SELF-EXPLANATORY
C
C                                  **********
C
      SUBROUTINE SIMINT(JCSPTR,JCONOO,JCONOT,JCRORC,JCCLNO)
      COMMON/ALLOC/JCCORE(1)
C                  ENTER
      CALL SYSENT(1,3,2,7,JCCLNO)
C                  OBTAIN ADDRESSES
      MZ=1
      MY=1
      DO 1 I=1,2
      M1=JCCNCO+I-2
      MA=J70SUP(JCSPTR,M1,JCRORC,MZ,I)
      M1=JCONOT+I-2
      M2=I+2
      MB=J70SUP(JCSPTR,M1,JCRORC,MY,M2)
C                  PERFORM INTERCHANGE
      JCCORE(MZ)=MB
    1 JCCORE(MY)=MA
C                  EXIT
      CALL SYSEXT
      RETURN
      END
```

```
C
C*********************************************************************
C                              **********
C                              * SIMOPT *
C                              **********
C
C       PURPOSE
C           SIMOPT RECORDS OR ERASES AN OUTPUT ASSIGNMENT IN BOTH THE
C           HORIZONTAL AND VERTICAL ORDINATES.
C
C       USAGE
C           CALL SIMOPT(JCSPTR,JCRNUM,JCCNUM,JCSENS,JCCLNO)
C
C       DATA FORMAT
C           N/A
C
C       DESCRIPTION OF PARAMETERS
C           JCSPTR = THE POINTER TO THE SIMBL
C           JCRNUM = THE ROW NUMBER
C           JCCNUM = THE COLUMN NUMBER
C           JCSENS = +N TO ASSIGN OUTPUT OR -1 TO ERASE OUTPUT.  N = 1
C                    OR 2 AS COUNT INITIALIZATION IS OR IS NOT DESIRED.
C           JCCLNO = A USER ASSIGNED CALL STATEMENT ID. PARAMETER
C
C       REMARKS
C           THE NUMBER REPRESENTED BY JCTCIA MUST, FOR SOME MACHINES,
C           REPLACE THE 1 IN THE NAMED COMMON STATEMENT.
C
C       SUBROUTINE AND FUNCTION SUBPROGRAMS REQUIRED
C           J7OTPT
C           SYSENT
C           SYSEXT
C
C       ERROR CODES FOR THIS PROGRAM
C           NONE
C
C       METHOD
C           SELF-EXPLANATORY
C
C                              **********
C
      SUBROUTINE SIMOPT(JCSPTR,JCRNUM,JCCNUM,JCSENS,JCCLNO)
C                 ENTER
      CALL SYSENT(1,3,2,8,JCCLNO)
C                 SERVICE ROW
      IF(JCRNUM.GT.0)CALL J7OTPT(JCSPTR,JCRNUM,0,JCSENS,JCCNUM,1)
C                 SERVICE COLUMN
      IF(JCCNUM.GT.0)CALL J7OTPT(JCSPTR,JCCNUM,1,JCSENS,JCRNUM,2)
C                 EXIT
      CALL SYSEXT
      RETURN
      END
```

```
C
C                                  **********
C                                  * SIMRTV *
C                                  **********
C
C
C       PURPOSE
C           SIMRTV SETS THE ELEMENT STATUS FLAGS OF TEAR VARIABLES TO
C           THREE.
C
C       USAGE
C           CALL SIMRTV(JCGRUP,JCCLNO)
C
C       DATA FORMAT
C           N/A
C
C       DESCRIPTION OF PARAMETERS
C           JCGRUP = ABSOLUTE ADDRESS OF GENDER GROUP
C           JCCLNO = A USER ASSIGNED CALL STATEMENT ID. PARAMETER
C
C       REMARKS
C           THE NUMBER REPRESENTED BY JCTCIA MUST, FOR SOME MACHINES,
C           REPLACE THE 1 IN THE NAMED COMMON STATEMENT.
C
C       SUBROUTINE AND FUNCTION SUBPROGRAMS REQUIRED
C           FETCH
C           FIND
C           FRMCEL
C           J7LOCV
C           STORE
C           SYSENT
C           SYSERR
C           SYSEXT
C           TGCELL
C
C       ERROR CODES FOR THIS PROGRAM
C           CODE FATAL  ERROR(DATA PROVIDED)
C           1    NF     NO SIM (JCGRUP)
C           2    NF     NO TEAR VARIABLES (JCGRUP)
C           3    NF     TEAR VARIABLE NOT FOUND (VARIABLE CODE NAME)
C
C       METHOD
C           SIMRTV TRACES THE TEAR VARIABLE LIST OF JCGRUP.  EXCEPT FOR
C           THE ELEMENT REPRESENTING A TEAR VARIABLE AS AN OUTPUT, ALL
C           TEAR VARIABLE ELEMENTS ARE FITTED WITH STATUS FLAGS OF 3.
C
C                                  **********
C
      SUBROUTINE SIMRTV(JCGRUP,JCCLNO)
      COMMON/ALLOC/JCCORE(1)
      DIMENSION O(3),V(3),G(17),P(4),N(7),E(10),L(3),O(4)
      INTEGER C,V,G,P,E,O
      DATA G/3,1,1,1,2,1,1,21,1,1,2,7,3,3,8,1,1/,P/1,6,1,1/,N/2,7,1,1,9,
     1 1,1/,E/3,2,1,1,5,1,1,3,1,1/,L/1,1,0/,O/1,13,1,1/
C                 ENTER
      CALL SYSENT(1,3,2,9,JCCLNO)
C                 SETTING UP REQUIRED PARAMETERS
      MA=JCCORE(16)+1
      M1=JCCORE(17)+8
      MA=JCCORE(MA)+5*(JCCORE(M1)-1)
C                 ACCESS GROUP
      CALL FRMCEL(JCCORE(MA),JCGRUP,V,G(11),1)
      M1=1
```

```
      IF(V(2).EQ.0)GO TO 9
      M1=2
      IF(V(1).EQ.0)GO TO 9
      G(11)=-1
      G(14)=1
      G(15)=1
      G(16)=3
C                     ACCESS SIMBL
      M1=V(2)+2
      MB=JCCORE(M1)-5
      MC=MB-5
      MD=V(2)
      ME=J7OSUP(MC,1,1,0,1)
      M2=2
      MH=ME+7
C                     SECURE WORKING AREA FOR INDEX CALCULATIONS
      M5=JCCORE(17)+14
      ML=0
      IF(JCCORE(M5).EQ.0)GO TO 1
      MK=MB+5*JCCORE(M5)+10
      CALL NEWCEL(JCCORE(MK),ML,1)
      MM=ML+1
      MN=ML+2
C                     RETRIEVE DATA FROM TEAR VARIABLE LIST ENTRY
    1 M4=V(1)
      CALL FRMCEL(JCCORE(MA),M4,V,G,2)
      IF(V(2).NE.0)GO TO 8
C                     SCALAR VARIABLE
      M5=3
      CALL FIND(JCCORE(MB),L(M2),V(2),N,N,JCCORE(ME),M5,1)
    2 M2=1
      IF(M5.NE.0)GO TO 4
    3 CALL SYSERR(3,V(3))
      GO TO 7
C                     ACCESS OUTPUT
    4 V(2)=0
      L(3)=0
      CALL FETCH(JCCORE(MB),L(2),V(2),0,L,JCCORE(ME),0,1)
      M3=4
C                     TRACE ELEMENTS IN COLUMN
      CALL FETCH(JCCORE(MB),L(2),D,P,L,JCCORE(ME),0,2)
    6 IF(D(1).EQ.0)GO TO 7
C                     EXTRACT DATA FROM ELEMENT
      M5=D(1)
      CALL FRMCEL(JCCORE(MC),M5,D,E,3)
      MZ=3
      IF(D(2).EQ.1)MZ=4
      IF(D(3).NE.V(2))CALL TOCELL(JCCORE(MC),M5,MZ,E(4),1)
      GO TO 6
C                     ADVANCE TO NEXT TEAR VARIABLE LIST ENTRY
    7 IF(V(1).GT.0)GO TO 1
C                     RETURN WORKING SPACE
      IF(ML.GT.0)CALL RETURN(JCCORE(MK),ML,ML,1)
      GO TO 10
C                     DIMENSIONED VARIABLE
    8 JCCORE(ML)=V(3)
      JCCORE(MM)=V(2)
      G(17)=G(16)+V(2)-1
      CALL FRMCEL(JCCORE(MK),M4,JCCORE(MN),G(14),4)
C                     LOCATE DIMENSIONED VARIABLE
      CALL J7LODV(MD,L(M2),MK,M5,MF,1)
      GO TO 2
```

```
C                    NO SIM OR OUTPUT LIST
    9 CALL SYSERR(M1,JCGRUP)
C                    EXIT
   10 CALL SYSEXT
      RETURN
      END
```

```
C
C*********************************************************************
C                              **********
C                              * J78ODY *
C                              **********
C
C        PURPOSE
C            J78ODY HANDLES ER'S, FUNCTIONS AND CONSTRAINTS WITH RESPECT
C            TO SIM GENERATION.
C
C        USAGE
C            CALL J78ODY(JCSPRG,JCPOFP,JCSPTR,JCROWN,JCMODE,JCCLNO)
C
C        DATA FORMAT
C            N/A
C
C        DESCRIPTION OF PARAMETERS
C            JCSPRG = THE ADDRESS OF THE SPUR FOR GENDER
C            JCPOFP = THE ADDRESS OF THE ENTRY (ER, FUNCTION, ETC.)
C            JCSPTR = THE ADDRESS OF THE SIMBL
C            JCROWN = THE SIM ROW NUMBER
C            JCMODE = A POSITIVE INTEGER HAVING THE FOLLOWING STATESO
C                     1 = GROUP, PROTECTED, ADD TO ORDINATE
C                     2 = GROUP BODY ENTRY, NOT PROTECTED GROUP, ADD TO
C                         ORDINATE
C                     3 = GROUP BODY ENTRY, PROTECTED GROUP, INSPECT
C                         OUTPUT LIST
C                     4 = GROUP BODY ENTRY, PROTECTED GROUP, INSPECT
C                         VARIABLE INCIDENCE
C                     5 = SIM ENTRY, INSPECT VARIABLE INCIDENCE
C                     6 = GROUP BODY, POST SET-UP INITIALIZATION
C
C        REMARKS
C            THE NUMBER REPRESENTED BY JCTCIA MUST, FOR SOME MACHINES,
C            REPLACE THE 1 IN THE NAMED COMMON STATEMENT.
C
C        SUBROUTINE AND FUNCTION SUBPROGRAMS REQUIRED
C            FETCH
C            FRMCEL
C            J7ICOM
C            J7OSUP
C            J7VARI
C            STORE
C            SYSENT
C            SYSEXT
C
C        ERROR CODES FOR THIS PROGRAM
C            NONE
C
C        METHOD
C            SELF-EXPLANATORY
C
C                              **********
C
      SUBROUTINE J78ODY(JCSPRG,JCPOFP,JCSPTR,JCROWN,JCMODE,JCCLNO)
      COMMON/ALLOC/JCCORE(1)
      DIMENSION L(4),G(20),O(6),F(2),R(19),S(5),P(4)
      INTEGER G,O,F,R,S,P
      DATA L/3,1,1,0/,G/1,7,4,4,5,19,1,1,17,1,1,3,2,2,20,1,1,18,1,1/,
     1 F/1,0/,R/6,7,1,1,9,1,1,8,1,1,11,1,1,5,1,1,12,1,1/,S/5,0,0,
     2 0,1/,P/1,1,1,1/
C                    ENTER
```

```
      CALL SYSENT(1,3,2,10,JCCLNO)
      D(1)=0
      D(2)=JCPOFP
      D(3)=3
      D(4)=0
      D(5)=0
      D(6)=1
C                     PARAMETERS FOR SECEDE
      M1=JCCORE(15)+1
      MAA=JCCORE(M1)
      M1=M1+1
      MAB=JCCORE(M1)
      M1=MAB+28
      JCCORE(M1)=-1
C                     MA=ADDRESS OF SPUR BLOCK
      M1=JCSPTR+2
      MA=JCCORE(M1)
C                     MB=ADDRESS OF SPUR FOR ORDINATE
      MB=MA-5
      MC=0
      IF((JCMODE.GT.2).AND.(JCMODE.NE.5))GO TO 1
C                     MD=ADDRESS OF LOAD VECTOR FOR ORDINATE
      MC=1
      MD=J70SUP(JCSPTR,1,0,0,1)
      M2=MD+7
      IF(JCMODE.EQ.5)GO TO 31
      ME=3
      M1=MD+14
      IF(JCCORE(M1).EQ.-1)ME=1
      M1=MD+8
      L(4)=JCCORE(M1)-JCCORE(M2)+1
      M1=JCCORE(17)+18
      F(2)=JCCORE(M1)
C                     MODE FOR J7VARI CALL
    1 MH=JCMODE+1
C                     DATA
      M1=5-4*(1/JCMODE)
      CALL FRMCEL(JCCORE(JCSPRG),JCPOFP,D,G(M1),1)
C                     DIMENSION CHECK
      IF(D(1).GT.0)GO TO 2
      GO TO (5,5,8,6,61,8),JCMODE
C                     INDICES
    2 M1=JCCORE(17)+9-(1/JCMODE)
      MI=JCPOFP+JCCORE(M1)
      M2=4+1/JCMODE
    3 CALL J7ICOM(JCSPRG,MI,MA,MC,2,M2,D(1),L,M1,1)
      IF(M1.NE.1)GO TO 9
      GO TO (4,4,8,6,61,8),JCMODE
C                     ACCESS ROW ORDINATE
   31 MH=5
      L(4)=JCROWN-JCCORE(M2)
      CALL FETCH(JCCORE(MB),L(3),D,R,F,JCCORE(MD),0,1)
      MZ=0
      IF(D(1).EQ.0)GO TO 61
      MZ=1
      M1=D(2)
      D(2)=JCCORE(M1)
      M2=JCCORE(16)+8
      M3=JCCORE(M2)
      JCCORE(M2)=M1+2
      GO TO 61
C                     ADD TO ORDINATE
```

```
    4 IF(JCMODE.EQ.1)D(2)=JCPOFP
      JCCORE(MC)=D(2)
      M1=MC+1
      JCCORE(M1)=D(1)
      D(1)=1
      D(2)=MC
    5 D(3)=(D(3)+2)/3+2*(2-JCMODE)
      IF(D(4).EQ.1)D(6)=0
      M1=D(5)
      D(5)=1/JCMODE
      CALL STORE(JCCORE(MB),L(ME),D,R,F,JCCORE(MD),1)
      JCROWN=JCCORE(M2)
      IF(JCMODE.EQ.1)GO TO 9
      IF(D(1).EQ.1)D(2)=JCCORE(MC)
      D(5)=M1
      GO TO 8
C                     ACCESS FUNCTION IN SECEDE
    6 IF(M2.LT.0)GO TO 7
   61 M1=JCCORE(17)+6
      M1=JCCORE(M1)
      S(2)=D(2)/M1
      S(4)=D(2)-S(2)*M1
      MS=S(2)
      S(2)=S(2)+4
      S(3)=D(3)+6
    7 CALL FETCH(JCCORE(MAA),S,D(5),P,F,JCCORE(MAB),0,2)
C                     J7VARI CALL
      IF(D(5).EQ.0)GO TO 9
      CALL J7VARI(MAA,D(5),JCSPTR,JCROWN,MH,MAB,MS,1)
      IF((JCMODE*MZ).EQ.5)JCCORE(M2)=M3
      MZ=0
      GO TO 9
    8 IF(D(5).EQ.0)GO TO 9
      CALL J7VARI(JCSPRG,D(5),JCSPTR,JCROWN,MH,MAB,MS,2)
      IF(JCMODE.LE.2)GO TO 9
C                     ITERATE
      M2=-4
      IF(D(1).GT.0)GO TO 3
C                     EXIT
    9 CALL SYSEXT
      RETURN
      END
```

```
C
C*****************************************************************
C                              **********
C                              * J7COMV *
C                              **********
C
C       PURPOSE
C           J7COMV CHECKS A VARIABLE FOR POSSESSING A COMMON VARIABLE
C           DECLARATION.
C
C       USAGE
C           CALL J7COMV(JCSPUR,JCVNAM,JCLOAD,JCCLNO)
C
C       DATA FORMAT
C           N/A
C
C       DESCRIPTION OF PARAMETERS
C           JCSPUR = THE ADDRESS OF THE SPUR FOR SECEDE
C           JCVNAM = THE VARIABLE NAME.  IF THE VARIABLE IS FOUND TO BE
C                    COMMON, UPON RETURN JCVNAM WILL CONTAIN THE NAME
C                    OF THE COMMON VARIABLE.
C           JCLOAD = THE ADDRESS OF THE LOAD FOR SECEDE
C           JCCLNO = A USER ASSIGNED CALL STATEMENT ID. PARAMETER
C
C       REMARKS
C           THE NUMBER REPRESENTED BY JCTCIA MUST, FOR SOME MACHINES,
C           REPLACE THE 1 IN THE NAMED COMMON STATEMENT.
C
C       SUBROUTINE AND FUNCTION SUBPROGRAMS REQUIRED
C           FETCH
C           SYSENT
C           SYSEXT
C
C       ERROR CODES FOR THIS PROGRAM
C           NONE
C
C       METHOD
C           SELF-EXPLANATORY
C
C                              **********
C
      SUBROUTINE J7COMV(JCSPUR,JCVNAM,JCLOAD,JCCLNO)
      COMMON/ALLOC/JCCORE(1)
      DIMENSION P(4),V(4),L(5)
      INTEGER P,V,D
      DATA P/1,1,1,1/,V/1,1,2,2/,L/5,0,1,1,0/
C                   ENTER
      CALL SYSENT(1,3,2,11,JCCLNO)
C                   SETTING UP LISTER VECTOR
      M1=JCLOAD+28
      JCCORE(M1)=-1
      M1=JCCORE(17)+6
      M1=JCCORE(M1)
      L(2)=JCVNAM/M1
      IF(L(2).EQ.0)GO TO 104
      L(5)=JCVNAM-M1*L(2)
      L(2)=L(2)+9
C                   RETRIEVE NAME OF COMMON VARIABLE
      CALL FETCH(JCCORE(JCSPUR),L,M2,P,L,JCCORE(JCLOAD),0,1)
      L(5)=M2
      CALL FETCH(JCCORE(JCSPUR),L(4),M2,V,L,JCCORE(JCLOAD),0,2)
      IF(M2.GT.0)JCVNAM=M2
```

```
C                       EXIT
  104 CALL SYSEXT
      RETURN
      END
```

```
C
C*********************************************************************
C                              **********
C                              * J7ICOM *
C                              **********
C
C       PURPOSE
C           J7ICOM PERFORMS INDEX COMPUTATIONS.
C
C       USAGE
C           CALL J7ICOM(JCSPUR,JCPOFP,JCSPBL,JCPOSP,JCSKIP,JCMODE,
C           JCDIMS,JCLOAD,JCSORF,JCCLNO)
C
C       DATA FORMAT
C           N/A
C
C       DESCRIPTION OF PARAMETERS
C           JCSPUR = THE ADDRESS OF THE SPUR FOR SECEDE IF JCPOFP = N,
C                    OR FOR GENDER LIST OTHERWISE
C           JCPOFP = THE NUMBER OF WORDS TO PROGRESS IF DIMENSIONED
C                    ENTITY IS IN SECEDE, OR JCPOFP IS THE POINTER TO A
C                    GENDER LIST COMPONENT INDICES
C           JCSPBL = THE ADDRESS OF THE SPUR BLOCK
C           JCPOSP = THE ADDRESS OF THE STORAGE LOCATION FOR THE
C                    INDICES.  IF JCPOSP IS PASSED AS 1, STORAGE WILL
C                    BE AUTOMATICALLY PROVIDED.  IF JCPOSP IS PASSED AS
C                    0, THE OPTION OF JCPOSP POINTING TO INDEX STORAGE
C                    IS WAIVED.
C           JCSKIP = THE NUMBER OF WORDS OF JCPOSP TO SKIP BEFORE
C                    COMMENCING INDEX STORAGE
C           JCMODE = AN INTEGER EQUAL IN MAGNITUDE OF 1 THROUGH 5 AS
C                    RESPECTIVELY THE INDICES TO BE COMPUTED REFER TO
C                    A VARIABLE (REMOTE), A VARIABLE (LIST), A FUNCTION
C                    (REMOTE), A FUNCTION (LIST) OR A GROUP.  IF JCMODE
C                    IS NEGATIVE, INCREMENTING THE CURRENT INDEX SET BY
C                    1 IS INDICATED.  IF POSITIVE, JCMODE INDICATES
C                    THE COMPUTATION OF THE MINIMUM, CURRENT (EQUAL TO
C                    THE MINIMUM) AND MAXIMUM OF EACH INDEX IS DESIRED.
C           JCDIMS = THE DIMENSIONALITY OF A VARIABLE OR FUNCTION, OR
C                    THE DEPTH (SEE REMARKS) OF A GROUP
C           JCLOAD = THE ADDRESS OF THE LOAD VECTOR FOR SECEDE
C                    REFERENCING THE INDICES
C           JCSORF = +1 OR -1 AS RESPECTIVELY THE FIRST INDEX IS WITHIN
C                    ITS MAXIMUM LIMIT
C           JCCLNO = A USER ASSIGNED CALL STATEMENT ID. PARAMETER
C
C       REMARKS
C           THE NUMBER REPRESENTED BY JCTCIA MUST, FOR SOME MACHINES,
C           REPLACE THE 1 IN THE NAMED COMMON STATEMENT.
C           THE DEPTH OF A GROUP DIFFERS FROM ITS LEVEL IN THAT PARENT
C           GROUPS NOT INDEXED DO NOT ENTER INTO DETERMINING A GROUP'S
C           DEPTH.
C           USE THE FUNCTION MODE FOR VARIABLES POSSESSING MINIMUM AND
C           MAXIMUM LIMITS (AS SELF-MAPPING IS ALLOWED).
C
C       SUBROUTINE AND FUNCTION SUBPROGRAMS REQUIRED
C           FETCH
C           FRMCEL
C           NEWCEL
C           RETURN
C           SYSENT
C           SYSEXT
```

```
C
C       ERROR CODES FOR THIS PROGRAM
C           NONE
C
C       METHOD
C           J7ICOM EMPLOYS THE INDEX STORAGE VECTORS PROVIDED BY CUDGEL
C           FOR THE GENDER LIST.  JCLOAD MUST REFER TO THE BEGINNING OF
C           THE FILE-STORED INDICES.  J7ICOM EMPLOYS A LISTER OF (1,N)
C           INITIALLY AND (1,1) THEREAFTER.
C
C                                  **********
C
      SUBROUTINE J7ICOM(JCSPUR,JCPOFP,JCSPBL,JCPOSP,JCSKIP,JCMODE,
     1 JCDIMS,JCLOAD,JCSORF,JCCLNO)
      COMMON/ALLOC/JCCORE(1)
      DIMENSION D(7),G(22),L(2),P(3,4)
      INTEGER D,G,P
      DATA G/7,9,1,1,10,1,1,11,1,1,12,1,1,13,1,1,14,1,1,15,1,1/,L/1,0/
C                   ENTER
      CALL SYSENT(1,3,2,12,JCCLNO)
C                   PARAMETERS
      M1=JCCORE(16)+2
      DO 1 I=1,2
      M2=5-I
      DO 1 J=1,M2
      M1=M1+1
    1 P(I,J)=JCCORE(M1)
      M1=P(1,4)+JCDIMS-1
      IF(JCMODE.EQ.5)JCCORE(M1)=JCPOFP
      MA=1
      MB=0
      L(2)=JCPOFP
C                   MODE
      MC=JCDIMS
      MD=IABS(JCMODE)
      IF(MD.EQ.5)MC=1
      MC=JCSPBL+5*(MC+JCSKIP-1)
      ME=JCPOSP
      IF((JCPOSP.EQ.1).OR.((MD.LE.2).AND.(JCPOSP.LE.0)))CALL NEWCEL(
     1 JCCORE(MC),ME,1)
      IF(JCDIMS.EQ.0)GO TO 11
      P(3,2)=ME+JCSKIP
      MF=3
      IF(MD.LE.2)GO TO 2
      MF=2
      IF(MD.EQ.5)MF=1
      IF(JCMODE.GT.0)GO TO 2
C                   INCREMENT
      JCSORF=-1
      DO 101 I=1,JCDIMS
      IF(JCSORF.EQ.1)GO TO 101
      M1=JCDIMS-I
      M2=P(MF,2)+M1
      M1=P(MF,3)+M1
      JCSORF=1
      JCCORE(M2)=JCCORE(M2)+1
      IF(JCCORE(M2).LE.JCCORE(M1))GO TO 101
      JCSORF=-1
      MA=I
      MB=1
  101 CONTINUE
C                   MODE
```

```
      IF(MB.EQ.0)GO TO 11
      IF(JCSORF.GT.0)GO TO 2
C                     RETURN ME
      IF((JCPOSP.EQ.1).OR.((MD.LE.2).AND.(JCPOSP.LE.0)))CALL RETURN(
     1 JCCORE(MC),ME,ME,1)
      GO TO 13
C                     MIN, CURRENT AND MAX
    2 DO 10 I=MA, JCDIMS
      M1=2
    3 GO TO (4,5,4,5,7),MD
C                     REMOTE
    4 CALL FETCH(JCCORE(JCSPUR),L,D,G,L,JCCORE(JCLOAD),0,1)
      L(2)=1
      GO TO 8
C                     LIST
    5 M2=JCPOFP+2*I-1+M1/3
    6 CALL FRMCEL(JCCORE(JCSPUR),M2,D,G,1)
      GO TO 8
C                     GROUP
    7 M2=P(1,4)+I-1
      M2=JCCORE(M2)+M1/3
      GO TO 6
C                     CURRENT OR MAX
    8 IF(D(1).EQ.0)GO TO 9
      M2=D(1)+(D(1)/3)*(MF-D(1))
      D(7)=((-1)**D(4))*(P(M2,2)+D(2)-1)*(D(5)**((-1)**D(3)))+((-1)**
     1 D(6))*D(7)
    9 M2=P(MF,M1)+I-1
      JCCORE(M2)=D(7)
      IF((M1.EQ.0).OR.(MF.EQ.3))GO TO 10
C                     MIN
      M2=P(MF,1)+I-1
      JCCORE(M2)=D(7)
      M1=3
      IF(MD.GT.2)GO TO 3
   10 CONTINUE
      JCSORF=1
C                     COPY
   11 IF(ME.GT.1)JCPOSP=ME
      IF((JCPOSP.LE.1).OR.(MD.EQ.3).OR.(JCDIMS.EQ.3))GO TO 13
      M1=JCPOSP+JCSKIP-1
      M2=P(MF,2)-1
      MA=1
      IF(MF.EQ.1)MA=JCDIMS
      DO 12 I=MA,JCDIMS
      M1=M1+1
      M2=M2+1
   12 JCCORE(M1)=JCCORE(M2)
C                     EXIT
   13 CALL SYSEXT
      RETURN
      END
```

```
C
C*********************************************************************
C                              **********
C                              * J7LODV *
C                              **********
C
C       PURPOSE
C           J7LODV SEARCHES AN ORDINATE FOR A DIMENSIONED VARIABLE.
C
C       USAGE
C           CALL J7LODV(JCSPTR,LISTER,MASTER,INDIC,JCORDN,JCCLNO)
C
C       DATA FORMAT
C           N/A
C
C       DESCRIPTION OF PARAMETERS
C           JCSPTR = THE ADDRESS OF THE SIMBL
C           LISTER = SEE C4AORS DOCUMENTATION.  RESTRICTED TO LISTER(1)
C                    EQUAL TO 1.
C           MASTER = ADDRESS OF THE BLOCK OF WORDS CONTAINING VARIABLE
C                    NAME, DIMENSION AND INDICES.
C           INDIC  = 1 FOR SUCCESS AND 0 FOR FAILURE OF THE SEARCH
C           JCORDN = THE ADDRESS OF THE ORDINATE LOAD VECTOR
C           JCCLNO = A USER ASSIGNED CALL STATEMENT ID. PARAMETER
C
C       REMARKS
C           THE NUMBER REPRESENTED BY JCTCIA MUST, FOR SOME MACHINES,
C           REPLACE THE 1 IN THE NAMED COMMON STATEMENT.
C
C       SUBROUTINE AND FUNCTION SUBPROGRAMS REQUIRED
C           FETCH
C           FIND
C           SYSENT
C           SYSEXT
C
C       ERROR CODES FOR THIS PROGRAM
C           NONE
C
C       METHOD
C           SELF-EXPLANATORY
C
C                              **********
C
      SUBROUTINE J7LODV(JCSPTR,LISTER,MASTER,INDIC,JCORDN,JCCLNO)
      COMMON/ALLOC/JCCORE(1)
      DIMENSION LISTER(2),L(7)
      DATA L/1,7,1,1,9,1,1/
C                 ENTER
      CALL SYSENT(1,3,2,13,JCCLNO)
      MA=JCSPTR+2
      MA=JCCORE(MA)-5
      INDIC=0
      MB=0
      M1=JCORDN+7
      M2=JCCORE(M1)
C                 FIND A DIMENSIONED VARIABLE
    1 MB=MB+1
    2 CALL FIND(JCCORE(MA),LISTER,L,L,L,JCCORE(JCORDN),MB,1)
      IF(MB.NE.0)GO TO 3
      LISTER(2)=M2-JCCORE(M1)-1
      IF(MB.EQ.1)GO TO 1
      GO TO 5
```

```
C                    COMPARE INDICES
    3 LISTER(2)=0
      CALL FETCH(JCCORE(MA),LISTER,M3,L(4),L,JCCORE(JCORON),0,1)
      LISTER(2)=3-2*MB
      IF(M3.LE.0)GO TO 2
      M4=MASTER+1
      M4=JCCORE(M4)
      M5=0
      M6=MASTER
      DO 4 I=1,M4
      IF(M5.GT.0)GO TO 4
      IF(JCCORE(M3).NE.JCCORE(M6))M5=1
      M3=M3+1
      M6=M6+1
    4 CONTINUE
      IF(M5.GT.0)GO TO 2
C                    EXIT
      INDIC=1
    5 CALL SYSEXT
      RETURN
      END
```

```
C
C*****************************************************************
C                              **********
C                              * J70SUP *
C                              **********
C
C       PURPOSE
C           J70SUP ESTABLISHES THE LOAD VECTOR FOR AN ORDINATE.
C
C       USAGE
C           J70SUP(JCSPTR,JCONUM,JCRORC,JCAOCE,JCCLNO)
C
C       DATA FORMAT
C           N/A
C
C       DESCRIPTION OF PARAMETERS
C           JCSPTR = THE PUNTER TO THE SIMBL
C           JCONUM = THE ORDINATE NUMBER
C           JCRORC = 0 FOR ROW OR 1 FOR COLUMN
C           JCAOCE = THE ADDRESS OF THE CATALOGUE ENTRY OR 0 IF THIS
C                    VALUE IS NOT DESIRED
C           JCCLNO = A USER ASSIGNED CALL STATEMENT IO. PARAMETER
C
C       REMARKS
C           THE NUMBER REPRESENTED BY JCTCIA MUST, FOR SOME MACHINES,
C           REPLACE THE 1 IN THE NAMED COMMON STATEMENT.
C
C       SUBROUTINE AND FUNCTION SUBPROGRAMS REQUIRED
C           C4AORS
C           NEWCEL
C           RETURN
C           SYSENT
C           SYSEXT
C
C       ERROR CODES FOR THIS PROGRAM
C           NONE
C
C       METHOD
C           THE ADDRESS OF THE LOAD VECTOR IS STORED IN THE CATALOGUE
C           WORD IMMEDIATELY FOLLOWING THE WORD APPROPRIATE TO ORDINATE
C           NUMBER JCONUM.  J70SUP INITIALIZES THE 15TH AND 30TH WORDS
C           OF THE LOAD VECTOR.
C
C                              **********
C
      FUNCTION J70SUP(JCSPTR,JCONUM,JCRORC,JCAOCE,JCCLNO)
      COMMON/ALLOC/JCCORE(1)
      DIMENSION L(2),K(4)
C                 ENTER
      CALL SYSENT(1,3,2,14,JCCLNO)
C                 ACQUIRE SPACE FOR LOAD VECTOR
      MA=JCSPTR+2
      MA=JCCORE(MA)-15
      CALL NEWCEL(JCCORE(MA),MB,1)
C                 INITIALIZE LOAD VECTOR FOR C4AORS CALL
      M1=MB+29
      JCCORE(M1)=JCSPTR-JCRORC+1
      M1=MB+13
      JCCORE(M1)=1
      M1=MB+14
      JCCORE(M1)=1
C                 ACCESS LOAD VECTOR POINTER
```

```
      L(1)=2
      L(2)=JCONUM+1
      MC=MA+10
      CALL C4ADRS(JCCORE(MC),L,K,K,K,K,K,JCCORE(MB),0,1)
      M2=JCCORE(MB)
      IF(JCAOCE.NE.0)JCAOCE=M2
      IF(JCCORE(M2).GT.0)GO TO 1
C                   INSTALL JCCORE(MB) AS LOAD VECTOR
      JCCORE(M2)=MB
      J7OSUP=MB
      JCCORE(M1)=-1
      GO TO 2
C                 REPORT JCCORE(M2) AS LOAD VECTOR
    1 J7OSUP=JCCORE(M2)
      CALL RETURN(JCCORE(MA),MB,MB,1)
C                   EXIT
    2 CALL SYSEXT
      RETURN
      END
```

```
C
C*******************************************************************
C                                **********
C                                * J7OTPT *
C                                **********
C
C       PURPOSE
C           J7OTPT RECORDS OR ERASES AN OUTPUT ASSIGNMENT.
C
C       USAGE
C           CALL J7OTPT(JCSPTR,JCENUM,JCRORC,JCSENS,JCOTPT,JCCLNO)
C
C       DATA FORMAT
C           N/A
C
C       DESCRIPTION OF PARAMETERS
C           JCSPTR = THE POINTER TO THE SIMBL
C           JCENUM = THE ORDINATE ENTRY NUMBER
C           JCRORC = 0 FOR ROW OR 1 FOR COLUMN
C           JCSENS = SEE SIMOPT
C           JCOTPT = OUTPUT ENTRY NUMBER
C           JCCLNO = A USER ASSIGNED CALL STATEMENT ID. PARAMETER
C
C       REMARKS
C           THE NUMBER REPRESENTED BY JCTCIA MUST, FOR SOME MACHINES,
C           REPLACE THE 1 IN THE NAMED COMMON STATEMENT.
C
C       SUBROUTINE AND FUNCTION SUBPROGRAMS REQUIRED
C           FETCH
C           J7OSUP
C           NEWCEL
C           RETURN
C           STORE
C           SYSENT
C           SYSERR
C           SYSEXT
C
C       ERROR CODES FOR THIS PROGRAM
C           CODE FATAL  ERROR(DATA PROVIDED)
C           1    NF     NEGATIVE NUMBER OF UNASSIGNED OUTPUTS (ENT. NO)
C           2    NF     UNASSIGNED OUTPUT COUNT GREATER THAN UNITY
C                       (ENTRY NUMBER)
C           3    NF     OUTPUT DOES NOT MATCH JCOTPT (ENTRY NUMBER)
C           4    NF     JCOTPT NOT FOUND ON OUTPUT LIST (ENTRY NUMBER)
C
C       METHOD
C           SELF-EXPLANATORY
C
C                                **********
C
      SUBROUTINE J7OTPT(JCSPTR,JCENUM,JCRORC,JCSENS,JCOTPT,JCCLNO)
      COMMON/ALLOC/JCCORE(1)
      DIMENSION C(4),O(7),DL(2),L(2)
      INTEGER C,O,D
      DATA C/1,11,1,1/,O/2,12,1,1,13,1,1/,L/1,0/
C                 ENTER
      CALL SYSENT(1,3,2,15,JCCLNO)
C                 INITIALIZE SPUR POINTERS
      M1=JCSPTR+2
      M1=JCCORE(M1)
      MA=M1-5
      MB=M1+5
```

```
C                    INITIALIZE LOAD VECTOR POINTERS
      MC=J70SUP(JCSPTR,1,JCRORC,0,1)
      ME=2
C                    INITIALIZE L VECTOR
      M1=MC+7
      L(2)=JCENUM-JCCORE(M1)
      M2=0
C                    SERVICE COUNT ORDINATE
    1 IF(IABS(JCSENS).EQ.2)GO TO 4
      CALL FETCH(JCCORE(MA),L,M1,C,L,JCCORE(MC),0,1)
      L(2)=0
      M2=M1
      M1=M1-JCSENS/IABS(JCSENS)
      IF(M1.GE.0)GO TO 2
      CALL SYSERR(1,JCENUM)
      M1=0
      GO TO 3
    2 IF((M1*JCRORC).LE.JCRORC)GO TO 3
      CALL SYSERR(2,JCENUM)
      M1=1
    3 CALL STORE(JCCORE(MA),L,M1,C,L,JCCORE(MC),1)
C                    CHECK FOR MULTIPLE OUTPUT
    4 CALL FETCH(JCCORE(MA),L,D,0,L,JCCORE(MC),0,2)
      L(2)=0
      IF((D(1).EQ.1).OR.(M2.GT.1))GO TO 7
C                    SINGLE OUTPUT
      M1=D(2)
      D(2)=JCOTPT
      IF(JCSENS.GT.0)GO TO 6
      IF(M1.NE.D(2))CALL SYSERR(3,JCENUM)
      D(2)=0
    6 CALL STORE(JCCORE(MA),L,D(2),D(4),L,JCCORE(MC),ME)
      GO TO 11
C                    MULTIPLE OUTPUT
    7 D(1)=1
      M1=D(2)
      M2=0
      IF(JCSENS.LT.0)GO TO 9
C                    ASSIGN OUTPUT
      CALL NEWCEL(JCCORE(MB),D(2),1)
      CALL STORE(JCCORE(MA),L,D,0,L,JCCORE(MC),3)
      M2=D(2)
      JCCORE(M2)=M1
      M2=M2+1
      JCCORE(M2)=JCOTPT
      GO TO 11
C                    ERASE OUTPUT
    8 M2=M1
      M1=D(2)
    9 IF(M1.GT.0)GO TO 10
      CALL SYSERR(4,JCENUM)
      GO TO 11
   10 D(2)=JCCORE(M1)
      M3=M1+1
      IF(JCCORE(M3).NE.JCOTPT)GO TO 8
      CALL RETURN(JCCORE(MB),M1,M1,1)
      ME=4
      IF(M2.EQ.0)GO TO 6
      JCCORE(M2)=D(2)
C                    EXIT
   11 CALL SYSEXT
      RETURN
```

END

```
C
C*****************************************************************
C                          **********
C                          * J7VARI *
C                          **********
C
C       PURPOSE
C           J7VARI HANDLES VARIABLES WITH RESPECT TO SIM GENERATION.
C
C       USAGE
C           CALL J7VARI(JCSPUR,JCPOFP,JCSPTR,JCROWN,JCMODE,JCLOAD,
C           JCUNIT,JCCLNO)
C
C       DATA FORMAT
C           N/A
C
C       DESCRIPTION OF PARAMETERS
C           JCSPUR = THE ADDRESS OF THE SPUR FOR SECEDE IF JCMODE = 5,
C                    OR FOR GENDER OTHERWISE
C           JCPOFP = THE DISPLACEMENT TO THE FILE ENTRY IF JCMODE = 5,
C                    OR THE ADDRESS OF A LIST ENTRY
C           JCSPTR = THE ADDRESS OF THE SIMBL
C           JCROWN = THE ROW NUMBER
C           JCMODE = A POSITIVE INTEGER HAVING THE FOLLOWING STATES.
C                    1 = TEAR VARIABLE, ROW ORDINATE SET UP
C                    2 = DECISION VARIABLE, ROW ORDINATE SET UP
C                    3 = OUTPUT VARIABLE, NOT PROTECTED GROUP, ROW
C                        ORDINATE SET UP
C                    4 = OUTPUT VARIABLE, PROTECTED GROUP, ROW ORDINATE
C                        SET UP
C                    5 = INCIDENCE VARIABLE
C                    6 = DECISION VARIABLE, POST SET UP INITIALIZATION
C                    7 = OUTPUT VARIABLE, POST SET UP INITIALIZATION
C           JCLOAD = THE ADDRESS OF THE LOAD VECTOR FOR SECEDE
C           JCUNIT = THE UNIT NUMBER
C           JCCLNO = A USER ASSIGNED CALL STATEMENT ID. PARAMETER
C
C       REMARKS
C           THE NUMBER REPRESENTED BY JCTCIA MUST, FOR SOME MACHINES,
C           REPLACE THE 1 IN THE NAMED COMMON STATEMENT.
C
C       SUBROUTINE AND FUNCTION SUBPROGRAMS REQUIRED
C           BSTLNK
C           FETCH
C           FIND
C           FRMCEL
C           FSTLNK
C           J7CONV
C           J7ICOM
C           J7LOCV
C           J7OSUP
C           J7OTPT
C           LNKBWD
C           LNKFWD
C           NEWCEL
C           RETURN
C           SIMCIN
C           SIMFLG
C           SIMOPT
C           STORE
C           SYSENT
C           SYSEXT
```

```
C            TOCELL
C
C         ERROR CODES FOR THIS PROGRAM
C             NONE
C
C         METHOD
C             SELF-EXPLANATORY
C
C                                  **********
C
      SUBROUTINE J7VARI(JCSPUR,JCPOFP,JCSPTR,JCROWN,JCMODE,JCLOAD,
     1 JCUNIT,JCCLNO)
      COMMON/ALLOC/JCCORE(1)
      DIMENSION L(5),F(3),G(7),S(16),E(13),N(7),C(7),P(4),D(6),O(4)
      INTEGER F,G,S,E,C,P,D,O
      DATA L/3,1,1,1,1,0/,F(1)/1/,G/2,2,1,1,21,1,1/,S/5,19,1,1,8,1,1,7,1,
     1 1,5,1,1,1,1,1/,E/4,6,1,1,5,1,1,3,1,1,4,1,1/,N/2,7,1,1,9,1,1/,C/1,
     2 10,1,1,11,1,1/,P/1,6,1,1/,O/1,13,1,1/
C                    ENTER
      CALL SYSENT(1,3,2,16,JCCLNO)
C                    PARAMETERS
C                    MA=ADDRESS OF SPUR'S FOR SIM
      M1=JCSPTR+2
      MA=JCCORE(M1)
C                    MB=ADDRESS OF SPUR FOR ORDINATE
      MB=MA-5
C                    MC=ADDRESS OF SPUR FOR ELEMENT
      MC=MA-10
C                    ORDINATE
      MD=J7CSUP(JCSPTR,1,1,0,1)
      ME=1
      M1=MD+14
      IF(JCCORE(M1).NE.-1)ME=4
      MF=MD+7
      MG=MD+8
      M1=JCCORE(17)+18
      F(2)=JCCORE(M1)
      MAA=0
C                    MH=NUMBER OF VARIABLES
      MH=0
C                    MS=ADDRESS OF LOAD-II FOR SECEDE
      MS=JCLOAD+30
      M1=MS+28
      JCCORE(M1)=-1
C                    MU=UNIT SCALE
      MU=JCCORE(17)+6
      MU=JCCORE(MU)
C                    DATA
      IF(JCMODE.EQ.5)GO TO 1
   99 CALL FRMCEL(JCCORE(JCSPUR),JCPOFP,D,G,1)
      M1=JCCORE(17)+10
      M2=JCPOFP
      GO TO 2
    1 L(3)=JCPOFP
      CALL FETCH(JCCORE(JCSPUR),L(2),D,S,F,JCCORE(JCLOAD),O,1)
      JCPOFP=C(5)
      L(3)=1
      IF(O(4).NE.1)GO TO 31
      M1=JCCORE(17)+3
      M2=0
C                    COMMON CHECK
      D(2)=O(2)+JCUNIT*MU
```

```
      CALL J7COMV(JCSPUR,D(2),MS,1)
C                     DIMENSION CHECK
C                     MI=INCICATOR
    2 MI=0
      M3=1
      IF(D(1).EQ.0)GO TO 11
C                     DIMENSIONED
C                     MI=MODE FOR J7ICOM CALL
C                     MJ=JCPOSP FOR J7ICOM CALL
      MI=4
      MJ=1
      MN=M2+JCCORE(M1)
      GO TO (7,7,6,6,3,5,4),JCMODE
C                     MODE=1
    3 MI=1
      GO TO 7
C                     MODE=2
    4 MI=2
C                     GET SPACE (MJ)
    5 M2=JCCORE(17)+14
      M2=MA+5*(JCCORE(M2)+1)
      CALL NEWCEL(JCCORE(M2),MJ,1)
      GO TO 7
C                     MODE=2
    6 MI=2
C                     COMPUTE INCICES
    7 CALL J7ICOM(JCSPUR,MN,MA,MJ,2,MI,D(1),JCLOAD,M2,1)
      IF(M2.EC.-1)GO TO 95
      GO TO (8,14,14,14,8,8,8),JCMODE
C                     SEARCH
    8 M2=0
      IF(ME.EQ.1)GO TO 9
      JCCORE(MJ)=C(2)
      M2=MJ+1
      JCCORE(M2)=C(1)
      CALL J7LODV(JCSPTR,L(4),MJ,M2,MD,1)
    9 M2=JCMODE+7*M2
      GO TO (14,14,14,14,14,33,95,10,14,14,14,10,20,20),M2
C                     RETURN MJ
   10 M2=MA+5*(D(1)+1)
      CALL RETURN(JCCORE(M2),MJ,MJ,1)
      MJ=1
      IF(JCMODE.EQ.5)GO TO 22
      GO TO 33
C                     SEARCH
   11 GO TO (12,15,15,15,12,12,12),JCMODE
   12 M2=0
      IF(ME.EQ.1)GO TO 13
      M2=3
      CALL FIND(JCCORE(M8),L(4),C,N,F,JCCORE(MO),M2,1)
   13 M2=JCMODE+7*(M2/3)
      GO TO (15,15,15,15,15,95,95,95,15,15,15,22,20,20),M2
C                     RECORD NAME
   14 M3=4
      D(5)=MJ
      D(4)=1
   15 L(5)=JCCORE(MG)-JCCORE(MF)+1
      CALL STORE(JCCORE(M8),L(ME),D(M3),N,F,JCCORE(MO),1)
      L(5)=0
      ME=4
      IF(JCMODE.EQ.4)GO TO 16
C                     NUMBER OF OUTPUTS = 1
```

```
      CALL STORE(JCCORE(MB),L(4),F,C(4),F,JCCORE(MD),2)
      GO TO (151,16,18,16,22,20,20),JCMODE
  151 F(1)=4
C                     COLUMN STATUS FLAG = 1
   16 CALL STORE(JCCORE(MB),L(4),F,E(4),F,JCCORE(MD),3)
      F(1)=1
      GO TO (17,32,18,18,22,32,95),JCMODE
C                     OUTPUT ASSIGNMENT - COLUMN ONLY
   17 CALL J7CTPT(JCSPTR,JCCORE(MF),1,2,JCROWN,1)
      GO TO 32
C                     OUTPUT ASSIGNMENT - BOTH ROW AND COLUMN
   18 M1=2
   19 CALL SIMCPT(JCSPTR,JCROWN,JCCORE(MF),M1,1)
      IF(JCMODE.EQ.5)GO TO 21
      GO TO 95
C                     GET CURRENT STATUS FLAG
   20 CALL FETCH(JCCORE(MB),L(4),M1,E(4),F,JCCORE(MD),0,2)
      IF(JCMODE.EQ.5)GO TO 23
C                     TEST FLAG
      IF(M1.NE.1)GO TO 21
      CALL SYSERR(1,JCCORE(MF))
      GO TO 32
C                     REMOVE COLUMN
   21 CALL SIMCIN(JCSPTR,JCCORE(MF),1,-1,0,F,F,1)
      CALL SIMFLG(JCSPTR,JCCORE(MF),1,F,0,1)
      IF(JCMODE.EQ.5)GO TO 30
      GO TO 16
C                     ADD ELEMENT TO SIM
   22 D(4)=0
      D(5)=JCROWN
      D(6)=JCCORE(MF)
C                     GET SPACE (MK)
      CALL NEWCEL(JCCORE(MC),MK,2)
      GO TO 20
C                     GET ROW STATUS FLAG
   23 IF(MAA.GT.0)GO TO 24
C                     ROW ORDINATE
      ML=J7OSUP(JCSPTR,1,0,0,2)
      MAA=1
      MM=ML+7
C                     SET LISTER
      L(5)=JCROWN-JCCORE(MM)
C                     GET FLAG
      CALL FETCH(JCCORE(MB),L(4),F(3),E(4),F,JCCORE(ML),0,3)
      L(5)=0
C                     DETERMINE ELEMENT STATUS FLAG
   24 IF((M1+F(3)).GT.0)D(4)=1
      IF(M1.NE.4)GO TO 241
      CALL FETCH(JCCORE(MB),L(4),F(3),0,F,JCCORE(MD),0,31)
      IF(F(3).NE.JCCORE(MM))D(4)=4
C                     STORE DATA
  241 CALL TOCELL(JCCORE(MC),MK,D(3),E,1)
C                     THREAD VERTICALLY
      CALL FETCH(JCCORE(MB),L(4),M2,P,F,JCCORE(MD),0,4)
      CALL STORE(JCCORE(MB),L(4),MK,P,F,JCCORE(MD),4)
      CALL BSTLNK(JCCORE(MC),MK,M2,1)
C                     THREAD HORIZONTALLY
      L(5)=JCROWN-JCCORE(MM)
      CALL FETCH(JCCORE(MB),L(4),M1,P,F,JCCORE(ML),0,5)
      L(5)=0
      M2=0
      M3=0
```

```
      IF(M1.GT.0)GO TO 27
C                     FIRST ELEMENT IN ROW
   25 CALL STORE(JCCORE(MB),L(4),MK,P,F,JCCORE(ML),5)
C                     LINK MK TO NEXT ELEMENT
   26 CALL FSTLNK(JCCORE(MC),MK,M2,1)
      GO TO 29
C                     ROW NOT EMPTY
   27 M2=M1
      CALL FRMCEL(JCCORE(MC),M2 ,M1,E(10),2)
      IF(M1.GT.D(6))GO TO 28
      M3=M2
      M1=LNKFWD(JCCORE(MC),M2,1)
      IF(M1.GT.0)GO TO 27
C                     LAST ELEMENT OF ROW
      M2=0
C                     SET LINK OF PRECEEDING ELEMENT TO MK
   28 IF(M3.EQ.0)GO TO 25
      CALL FSTLNK(JCCORE(MC),M3,MK,2)
      GO TO 26
C                     TEST ELEMENT STATUS FLAG
   29 IF(D(4).EQ.1)GO TO 31
C                     INCREMENT COLUMN COUNT
      CALL FETCH(JCCORE(MB),L(4),M2,C,F,JCCORE(MD),0,6)
      M2=M2+1
      CALL STORE(JCCORE(MB),L(4),M2,C,F,JCCORE(MD),6)
C                     INCREMENT ROW COUNT
      MH=MH+1
      CALL STORE(JCCORE(MB),L(4),MH,C,F,JCCORE(ML),7)
C                     COST
      IF(D(3).GT.0)GO TO 31
      M1=1
      GO TO 19
C                     REMOVE ROW IF LAST OUTPUT
   30 CALL FETCH(JCCORE(MB),L(4),M2,C(4),F,JCCORE(ML),0,7)
      IF(M2.GT.0)GO TO 31
      CALL SIMCIN(JCSPTR,JCROWN,C,-1,0,F,F,2)
      CALL SIMFLG(JCSPTR,JCROWN,0,F,0,2)
      F(3)=1
C                     NEXT CONSTITUENT
   31 IF(JCPOFP.EQ.0)GO TO 35
      IF((D(1).GT.0).AND.(D(4).EQ.1))JCPOFP=1
      GO TO 1
   95 JCPOFP=LNKFWD(JCCORE(JCSPUR),JCPOFP,2)
      IF(JCPOFP.EQ.0)GO TO 34
      GO TO 99
C                     ITERATE
   32 IF(D(1).EQ.0)GO TO 95
   33 MI=-IABS(MI)
      GO TO 7
C                     RETURN MJ
   34 IF(JCMODE.LT.6)GO TO 35
      M2=JCCORE(17)+14
      M2=MA+5*(JCCORE(M2)+1)
      CALL RETURN(JCCORE(M2),MJ,MJ,2)
C                     EXIT
   35 CALL SYSEXT
      RETURN
      END
```

```
C
C*****************************************************************
C                              **********
C                              * BREAK  *
C                              **********
C
C       PURPOSE
C           BREAK DESTROYS THE LINKAGE BETWEEN TWO SPECIFIED NODES.
C
C       USAGE
C           CALL BREAK(JCNTWK,JCPOFP,JCPOSP,JCPOSN,JCADBN,JCCLNO)
C
C       DATA FORMAT
C           N/A
C
C       DESCRIPTION OF PARAMETERS
C           JCNTWK = THE ADDRESS OF THE NETWORK INFORMATION BLOCK
C           JCPOFP = 0 OR THE ADDRESS OF THE FIRST NODE
C           JCPOSP = 0 OR THE ADDRESS OF THE SECOND NODE
C           JCPOSN = SIMILAR TO JCPOSF OF COUPLE, EXCEPT THAT JCPOSN
C                    REFERS TO WHICHEVER NODE (IF ANY) IS ASSIGNED A
C                    JCPO P OF 0.  IF NEITHER JCPOFP OR JCPOSP ARE 0,
C                    JCPOSN SHOULD BE ASSIGNED A VALUE OF -1.
C           JCADBN = SEE COUPLE
C           JCCLNO = A USER ASSIGNED CALL STATEMENT ID. PARAMETER
C
C       REMARKS
C           THE NUMBER REPRESENTED BY JCTCIA MUST, FOR SOME MACHINES,
C           REPLACE THE 1 IN THE NAMED COMMON STATEMENT.
C
C       SUBROUTINE AND FUNCTION SUBPROGRAMS REQUIRED
C           FRMCEL
C           FSTLNK
C           RETURN
C           SYSENT
C           SYSERR
C           SYSEXT
C           TOCELL
C
C       ERROR CODES FOR THIS PROGRAM
C           CODE FATAL  ERROR(DATA PROVIDED)
C           1    NF     NODES ARE NOT LINKED (ADDRESS OF NODE)
C           2    NF     NODE ON DIVERGER (IDENTIFICATION FLAG OF NODE)
C
C       METHOD
C           EITHER ERROR CAUSES BREAK TO ABORT.   POSSIBLE MODES OF
C           OPERATION ARE (A,A,-1), (A,0,M) OR (0,A,M) FOR (JCPOFP,
C           JCPOSP,JCPOSN), WHERE A IS AN ADDRESS AND M IS -1,0 OR A
C           POSITIVE INTEGER.
C
C                              **********
C
      SUBROUTINE BREAK(JCNTWK,JCPOFP,JCPOSP,JCPOSN,JCADBN,JCCLNO)
      COMMON/ALLOC/JCCORE(1)
      DIMENSION P(7),G(10),D(3)
      INTEGER P,G,D
      DATA P/2,1,1,1,5,1,1/,G/3,1,1,1,2,1,1,3,1,1/
C                ENTER
      CALL SYSENT(1,3,3,1,JCCLNO)
C                SPUR
      M1=JCNTWK+1
      M1=JCCORE(M1)
```

```
      M2=M1+4
      M2=JCCORE(17)+7+14*(JCCORE(M2)/5)-2*(JCCORE(M2)/6)
      MA=M1+5*(JCCORE(M2)-1)
      M2=M2+1
      MAA=M1+5*(JCCORE(M2)-1)
C                     MODE
      MB=JCPOFP
      MC=JCPOSP
      IF(JCPOSN.EQ.0)JCPOSN=1
      MD=JCPOSN
      MH=0
      IF((MB.GT.0).AND.(MC.GT.0))MD=-1
      IF(MD.EQ.-1)MH=1
      P(1)=2
      P(2)=1
      P(6)=1
      P(7)=1
      IF(MB.GT.0)GO TO 2
C                     REVERSE
    1 M1=MB
      MB=MC
      MC=M1
      P(1)=2
      P(2)=3-P(2)
      P(6)=P(2)
      P(7)=P(2)
C                     EXAMINE LINK FROM NODE
    2 ME=0
      MF=0
      MG=1
      CALL FRMCEL(JCCORE(MAA),MB,D,P,1)
      MI=D(2)-1
      P(1)=1
      IF(D(1).EQ.0)GO TO 11
C                     FLAG
      CALL FRMCEL(JCCORE(MA),D(1),M1,G(7),2)
      IF(M1.EQ.0)GO TO 4
C                     NODE TO NODE
      D(2)=0
      IF(D(1).EQ.MC)GO TO 8
      IF(MC.GT.0)GO TO 11
      MC=D(1)
      GO TO 8
C                     DIVERGER
    3 IF(D(1).EQ.0)GO TO 10
      MG=MG+1
    4 ME=MF
      MF=D(1)
      CALL FRMCEL(JCCORE(MA),MF,D,G,3)
      IF(D(3).NE.0)GO TO 12
C                     TEST
      IF((D(2).NE.MC).AND.(MG.NE.MD))GO TO 3
C                     SUCCESS
    5 MC=D(2)
      CALL RETURN(JCCORE(MA),MF,MF,1)
      IF(ME.EQ.0)GO TO 6
C                     NOT FIRST
      CALL FSTLNK(JCCORE(MA),ME,D(1),1)
      GO TO 7
C                     FIRST
    6 CALL TOCELL(JCCORE(MA),MB,D(1),P,1)
    7 IF(JCADBN.GT.0)GO TO 9
```

```
C                    TEST FOR SINGLE PATH
      CALL FRMCEL(JCCORE(MA),MB,ME,P,4)
      IF(ME.EQ.0)GO TO 9
      CALL FRMCEL(JCCORE(MA),ME,D,G,5)
      IF(D(1).GT.0)GO TO 9
C                    REMOVE DIVERGER ENTRY
      IF(D(3).NE.0)GO TO 12
      CALL RETURN(JCCORE(MA),ME,ME,2)
    8 CALL TOCELL(JCCORE(MA),MB,D(2),P,2)
C                    SECOND PASS
    9 CALL TOCELL(JCCORE(MAA),MB,MI,P(4),3)
      IF(P(2).EQ.2)GO TO 91
      JCPOFP=MB
      JCPOSP=MC
      IF((MH*JCPOSN).LE.0)GO TO 93
      GO TO 92
   91 IF((MH*JCPOSN).GE.0)GO TO 93
   92 JCPOSN=MG
      MH=0
   93 IF(M2.LT.0)GO TO 13
      MD=-1
      M2=-1
      GO TO 1
C                    NO CONNECTION
   10 IF(MC.GT.0)GO TO 11
      JCPOSN=MG
      GO TO 5
   11 CALL SYSERR(1,MB)
      GO TO 9
C                    NODE ON DIVERGER
   12 CALL SYSERR(2,D(3))
      GO TO 9
   13 CALL SYSEXT
      RETURN
      END
```

```
C
C*********************************************************************
C                              **********
C                              * COUPLE *
C                              **********
C
C       PURPOSE
C           COUPLE JOINS TWO SPECIFIED NODES.
C
C       USAGE
C           CALL COUPLE(JCNTWK,JCPOFP,JCPOSP,JCPOSF,JCPOSS,JCADBN,
C           JCCLNO)
C
C       DATA FORMAT
C           N/A
C
C       DESCRIPTION OF PARAMETERS
C           JCNTWK = THE ADDRESS OF THE NETWORK INFORMATION BLOCK
C           JCPOFP = THE ADDRESS OF THE FIRST NODE
C           JCPOSP = THE ADDRESS OF THE SECOND NODE
C           JCPOSF = -1, 0 OR N.  IF JCPOSF IS -1 OR 0, THE NEW PATH
C                    FROM JCPOFP IS POSITIONED AT THE END OR BEGINNING
C                    OF THE DIVERGER RESPECTIVELY.  IF EQUAL TO N,
C                    JCPOSF INDICATES THE POSITION IN THE DIVERGER THAT
C                    THE NEW PATH IS TO OCCUPY.
C           JCPOSS = THE SAME AS THE JCPOSF, BUT FOR THE REVERSE PATH
C                    FROM JCPOSP TO JCPOFP
C           JCADBN = 0 OR 1 AS RESPECTIVELY A DIVERGER IS NOT OR IS
C                    ALWAYS REQUIRED BETWEEN NODES
C           JCCLNO = A USER ASSIGNED CALL STATEMENT ID. PARAMETER
C
C       REMARKS
C           THE NUMBER REPRESENTED BY JCTCIA MUST, FOR SOME MACHINES,
C           REPLACE THE 1 IN THE NAMED COMMON STATEMENT.
C
C       SUBROUTINE AND FUNCTION SUBPROGRAMS REQUIRED
C           FRMCEL
C           FSTLNK
C           LNKFWD
C           NEWCEL
C           SYSENT
C           SYSEXT
C           TOCELL
C
C       ERROR CODES FOR THIS PROGRAM
C           NONE
C
C       METHOD
C           SELF-EXPLANATORY
C
C                              **********
C
      SUBROUTINE COUPLE(JCNTWK,JCPOFP,JCPOSP,JCPOSF,JCPOSS,JCADBN,
     1 JCCLNO)
      COMMON/ALLOC/JCCORE(1)
      DIMENSION P(7),G(10),O(5)
      INTEGER P,G,O
      DATA P/2,1,1,1,5,1,1/,G/3,1,1,1,2,1,1,3,1,1/,O(3)/0/
C                 ENTER
      CALL SYSENT(1,3,3,2,JCCLNO)
C                 SPUR
      MI=JCNTWK+1
```

```
      M1=JCCORE(M1)
      M2=M1+4
      M2=JCCORE(17)+7+14*(JCCORE(M2)/5)-2*(JCCORE(M2)/6)
      MA=M1+5*(JCCORE(M2)-1)
      M2=M2+1
      MAA=M1+5*(JCCORE(M2)-1)
      M2=0
C                     FORWARD PATH
      P(1)=2
      P(2)=1
      P(6)=1
      P(7)=1
      MB=JCPOFP
      D(2)=JCPOSP
      MC=JCPOSF
C                     CURRENT LINK
    1 CALL FRMCEL(JCCORE(MAA),MB,D(4),P,1)
      D(5)=D(5)+1
      CALL TOCELL(JCCORE(MAA),MB,D(5),P(4),1)
      P(1)=1
      D(1)=D(4)
      IF(MC.EQ.0)MC=1
      MD=0
      IF(D(1).GT.0)GO TO 2
C                     END OF PATH
      IF(JCADBN.GT.0)GO TO 5
C                     NO DIVERGER REQUIRED
      ME=D(2)
      GO TO 6
C                     FIND POSITION
    2 CALL FRMCEL(JCCORE(MA),D(1),M1,G(7),2)
      IF(M1.EQ.0)GO TO 4
C                     NODE TO NODE
      M1=D(2)
      D(2)=D(1)
      D(1)=0
      M2=1
      GO TO 5
C                     CONTINUE
    3 D(2)=M1
      D(1)=ME
      M2=0
C                     DIVERGER
    4 IF(MC.EQ.1)GO TO 5
      MD=D(1)
      MC=MC-1
      D(1)=LNKFWD(JCCORE(MA),MD,1)
      IF(D(1).GT.0)GO TO 4
C                     GET DIVERGER ENTRY
    5 CALL NEWCEL(JCCORE(MA),ME,1)
C                     STORE DATA
      CALL TOCELL(JCCORE(MA),ME,D,G,2)
      IF(MD.EQ.0)GO TO 6
C                     THREAD INTO DIVERGER
      CALL FSTLNK(JCCORE(MA),MD,ME,1)
      GO TO 7
C                     SET NODE LINK
    6 D(4)=ME
      CALL TOCELL(JCCORE(MAA),MB,D(4),P,3)
      IF(M2.EQ.1)GO TO 3
    7 IF(MB.EQ.JCPOSP)GO TO 8
C                     BACKWARD PATH
```

```
      D(2)=MB
      MB=JCPOSP
      MC=JCPOSS
      P(1)=2
      P(2)=2
      P(6)=2
      P(7)=2
      GO TO 1
C                     EXIT
    8 CALL SYSEXT
      RETURN
      END
```

```
C
C*****************************************************************
C                              **********
C                              * LNKBNT *
C                              **********
C
C       PURPOSE
C           LNKBNT RETRIEVES THE BACKWARD LINK FROM A SPECIFIED NODE.
C
C       USAGE
C           LNKBNT(JCNTWK,JCPOFP,JCPOSP,JCWHDE,JCCLNO)
C
C       DATA FORMAT
C           N/A
C
C       DESCRIPTION OF PARAMETERS
C           JCNTWK = THE ADDRESS OF THE NETWORK INFORMATION BLOCK
C           JCPOFP = THE ADDRESS OF A NODE OR DIVERGER ENTRY
C           JCPOSP = THE ADDRESS OF THE NEXT DIVERGER ENTRY
C           JCWHDE = -1, 0 OR N, WHERE N IS A POSITIVE INTEGER.  IF
C                    JCPOFP IS A NODE, N IS THE SEQUENCE NUMBER OF A
C                    DIVERGER ENTRY.  IF JCPOFP IS A DIVERGER ENTRY,
C                    N IS THE NUMBER OF ENTRIES TO ADVANCE BEFORE
C                    RETRIEVING THE ADDRESS OF A NODE (I.E. LNKBNT).
C                    IF JCWHDE = 0, THE FIRST PATH IS TO BE SELECTED.
C                    IF JCWHDE = -1, THE LAST PATH IS TO BE SELECTED.
C           JCCLNO = A USER ASSIGNED CALL STATEMENT ID. PARAMETER
C
C       REMARKS
C           THE NUMBER REPRESENTED BY JCTCIA MUST, FOR SOME MACHINES,
C           REPLACE THE 1 IN THE NAMED COMMON STATEMENT.
C
C       SUBROUTINE AND FUNCTION SUBPROGRAMS REQUIRED
C           J7LINK
C           SYSENT
C           SYSEXT
C
C       ERROR CODES FOR THIS PROGRAM
C           NONE
C
C       METHOD
C           SELF-EXPLANATORY
C
C                              **********
C
      FUNCTION LNKBNT(JCNTWK,JCPOFP,JCPOSP,JCWHDE,JCCLNO)
      COMMON/ALLOC/JCCORE(1)
C                 ENTER
      CALL SYSENT(1,3,3,3,JCCLNO)
C                 GET LINK
      LNKBNT=J7LINK(JCNTWK,JCPOFP,JCPOSP,JCWHDE,-1,1)
C                 EXIT
      CALL SYSEXT
      RETURN
      END
```

```
C
C*****************************************************************
C                              **********
C                              * LNKFNT *
C                              **********
C
C       PURPOSE
C           LNKFNT RETRIEVES THE FORWARD LINK FROM A SPECIFIED NODE.
C
C       USAGE
C           LNKFNT(JCNTWK,JCPOFP,JCPOSP,JCWHDE,JCCLNO)
C
C       DATA FORMAT
C           N/A
C
C       DESCRIPTION OF PARAMETERS
C           SEE LNK8NT
C
C       REMARKS
C           THE NUMBER REPRESENTED BY JCTCIA MUST, FOR SOME MACHINES,
C           REPLACE THE 1 IN THE NAMED COMMON STATEMENT.
C
C       SUBROUTINE AND FUNCTION SUBPROGRAMS REQUIRED
C           SEE LNK8NT
C
C       ERROR CODES FOR THIS PROGRAM
C           NONE
C
C       METHOD
C           SELF-EXPLANATORY
C
      FUNCTION LNKFNT(JCNTWK,JCPOFP,JCPOSP,JCWHDE,JCCLNO)
      COMMON/ALLOC/JCCORE(1)
C                 ENTER
      CALL SYSENT(1,3,3,4,JCCLNO)
C                 GET LINK
       LNKFNT=J7LINK(JCNTWK,JCPOFP,JCPOSP,JCWHDE,1,1)
C                 EXIT
      CALL SYSEXT
      END
```

```
C
C*****************************************************************
C                              **********
C                              *  NEXT  *
C                              **********
C
C
C       PURPOSE
C           NEXT SPECIFIES THE NEXT NODE IN A TRACE GIVEN THE CURRENT
C           NODE.
C
C       USAGE
C           NEXT(JCNTWK,JCPOFP,JCTKEY,JCOOFT,JCLEVL,JCHDCL,LION,JCDATA,
C           JCCLNO)
C
C       DATA FORMAT
C           N/A
C
C       DESCRIPTION OF PARAMETERS
C           JCNTWK = THE ADDRESS OF THE NETWORK INFORMATION BLOCK
C           JCPOFP = THE ADDRESS OF THE CURRENT NODE
C           JCTKEY = THE TRACE KEY
C           JCOOFT = +N FOR FORWARD TRACE OR -N FOR BACKWARD TRACE,
C                    WHERE N IS A POSITIVE INTEGER.  IF N = 1, THE NODE
C                    JCPOFP IS INSPECTED WITH RESPECT TO KEY AND COUNT
C                    MATCH.  IF N = 2, NEXT WILL ALWAYS ADVANCE FROM
C                    JCPOFP UNLESS JCPOFP IS THE LAST NODE.
C           JCLEVL = -1 OR A NON-NEGATIVE INTEGER N.  IF JCLEVL EQUALS
C                    -1, NEXT WILL NOT ENTER A NODE TO INSPECT SUB-
C                    NODES.  IF JCLEVL EQUALS N, N IS THE LEVEL OF THE
C                    CURRENT NODE.
C           JCHDCL = THE ADDRESS OF THE TEMPORARY LIST HEADCELL
C           LION   = A VECTOR OF DATA FOR STORAGE IN THE TEMPORARY LIST
C                    WHEN NEXT MUST ENTER A SUB-NODE LIST.
C           JCDATA = THE NUMBER OF DATA ITEMS IN LION (0 IF NONE)
C           JCCLNO = A USER ASSIGNED CALL STATEMENT ID. PARAMETER
C
C       REMARKS
C           THE NUMBER REPRESENTED BY JCTCIA MUST, FOR SOME MACHINES,
C           REPLACE THE 1 IN THE NAMED COMMON STATEMENT.
C           IF THE TRACE IS COMPLETE, NEXT IS RETURNED AS 0.
C           THE LEVEL PASSED TO NEXT IS THAT APPROPRIATE TO JCPOFP,
C           WHILE UPON RETURN THE JCLEVL IS THAT OF THE NODE INDICATED
C           BY THE VALUE OF NEXT.
C           WHEN NEXT ENTERS A SUB-NODE LIST, INFORMATION IS
C           TRANSFERRED FROM LION TO A TEMPORARY LIST ENTRY.  UPON
C           COMPLETION OF A SUB-NODE LIST, INFORMATION IS TRANSFERRED
C           FROM THE TEMPORARY LIST ENTRY TO LION.
C
C       SUBROUTINE AND FUNCTION SUBPROGRAMS REQUIRED
C           FRMCEL
C           LAND
C           J7LINK
C           POPUP
C           PUSH
C           SYSENT
C           SYSERR
C           SYSEXT
C           TOCELL
C
C       ERROR CODES FOR THIS PROGRAM
C           CODE FATAL  ERROR(DATA PROVIDED)
C           1    NF     COUNT MISMATCH WITH NULL TEMPORARY LIST
```

```
C                          (ADDRESS OF NODE)
C
C         METHOD
C             THE KEYS AND COUNTS MUST MATCH FOR NEXT TO ASSUME THE
C             ADDRESS OF A NODE.  NEXT TRAVERSES A PATH THROUGH THE
C             NETWORK UNTIL IT IS FORCED TO BACK-TRACK BY END OF PATH OR
C             COUNT MISMATCH.  BACK-TRACK DATA IS STORED IN THE
C             TEMPORARY LIST.  IF THE KEYS AND COUNTS MATCH FOR JCPOFP,
C             AND JCLEVL IS NON-NEGATIVE, NEXT WILL ENTER THE SUBNODE
C             LIST FOR JCPOFP.
C
C                                **********
C
      FUNCTION NEXT(JCNTWK,JCPOFP,JCTKEY,JCOOFT,JCLEVL,JCHOCL,LION,
     1 JCOATA,JCCLNO)
      COMMON/ALLOC/JCCORE(1)
      DIMENSION N(19),P(4),Q(4),R(4),O(6)
      INTEGER P,Q,R,O
      DATA N/5,3,1,1,4,2,2,4,1,1,5,3,3,5,1,0,7,1,1/,P/1,6,0,0/,
     1 Q/-1,6,0,0/,R/1,6,0,0/
C                     ENTER
      CALL SYSENT(1,3,3,5,JCCLNO)
C                     SET LIMITS
      N(1)=5
      N(15)=2
      IF(JCLEVL.GE.0)N(1)=6
      IF(JCOOFT.LT.0)N(15)=1
      N(16)=N(15)
      M1=JCNTWK+1
      M1=JCCORE(M1)
      M2=M1+4
      MA=JCCORE(M2)+20
      MA=JCCORE(MA)+16
      IF(JCCORE(MA).NE.0)JCTKEY=LAND(JCTKEY,JCCORE(MA))
      IF(JCTKEY.EQ.0)JCTKEY=1
      M2=JCCORE(17)+7+14*(JCCORE(M2)/5)-2*(JCCORE(M2)/6)
      P(3)=JCCORE(M2)+1
      P(4)=P(3)+3
      Q(1)=-1
      Q(4)=P(4)
      IF(JCDATA.LE.0)GO TO 1
      Q(1)=1
      Q(3)=P(4)+1
      Q(4)=P(4)+JCDATA
    1 R(3)=P(3)
      R(4)=P(3)
C                     SPUR'S
      M2=M2+1
      MA=M1+5*(JCCORE(M2)-1)
      MB=M1+5*(Q(4)-1)
      MC=M1+5*(R(4)-1)
C                     SET TEMPORARY PARAMETERS
      NEXT=JCPOFP
      MD=0
      MG=0
      IF((IABS(JCOOFT)).GT.1)GO TO 6
      IF(NEXT.EQ.0)GO TO 5
C                     TEST NEXT
    2 CALL FRMCEL(JCCORE(MA),NEXT,D,N,1)
      IF(D(1).EQ.0)GO TO 6
C                     NODE
      IF(D(3).EQ.JCTKEY)GO TO 9
```

```
C                    SET KEY
      D(3)=JCTKEY
      D(4)=1
      CALL TOCELL(JCCORE(MA),NEXT,D(3),N(7),1)
C                    COMPARE COUNT
    3 IF(D(4).GE.D(5))GO TO 12
      M1=NEXT
      NEXT=0
      ME=1
C                    REMOVE LIST ENTRY
    4 IF(JCHDCL.EQ.0)GO TO 11
      CALL FRMCEL(JCCORE(MB),JCHDCL,M1,R,2)
      IF(M1.EQ.0)GO TO 5
C                    AUXILIARY
      CALL POPUP(JCCORE(MC),M1,M1,NEXT,R,1,1)
      CALL TOCELL(JCCORE(MB),JCHDCL,M1,R,2)
      GO TO 6
C                    MAIN
    5 IF(MG.EQ.1)GO TO 12
      CALL FRMCEL(JCCORE(MB),JCHDCL,LION,Q,3)
      CALL POPUP(JCCORE(MB),JCHDCL,JCHDCL,D,P,1,2)
C                    REDUCE LEVEL
      NEXT=D(2)
      JCTKEY=D(3)
      IF(NEXT.EQ.0)GO TO 12
      JCLEVL=JCLEVL-1
      MG=1
C                    ADVANCE
    6 M1=NEXT
      NEXT=J7LINK(JCNTWK,M1,MF,0,JCDOFT,1)
      ME=0
      IF(NEXT.EQ.0)GO TO 4
C                    ADD LIST ENTRY
      IF(MF.LE.0)GO TO 2
C                    MAIN
      IF(JCHDCL.GT.0)GO TO 8
      D(2)=JCTKEY
    7 D(1)=0
      D(3)=JCTKEY
      JCTKEY=D(2)
      D(2)=MD
      CALL PUSH(JCCORE(MB),JCHDCL,D,P,1,1)
      CALL TOCELL(JCCORE(MB),JCHDCL,LION,Q,3)
      IF(MF.EQ.0)GO TO 2
C                    AUXILIARY
    8 CALL FRMCEL(JCCORE(MB),JCHDCL,M1,R,4)
      CALL PUSH(JCCORE(MC),M1,MF,R,1,2)
      CALL TOCELL(JCCORE(MB),JCHDCL,M1,R,4)
      GO TO 2
C                    INCREMENT COUNT
    9 IF((D(1).EQ.1).AND.(JCLEVL.GE.0).AND.(D(4).GE.D(5)))GO TO 10
      IF(D(4).GE.D(5))GO TO 6
      D(4)=D(4)+1
      CALL TOCELL(JCCORE(MA),NEXT,D(4),N(10),5)
      GO TO 3
C                    ENTER NODE
   10 MD=NEXT
      D(2)=D(2)+1
      CALL TOCELL(JCCORE(MA),MD,D(2),N(4),6)
      CALL FRMCEL(JCCORE(MA),MD,D(2),N(4),5)
      IF(D(2).EQ.0)GO TO 10
      NEXT=D(6)
```

```
      MF=0
      JCLEVL=JCLEVL+1
      GO TO 7
C                     FAILURE
   11 IF(ME.NE.1)GO TO 12
      M2=JCNTWK+1
      M2=JCCORE(M2)+4
      IF(JCCORE(M2).NE.6)CALL SYSERR(1,M1)
C                     EXIT
   12 CALL SYSEXT
      RETURN
      END
```

```
C
C*****************************************************************
C                              **********
C                              * REMOVE *
C                              **********
C
C       PURPOSE
C           REMOVE RETURNS TO THE APPROPRIATE AVAILABLE SPACE LISTS A
C           SPECIFIED SEQUENCE OF NODES.
C
C       USAGE
C           CALL REMOVE(JCNTWK,JCPOFP,JCPOSN,JCDOFT,JCCLNO)
C
C       DATA FORMAT
C           N/A
C
C       DESCRIPTION OF PARAMETERS
C           JCNTWK = THE ADDRESS OF THE NETWORK INFORMATION BLOCK
C           JCPOFP = THE ADDRESS OF THE NODE AT WHICH THE OPERATION IS
C                    TO COMMENCE
C           JCPOSN = 0 IF JCPOFP IS TO BE REMOVED, AND -1 OR A POSITIVE
C                    INTEGER N TO INDICATE WITH WHICH PATH FROM JCPOFP
C                    THE REMOVE OPERATION IS TO COMMENCE.
C           JCDOFT = -1 OR +1 AS RESPECTIVELY BACKWARD OR FORWARD TRACE
C                    OF THE NODES FOR REMOVAL IS DESIRED
C           JCCLNO = A USER ASSIGNED CALL STATEMENT ID. PARAMETER
C
C       REMARKS
C           THE NUMBER REPRESENTED BY JCTCIA MUST, FOR SOME MACHINES,
C           REPLACE THE 1 IN THE NAMED COMMON STATEMENT.
C           REMOVE SHOULD BE EMPLOYED FOR GENDER AND DYNAMIC
C           PROGRAMMING NETWORKS ONLY.
C
C       SUBROUTINE AND FUNCTION SUBPROGRAMS REQUIRED
C           BREAK
C           CORN
C           FRMCEL
C           J7LINK
C           LNKBWD
C           LNKFWD
C           POPUP
C           PUSH
C           RETURN
C           SYSENT
C           SYSEXT
C
C       ERROR CODES FOR THIS PROGRAM
C
C       METHOD
C           REMOVE BEGINS WITH JCPOFP FOR JCPOSN = 0, OR WITH THE NODE
C           FOLLOWING JCPOFP OTHERWISE.  EACH NODE ENCOUNTERED IN THE
C           DIRECTION OF TRACE IS RETURNED, PROVIDED IT POSSESSES AT
C           MOST ONE REVERSE LINK.  THE REVERSE LINK STOPPING CRITERIA
C           PREVENTS THE INADVERTENT DESTRUCTION OF NETWORK FEATURES
C           OTHER THAN THOSE SPECIFICALLY INTENDED.  FOR GENDER
C           NETWORKS, REMOVE ENTERS EACH GROUP AND RETURNS THE GROUP
C           BODY ALONG WITH ALL APPENDAGES.
C
C                              **********
C
      SUBROUTINE REMOVE(JCNTWK,JCPOFP,JCPOSN,JCDOFT,JCCLNO)
      COMMON/ALLOC/JCCORE(1)
```

```
      DIMENSION F(4),G(7),B(10),V(4),E(4),M(10),T(4),C(2),I(4),D(6)
      INTEGER F,G,B,V,E,T,C,D
      DATA F/1,3,1,1/,G/2,7,1,3,8,1,3/,B/3,11,1,1,9,1,1,12,1,1/,V/1,2,1,
     1 1/,E/1,22,1,1/,M/3,17,1,1,16,1,1,18,1,1/,T/1,6,0,0/,I/1,3,2,2/
C                     ENTER
      CALL SYSENT(1,3,3,7,JCCLNO)
C                     SPUR'S - GENERAL
      M1=JCNTWK+ 1
      MA=JCCORE(M1)
      M1=MA+4
      MB=JCCORE(M1)
      M1=JCCORE(17)+7+14*(MB/5)-2*(MB/6)
      M2=JCCORE(M1)+1
      MC=MA+5*M2
      M1=M1+1
      MD=MA+5*(JCCORE(M1)-1)
      IF(MB.NE.2)GO TO 1
C                     ADDITIONAL SPUR'S OR GENDER LIST
      M1=M1+1
      ME=MA+5*(JCCORE(M1)-1)
      M1=M1+1
      MF=MA+5*(JCCORE(M1)-1)
      M1=M1+1
      MG=MA+5*(JCCORE(M1)-1)
      M1=M1+1
      MH=MA+5*(JCCORE(M1)-1)
      M1=M1+1
      MI=MA+5*(JCCORE(M1)-1)
      MJ=MC-10
C                     SET T(3) AND T(4)
    1 T(3)=M2
      T(4)=M2+1
C                     INITIALIZATION
      MK=0
      C(1)=JCPOFP
      ML=JCPOSN
      MM=0
      IF(MB.NE.2)MM=1
      IF(ML.EQ.0)GO TO 4
      GO TO 3
C                     BACK-TRACK
    2 IF(MK.EQ.0)GO TO 20
      CALL POPUP(JCCORE(MC),MK,MK,C,T,1,1)
C                     ADVANCE TO NEXT NODE
    3 M1=J7LINK(JCNTWK,C(1),M2,ML,JCDOFT,1)
      IF(M1.EQ.0)GO TO 2
      ML=0
      C(2)=LNKBWD(JCCORE(MJ),C(2),1)
C                     BREAK LINKS
      CALL BREAK(JCNTWK,C(1),C(2),-1,MM,1)
      C(1)=M1
      C(2)=J7LINK(JCNTWK,C(1),M1,0,-JCDOFT,2)
      IF(M1.GT.0)GO TO 2
      C(2)=M3
C                     ADD TO T. LIST
      IF(M2.GT.0)CALL PUSH(JCCORE(MC),MK,C,T,1,1)
C                     GET FLAGS
    4 IF(MB.NE.2)GO TO 6
      CALL FRMCEL(JCCORE(MD),C(1),D,F,1)
      M1=D(1)+1
      GO TO (3,5,11,16,3,3,3,3),M1
C                     GROUP
```

```
    5 CALL FRMCEL(JCCORE(MD),C(1),D,G,2)
    6 CALL RETURN(JCCORE(MD),C(1),C(1),1)
      IF(MB.NE.2)GO TO 3
C                     DECISION VARIABLE LIST
      M1=2
    7 IF(D(M1).EQ.0)GO TO 8
      CALL FRMCEL(JCCORE(MF),D(M1),M2,V,3)
      M2=MF+5*M2*(1+M1/2)
      M3=D(M1)
      D(M1)=LNKFWD(JCCORE(M2),M3,1)
      CALL RETURN(JCCORE(M2),M3,M3,2)
      GO TO 7
C                     TEAR VARIABLE LIST
    8 M1=M1+1
      IF(M1.EQ.2)GO TO 12
      IF(M1.EQ.3)GO TO 7
C                     SIM
      M1=4
    9 IF(D(M1).EQ.0)GO TO 10
      M2=D(M1)+1
      CALL CORN(JCCORE(M2))
C                     NETWORKS
   10 M1=M1+1
      IF(M1.LT.7)GO TO 9
C                     GROUP BODY
      C(1)=D(1)
      GO TO 3
C                     GROUP BODY
   11 CALL FRMCEL(JCCORE(ME),C(1),D,B,4)
      M1=ME+5*D(3)
      CALL RETURN(JCCORE(M1),C(1),C(1),3)
C                     OUTPUT VARIABLE LIST
      M1=1
      GO TO 7
C                     LINK EDITOR LIST
   12 IF(D(2).EQ.0)GO TO 3
C                     GET FLAG
      CALL FRMCEL(JCCORE(MJ),D(2),M1,F,5)
      M1=M1+1
      GO TO (13,15,13,15,12,12,12,12),M1
C                     OPERATOR OR CONSTANT
   13 M1=MJ
   14 M2=D(2)
      D(2)=LNKFWD(JCCORE(M1),M2,2)
      CALL RETURN(JCCORE(M1),M2,M2,4)
      GO TO 12
C                     VARIABLE
   15 CALL FRMCEL(JCCORE(MG),D(2),M1,E,6)
       M1=MJ+5*M1
      GO TO 14
C                     INSERT
   16 CALL FRMCEL(JCCORE(MH),C(1),M1,I,7)
      IF(M1.EQ.1)GO TO 17
C                     UPDATE
      CALL FRMCEL(JCCORE(MH),C(1),D(2),M(4),8)
      CALL RETURN(JCCORE(MH),C(1),C(1),5)
      GO TO 12
C                     METHOD
   17 CALL FRMCEL(JCCORE(MI),C(1),D,M,9)
      CALL RETURN(JCCORE(MI),C(1),C(1),6)
C                     VARIABLE VALUE BLOCK
      M1=1
```

```
   18 IF(D(M1).EQ.0)GO TO 19
      M2=D(M1)
      M3=MA+5*(D(3)-1)
      CALL RETURN(JCCORE(M2),M2,M2,7)
C                    FUNCTION VALUE BLOCK
   19 M1=M1+1
      IF(M1.EQ.2)GO TO 18
      GO TO 3
C                    EXIT
   20 CALL SYSEXT
      RETURN
      END
```

```
C
C*****************************************************************
C                              **********
C                              * SEARCH *
C                              **********
C
C       PURPOSE
C           SEARCH SEARCHES A NETWORK FOR A SPECIFIED PATTERN OF DATA
C           ITEM VALUES.
C
C       USAGE
C           CALL SEARCH(JCNTWK,JCPOFP,JCPOSP,LION,LOCAL,JCTKEY,JCOOFS,
C           JCLEVL,JCHOCL,JCCLNO)
C
C       CATA FORMAT
C           N/A
C
C       CESCRIPTION OF PARAMETERS
C           JCNTWK = THE ADDRESS OF THE NETWORK INFORMATION BLOCK
C           JCPOFP = THE ADDRESS OF THE NODE ON WHICH THE SEARCH IS TO
C                    COMMENCE
C           JCPOSP = THE ADDRESS OF THE NOOE FOUND TO CONTAIN THE DATA
C                    PATTERN, OR 0 FOR SEARCH FAILURE
C           LICN   = THE VECTOR OF CATA ITEM VALUES
C           LOCAL  = THE VECTOR SPECIFYING THE LOCATION WITHIN THE NODE
C                    OF THE CATA ITEMS
C           JCTKEY = THE TRACE KEY
C           JCOOFS = SEE THE LOCATE OOCUMENTATION.  JCOOFS ABSOLUTE
C                    VALUES OF 1 AND 2 CAUSE JCPOFP TO BE FIRST NODE
C                    INSPECTED, WHILE 3 AND 4 CAUSE NEXT NOOE AFTER
C                    JCPOFP TO BE FIRST NODE INSPECTED
C           JCLEVL = SEE THE NEXT OOCUMENTATION
C           JCHOCL = THE ACCRESS OF THE TEMPORARY LIST HEAOCELL
C           JCCLNO = A USER ASSIGNED CALL STATEMENT ID. PARAMETER
C
C       REMARKS
C           THE NUMBER REPRESENTED BY JCTCIA MUST, FOR SCME MACHINES,
C           REPLACE THE 1 IN THE NAMED COMMON STATEMENT.
C           THE SEARCH MAY BE SPECIFIED AS EITHER FORWARO OR AS
C           BACKWARO ONLY.
C
C       SUBRCUTINE AND FUNCTION SUBPROGRAMS REQUIRED
C           J5COMP
C           NEXT
C           SYSENT
C           SYSEXT
C
C       ERRCR COCES FOR THIS PROGRAM
C           NONE
C
C       METHOD
C           SELF-EXPLANATORY
C
C                          . **********
C
      SUBROUTINE SEARCH(JCNTWK,JCPOFP,JCPOSP,LION,LOCAL,JCTKEY,JCOOFS,
     1 JCLEVL,JCHOCL,JCCLNO)
      COMMCN/ALLOC/JCCORE(1)
      DIMENSICN LION(1),LOCAL(1)
C                 ENTER
      CALL SYSENT(1,3,3,7,JCCLNO)
C                 SET TEMPORARY PARAMETERS
```

```
      JCPCSP=0
      MA=IABS(JCDOFS)
      MA=-(-1)**MA-(-1)**(MA/2+MA/4)
      MB=JCPOFP
C                     SPUR OF LARGEST POSSIBLE NODE
      M1=JCNTWK+1
      M1=JCCORE(M1)
      M2=M1+4
      M2=JCCORE(M2)
      M3=JCCORE(17)+8
      MC=M3+14*(M2/5)-2*(M2/6)
      MC=JCCORE(MC)
      IF(M2.NE.2)GO TO 1
      M2=M3+1
      M3=M3+6
      M2=JCCORE(M2)+JCCORE(M3)
      IF(M2.GT.MC)MC=M2
    1 MC=M1+5*(MC-1)
C                     GET NEXT NODE
    2 M1=MB
      MB=NEXT(JCNTWK,M1,JCTKEY,MA,JCLEVL,JCHOCL,LION,0,1)
      IF((MB.EQ.0).AND.(JCHOCL.EQ.0))GO TO 3
C                     TEST CONTENTS OF NODE
      CALL J5COMP(JCCORE(MC),MB,LION,LOCAL,JCDOFS,M1,1)
      IF(M1.EQ.0)GO TO 2
C                     SUCCESS
      JCPCSP=MB
      IF(JCDOFS.LT.0)JCDOFS=M1
C                     EXIT
    3 CALL SYSEXT
      RETURN
      END
```

```
C
C*****************************************************************
C                                  **********
C                                  * J7LINK *
C                                  **********
C
C
C        PURPOSE
C            J7LINK RETRIEVES A LINK FROM A SPECIFIED NODE.
C
C        USAGE
C            J7LINK(JCNTWK,JCPOFP,JCPOSP,JCWHDE,JCFORB,JCCLNO)
C
C        DATA FORMAT
C            N/A
C
C        DESCRIPTION OF PARAMETERS
C            JCFORB = -1 FOR BACKWARD OR +1 FOR FORWARD LINK
C            FOR ALL OTHER PASSED PARAMETERS SEE LNKBNT
C
C        REMARKS
C            THE NUMBER REPRESENTED BY JCTCIA MUST, FOR SOME MACHINES,
C            REPLACE THE 1 IN THE NAMED COMMON STATEMENT.
C
C        SUBROUTINE AND FUNCTION SUBPROGRAMS REQUIRED
C            FRMCEL
C            LNKFWD
C            SYSENT
C            SYSERR
C            SYSEXT
C
C        ERROR CODES FOR THIS PROGRAM
C            CODE FATAL  ERROR(DATA PROVIDED)
C            1    NF     JCWHDE TOO LARGE (NUMBER OF DIVERGER ENTRIES)
C
C        METHOD
C            SELF-EXPLANATORY
C                                  **********
C
      FUNCTION J7LINK(JCNTWK,JCPOFP,JCPOSP,JCWHDE,JCFORB,JCCLNO)
      COMMON/ALLOC/JCCORE(1)
      DIMENSION F(7),P(4),D(2)
      INTEGER F,P,D
      DATA F/2,3,1,1,1,1,1/,P/1,2,1,1/
C                 ENTER
      CALL SYSENT(1,3,3,8,JCCLNO)
C                 PREPARE SPUR ADDRESSES
      M1=JCNTWK+1
      M1=JCCORE(M1)
      M2=M1+4
      M2=JCCORE(17)+7+14*(JCCORE(M2)/5)-2*(JCCORE(M2)/6)
      MA=M1+5*(JCCORE(M2)-1)
C                 TEST JCPOFP
      F(1)=2
      F(5)=1
      IF(JCFORB.LT.0)F(5)=2
      CALL FRMCEL(JCCORE(MA),JCPOFP,D,F,1)
      JCPOSP=0
      IF(D(1).EQ.0)GO TO 7
C                 JCPOFP IS A NODE
      F(1)=1
      IF(D(2).EQ.0)GO TO 6
```

```
      CALL FRMCEL(JCCORE(MA),D(2),M1,F,2)
      IF(M1.NE.0)GO TO 6
C                     FIND DIVERGER ENTRY
    1 MB=1
    2 IF((MB.GE.JCWHDE).AND.(JCWHDE.GE.0))GO TO 3
      M1=D(2)
      D(2)=LNKFWD(JCCORE(MA),M1,1)
      MB=MB+1
      IF(D(2).GT.0)GO TO 2
      D(2)=M1
      IF(JCWHDE.LT.0)GO TO 4
C                     ERROR - NOT FOUND
      M2=MB-1
      CALL SYSERR(1,M2)
C                     FOUND
    3 JCPOSP=LNKFWD(JCCORE(MA),D(2),2)
    4 CALL FRMCEL(JCCORE(MA),D(2),J7LINK,P,3)
C                     EXIT
    5 CALL SYSEXT
      RETURN
C                     DIRECT NODE TO NODE
    6 IF(JCWHDE.GT.1)CALL SYSERR(2,JCWHDE)
      JCPOSP=-1
      J7LINK=D(2)
      GO TO 5
C                     JCPOFP IS DIVERGER
    7 D(2)=JCPOFP
      MB=0
      GO TO 2
      END
```

```
C
C*********************************************************************
C                                **********
C                                * DATIVE *
C                                **********
C
C       PURPOSE
C           CATIVE PERFORMS DECISION AND/OR TEAR VARIABLE FILE UPDATE
C           FROM THE GENDER DECISION AND TEAR VARIABLE LISTS.
C
C       USAGE
C           CALL DATIVE(JCGRUP,JCDTOB,JCCLNO)
C
C       DATA FORMAT
C           N/A
C
C       DESCRIPTION OF PARAMETERS
C           JCGRUP = THE ADDRESS OF A GROUP OR 0 IF THE ENTIRE GENDER
C                    LIST IS TO PARTICIPATE
C           JCDTOB = 0, 1 OR 2 AS RESPECTIVELY DECISION ONLY, TEAR ONLY
C                    OR BOTH DECISION AND TEAR VARIABLE FILES ARE TO BE
C                    UPDATED
C
C       REMARKS
C           THE NUMBER REPRESENTED BY JCTCIA MUST, FOR SOME MACHINES,
C           REPLACE THE 1 IN THE NAMED COMMON STATEMENT.
C
C       SUBROUTINE AND FUNCTION SUBPROGRAMS REQUIRED
C           FETCH
C           FRMCEL
C           LNKFWD
C           SEARCH
C           STORE
C           SYSENT
C           SYSEXT
C           TOCELL
C
C       ERROR CODES FOR THIS PROGRAM
C           NONE
C
C       METHOD
C           WHEN JCGRUP IS 0, THE ENTIRE SELECTED FILES ARE COMPLETELY
C           REWRITTEN.  WHEN JCGRUP REFERS TO A GROUP, THE SELECTED
C           FILES ARE EXPANDED BY THE GROUP'S DECISION AND/OR TEAR
C           VARIABLE LISTS.
C
C                                **********
C
      SUBROUTINE DATIVE(JCGRUP,JCDTOB,JCCLNO)
      COMMON/ALLOC/JCCORE(1)
      DIMENSION L(8),S(4),D(3),F(4),T(2),G(4),V(7),N(4)
      INTEGER S,D,F,T,G,V
      DATA S/1,1,1,3/,L/4,1,1,1,1,1,1,1/,F/1,3,1,1/,G/1,7,2,3/,V/2,21,1,
     1 1,2,1,1/,N/1,1,4,0/,MH/1/,MI/1/
C                 ENTER
      CALL SYSENT(1,3,4,1,JCCLNO)
C                 MA=ADDRESS OF SPUR FOR GROUP
      M1=JCCORE(16)+1
      M2=JCCORE(M1)
      M3=JCCORE(17)+8
      MA=M2+5*JCCORE(M3)-5
C                 M8=ADDRESS OF SPUR FOR DECISION AND TEAR LIST ENTRIES
```

```
      M3=M3+2
      MBA=JCCORE(M3)-1
      MB=M2+5*MBA
C                     MC=ADDRESS OF SPUR FOR SECEDE
      M2=JCCORE(15)+1
      MC=JCCORE(M2)
C                     MD=ADDRESS OF LOAD VECTOR 1
      M2=M2+1
      MD=JCCORE(M2)
      M3=MD+28
      JCCORE(M3)=-1
C                     ME=ADDRESS OF LOAD VECTOR 2
      ME=MD+30
      M3=ME+28
      JCCORE(M3)=-1
C                     MF=CURRENT GROUP
      MF=JCGRUP
      IF(MF.GT.0)GO TO 1
      M1=M1-1
      MF=JCCORE(M1)
C                     MG=KEY
      M1=M1+2
      MG=JCCORE(M1)+1
C                     MH AND MI ARE INDICATORS
C                     MJ=LEVEL
      MJ=0
C                     MK=TEMPORARY LIST HEADCELL
      MK=0
      GO TO 7
C                     FIND END OF FILE(S)
    1 IF(JCDTOB.EQ.2)GO TO 4
      M1=MD
    2 M2=1
      M3=1
    3 CALL FETCH(JCCORE(MC),L(M2),0,S,L,JCCORE(M1),M3,1)
      IF(M3.EQ.-1)GO TO 6
      M2=7
      L(8)=D(1)
      IF(D(1).GT.0)GO TO 3
      IF(M1.NE.MD)GO TO 5
      MH=5
      L(6)=3+2*D(3)
    4 IF(JCDTOB.EQ.1)GO TO 9
      M1=ME
      GO TO 2
    5 MI=7
      L(8)=3+2*D(3)
      GO TO 9
    6 IF(M1.EQ.MD)GO TO 4
      GO TO 9
C                     FIND A GROUP
    7 M1=1
    8 D(1)=1
      M2=MF
      CALL SEARCH(JCCORE(16),M2,MF,D,F,MG,1,MJ,MK,1)
      IF(MF.EQ.0)GO TO 15
      IF(M1.GT.1)GO TO 9
      M1=JCCORE(16)+2
      JCCORE(M1)=MG
C                     DECISION
    9 CALL FRMCEL(JCCORE(MA),MF,T,G,1)
      IF(JCDTOB.EQ.2)GO TO 13
```

```
      M2=MD
      M3=MH
      M4=1
C                    LIST
   10 IF(T(M4).EQ.0)GO TO 12
      CALL FRMCEL(JCCORE(MB),T(M4),D(2),V,2)
      D(1)=3+D(3)
      CALL STORE(JCCORE(MC),L(M3),D,S,V(5),JCCORE(M2),1)
      M3=5+2*(M4-1)
      M5=M3+1
      IF(D(3).EQ.0)GO TO 11
      M6=MBA+T(M4)
      L(M5)=0
      N(4)=D(1)
      CALL STORE(JCCORE(MC),L(M3),JCCORE(M6),N,V(5),JCCORE(M2),2)
   11 L(M5)=D(1)
      T(M4)=LNKFWD(JCCORE(MB),T(M4),1)
      GO TO 10
C                    TEAR
   12 IF(M4.EQ.1)MH=M3
      IF(M4.EQ.2)MI=M3
      IF((M4.EQ.2).OR.(JCDTOB.EQ.1))GO TO 14
   13 M2=ME
      M3=MI
      M4=2
      GO TO 10
C                    ITERATE
   14 IF(MF.NE.JCGRUP)GO TO 8
C                    ZERO LAST RELATIVE LINK
   15 IF(MH.EQ.1)GO TO 16
      M1=MD+4
      JCCORE(M1)=0
   16 IF(MI.EQ.1)GO TO 17
      M1=ME+4
      JCCORE(M1)=0
C                    EXIT
   17 CALL SYSEXT
      RETURN
      END
```

```
      SUBROUTINE JORTRN(NIBLG,NSPUR,IPTR,IOCALL)
      COMMON/ALLOC/JCCORE(1)
      DIMENSION NSPUR(1)
C             LOCAL VECTORS REQUIRED.
      DIMENSION LOCPTR(4),LOCTOP(7),NTOP(2),LOCNPE(4)
      DATA LOCPTR/1,7,1,1/
C             LOCTOP  2 ITEMS,TYPE FLAG, NOP
      DATA LOCTOP/2,9,2,2,5,1,1/
      DATA LOCNPE/1,5,2,2/
C
      CALL SYSENT(1,3,5,1,IOCALL)
      LN=IPTR
C             SET UP HEADCELL FOR PUSHDOWN LIST.
      LHC=0
   10 CALL FRMCEL( NSPUR,LN,NTOP,LOCTOP,2)
      NOP=NTOP(2)
      DO 30 I=1,NOP
      LNK=0
      CALL BREAK(NIBLG,LN,LNK,1,0,4)
      IF(NTOP(1).LE.1) GO TO 20
      CALL PUSH(NSPUR,LHC,LNK,LOCPTR,1,6)
      GO TO 30
   20 NWKEND=LNK
   30 CONTINUE
      CALL RETURN(NSPUR,LN,LN,8)
      IF(LHC.EQ.0) GO TO 40
      CALL POPUP(NSPUR,LHC,LHC,LN,LOCPTR,1,10)
      GO TO 10
   40 CALL FRMCEL(NSPUR,NWKEND,NPE,LOCNPE,12)
      IF(NPE.EQ.0) CALL RETURN(NSPUR,NWKEND,NWKEND,14)
      CALL SYSEXT
      RETURN
      END
```

```
      SUBROUTINE JOSVIF(NIBLG,NSPUR,ICNT,NIDVI,IDCALL)
      COMMON/ALLOC/JCCORE(1)
      DIMENSION NSPUR(1)
      DIMENSION IDOP(2)
C             LOCAL VECTORS REQUIRED
      DIMENSION LOCN1(7),LOCVIF(4),LOCNPE(4)
C             LOCN1  2 ITEMS, TYPE FLAG, NID
      DATA LOCN1/2,9,2,2,7,1,1/
      DATA LOCVIF/1,9,1,1/
      DATA LOCNPE/1,5,2,2/
C
      CALL SYSENT(1,3,5,2,IDCALL)
      ICNT=0
      NIBLG3=NIBLG+2
      JCCORE(NIBLG3)=JCCORE(NIBLG3)+1
      LHC=0
C             CLEAR ALL VIF FLAGS.
      LNK=JCCORE(NIBLG)
   10 NWKEND=LNK
      CALL TOCELL(NSPUR,LNK,0,LOCVIF,2)
      LNK =NEXT(NIBLG,LNK ,JCCORE(NIBLG3), 2,-1, LHC,0,0, 4)
      IF(LNK.GT.0) GO TO 10
   20 IF(NIDVI.EQ.0) GO TO 60
C             SET N TO THE NUMBER OF NODES ENTERING NWKEND.
      CALL FRMCEL(NSPUR,NWKEND,N,LOCNPE,6)
C             FOR EACH SIMPLE OPERAND, DETERMINE IF VARIABLE OF
C             INTEREST.  IF IT IS, TRACE BACKWARD THROUGH NETWORK
C             SETTING VIF FLAGS.
      DO 50 I=1,N
      LNK=LNKBNT(NIBLG,NWKEND,NU,I,8)
      CALL FRMCEL(NSPUR,LNK,IDOP,LOCN1,10)
C             CHECK IF OPERAND IS VARIABLE OF INTEREST.
      IF(IDOP(1).NE.1.OR.IDOP(2).NE.NIDVI) GO TO 50
C             VARIABLE OF INTEREST FOUND.
   30 CALL TOCELL(NSPUR,LNK,1,LOCVIF,12)
      IF(LNK.EQ.JCCORE(NIBLG)) GO TO 40
      LNK=LNKBNT(NIBLG,LNK,NU,1,14)
      GO TO 30
   40 ICNT=ICNT+1
   50 CONTINUE
   60 CALL SYSEXT
      RETURN
      END
```

```
      SUBROUTINE MATH(NIC,A,B,C,IOCALL)
      COMMON/ALLOC/JCCORE(1)
      CALL SYSENT(1,3,5,3,IOCALL)
      I=JCCORE(17)+1
      IF(JCCORE(I).NE.1) CALL SYSERR(2,JCCORE(I))
      IF(NIO.LT.1) GO TO 200
      J=NIO/100
      IF(J.GT.2) GO TO 200
      K=NIO-100*J
      IF(K.EQ.0) GO TO 200
      J=J+1
      GO TO(1000,2000,3000),J
   10 CALL SYSEXT
      RETURN
  200 C=0.0
      CALL SYSERR(1,NIO)
      GO TO 10
C             UNARY OPERATORS
 1000 IF(K.GT.10) GO TO 200
      GO TO(1010,1020,1030,1040,1050,1060,1070,1080,1090,1100),K
 1010 C=-A
      GO TO 10
 1020 C=SIN(A)
      GO TO 10
 1030 C=COS(A)
      GO TO 10
 1040 C=TAN(A)
      GO TO 10
 1050 C=ARSIN(A)
      GO TO 10
 1060 C=ARCOS(A)
      GO TO 10
 1070 C=ATAN(A)
      GO TO 10
 1080 C=ALOG(A)
      GO TO 10
 1090 C=EXP(A)
      GO TO 10
 1100 C=1./A
      GO TO 10
C
C               BINARY OPERATOR
 2000 IF(K.GT.4) GO TO 200
      GO TO(2010,2020,2030,2040),K
 2010 C=A
      GO TO 10
 2020 C=A-B
      GO TO 10
 2030 C=A/B
      GO TO 10
 2040 C=A**B
      GO TO 10
C
C               MULTIOPERAND OPERATORS
 3000 IF(K.GT.2) GO TO 200
      GO TO(3010,3020),K
 3010 C=A+B
      GO TO 10
 3020 C=A*B
      GO TO 10
      END
```

```
      FUNCTION MATHCP(NIO,A,B,IDCALL)
      COMMON/ALLOC/JCCORE(1)
      CALL SYSENT(1,3,5,4,IDCALL)
      I=JCCORE(17)+1
      IF(JCCORE(I).NE.1) CALL SYSERR(2,JCCORE(I))
      IF(NID.GT.307.OR.NID.LT.301) GO TO 200
      MATHCP=1
      K=NID-NID/100*100
      GO TO(10,20,30,40,50,60,70),K
   10 IF(A.LT.B) GO TO 90
      GO TO 100
   20 IF(A.LE.B) GO TO 90
      GO TO 100
   30 IF(A.EQ.B) GO TO 90
      GO TO 100
   40 IF(A.NE.B) GO TO 90
      GO TO 100
   50 IF(A.GE.B) GO TO 90
      GO TO 100
   60 IF(A.GT.B) GO TO 90
      GO TO 100
C
   70 IF(A-B) 80,90,100
   80 MATHCP=-1
      GO TO 100
   90 MATHCP=0
  100 CALL SYSEXT
      RETURN
C
  200 CALL SYSERR(1,NID)
      GO TO 100
      END
```

```
SUBROUTINE ROPA(A,B)
DOUBLE PRECISION A
B=A
RETURN
END
```

```
      SUBROUTINE JORLSE(NIBLG,LHC,IDCALL)
      COMMON/ALLOC/JCCORE(1)
      CALL SYSENT(1,3,5,5,IDCALL)
      IF(LHC.EQ.0) GO TO 10
C             SET L1 AND L2 TO SPUR BLOCK POINTERS NEEDED.
      L1=JCCORE(NIBLG+3)+5
      L2=L1+15
C             SET L3 TO POINTER TO AUXILLIARY LIST.
      L3=JCCORE(L1-2)+LHC
      L3=JCCORE(L3)
C             RETURN LHC AND AUXIALLARY LIST.
      CALL RETURN(JCCORE(L2),LHC,LHC,2)
      LHC=0
      L4=0
      IF(L3.GT.0) CALL RETURN(JCCORE(L1),L3,L4,4)
C
   10 CALL SYSEXT
      RETURN
      END
```

```
      FUNCTION JOCPND(ITEM1,ITEM2,IDCALL)
      DIMENSION ITEM1(1),ITEM2(1)
      CALL SYSENT(1,3,5,11,IDCALL)
      IF(ITEM1(1)-ITEM2(1)) 40,10,60
   10 IF(ITEM1(1).EQ.0) GO TO 30
      IF(ITEM1(2)-ITEM2(2)) 40,20,60
   20 IF(ITEM1(3)-ITEM2(3)) 40,50,60
C             CONSTANT
   30 IF(MATHCP(307,ITEM1(4),ITEM2(4),2)) 40,50,60
   40 JOCPND=-1
      GO TO 70
   50 JOCPND=0
      GO TO 70
   60 JOCPND=1
   70 CALL SYSEXT
      RETURN
      END
```

```
      SUBROUTINE JOMTCH(NIBLG,NSPUR,NODE,LOW,N,LIST,LION,LNKIX,LNKPT,
     1                  IOCALL)
      COMMON/ALLOC/JCCORE(1)
      DIMENSION LIST(1),LION(1)
      DIMENSION NSPUR(1)
      DIMENSION NOAT(6)
C             LOCAL VECTORS REQUIRED.
C             LOCALL  5 ITEMS, VIF, TYPE FLAG, NOP, NID, VALUE.
      DIMENSION LOCALL(13)
      DATA LOCALL/4,9,1,2,5,1,1,7,1,1,8,1,0/
C
      CALL SYSENT(1,3,5,12,IOCALL)
      L=LOW+1
      LNKIX=0
      LNKPT=0
      IF(L.GT.N) GO TO 30
      I=JCCORE(17)+1
      LOCALL(13)=JCCORE(I)
      DO 20 I=L,N
      LNK=LNKFNT(NIBLG,NODE,NU,I,2)
      CALL FRMCEL(NSPUR,LNK,NOAT,LOCALL,4)
      DO 10 J=1,4
      IF(LIST(J).EQ.1.AND.LION(J).NE.NDAT(J)) GO TO 20
   10 CONTINUE
      IF(LIST(5).EQ.1.AND.MATHCP(303,LION(5),NDAT(5),6).NE.0) GO TO 20
C             MATCH IS FOUND.
      LNKIX=I
      LNKPT=LNK
      GO TO 30
   20 CONTINUE
C             NO MATCH IS FOUND.
   30 CALL SYSEXT
      WRITE(6,900) NODE,LOW,N,LNKIX,LNKPT,(LIST(I),LION(I),I=1,6)
  900 FORMAT(1X,'JOMTCH-NODE,LOW,N,LNKIX,LNKPT,LIST,LION',5I10,/,12I10)
      JCCORE(13)=6
      RETURN
      END
```

```
      SUBROUTINE JONTPS(NIBLG,NSPUR,PSPUR,IPS,NWKEND,IDCALL)
      COMMON/ALLOC/JCCORE(1)
      INTEGER PSPUR,PSPURC,PSPURV,PSPURO
      DIMENSION NSPUR(1)
      DIMENSION NODE(5),LOCN(13),LOCPS(10),LOCC(7),LOCDIM(4),LOCOPR(10)
      DATA LOCN/4,9,2,2,7,1,1,5,1,1,8,1,0/
      DATA LOCC/2,3,1,1,23,1,0/
      DATA LOCDIM/1,22,1,1/
      DATA LOCOPR/3,3,1,1,2,1,1,21,1,1/
      DATA LOCPS/3,1,1,1,3,1,1,22,1,1/
C
      CALL SYSENT(1,3,5,13,IDCALL)
      I=JCCORE(17)+1
      LOCN(13)=JCCORE(I)
      LOCC(7)=LOCN(13)
C             POINTER TO SPUR BLOCKS CALCULATIONS
C               PSPURC        CONSTANTS (LOCAL)
C               PSPURO        OPERATORS
C               PSPURV        VARIABLES (NO INDICES)
      I=I-2
      PSPURV=(JCCORE(I+12)-1)*5+PSPUR
      PSPURC=(JCCORE(I+8)+JCCORE(I+2)-1)*5+PSPUR
      PSPURO=(JCCORE(I+11)-1)*5+PSPUR
C             TRACE NETWORK BACKWARDS USING THE VERB NEXT.  THE ORDER
C             OF SUCH A TRACE IS IN A REVERSE POLISH ORDER.
      LHC=0
      LNK=JCCORE(NIBLG)
      NIBLG3=NIBLG+2
      JCCORE(NIBLG3)=JCCORE(NIBLG3)+1
      JPS=0
   10 CALL FRMCEL( NSPUR,LNK,NODE,LOCN,2)
      JTYPE=NODE(1)+1
      GO TO(20,30,40),JTYPE
C             CONSTANT FOUND.
   20 NODE(3)=5
      CALL NEWCEL(JCCORE(PSPURC),NC,4)
      CALL TOCELL(JCCORE(PSPURC),NC,NODE(3),LOCC,4)
      GO TO 50
C             VARIABLE FOUND.
   30 CALL FRMCEL( JCCORE(PSPURV),NODE(2),NDIM,LOCDIM,6)
      ISPT=NDIM*5+PSPURV
      CALL NEWCEL( JCCORE(ISPT),NC,8)
      CALL COPY(JCCORE(ISPT),NODE(2),NC,0,10)
      GO TO 50
C             OPERATOR FOUND.
   40 NODE(1)=0
      CALL NEWCEL(JCCORE(PSPURO),NC,14)
      CALL TOCELL(JCCORE(PSPURO),NC,NODE,LOCOPR,14)
   50 CALL FSTLNK(JCCORE(PSPURV),NC,JPS,12)
      JPS=NC
C             GO TO NEXT ENTRY IN NETWORK.
      LNK =NEXT(NIBLG,LNK ,JCCORE(NIBLG3), 2,-1, LHC,0,0,16)
      IF(LNK.GT.0.AND.LNK.NE.NWKEND) GO TO 10
      CALL JORLSE(NIBLG,LHC,17)
C             RELEASE NETWORK
      CALL JORTRN(NIBLG,NSPUR,JCCORE(NIBLG),18)
C             RELEASE OLD POLISH STRING.
   60 CALL FRMCEL(JCCORE(PSPURV),IPS,NODE,LOCPS,20)
      J=NODE(2)+1
      GO TO(70,80,70,80,80,90),J
C             OPERATOR OR MODULE CONSTANT
   70 K=PSPURO
```

```
      GO TO 100
C                VARIABLE
   80 K=5*NODE(3)+PSPURV
      GO TO 100
C                LOCAL CONSTANT
   90 K=PSPURC
  100 CALL RETURN(JCCORE(K),IPS,IPS,22)
      IPS=NODE(1)
      IF(IPS.GT.0) GO TO 60
      IPS=JPS
      CALL SYSEXT
      RETURN
      END
```

```
      SUBROUTINE JOPSTN(NIBLG,NSPUR,PSPUR,IPS,NWKEND,NIDVI,IOCALL)
      COMMON/ALLOC/JCCORE(1)
      DIMENSION NSPUR(1)
      INTEGER PSPUR,PSPURV
      DIMENSION LION(6),NCHK(2),LION1(3)
C             LOCAL VECTORS REQUIRED
      DIMENSION LOCN(13),LOCP(16),LCLPTR(4)
      DIMENSION LOCNPE(4),LOCN2(7),LOCP1(10)
      DATA LOCN/4,9,1,2,7,1,1,3,1,1,8,1,0/
      DATA LOCP/5,21,1,1,3,1,1,2,1,1,22,1,1,23,1,0/
      DATA LCLPTR/1,7,1,1/
      DATA LOCNPE/1,5,2,2/
      DATA LOCN2/2,9,2,2,7,1,1/
      DATA LOCP1/3,3,1,1,2,1,1,22,1,1/
      DIMENSION NCST(6),NEQ(6)
      DATA NCST/0,0,0,1,0,0/
      DATA NEQ/0,2,101,1,0,0/
C
      CALL SYSENT(1,3,5,14,IOCALL)
      I=JCCORE(17)-1
      PSPURV=(JCCORE(I+12)-1)*5+PSPUR
      ISIZ=JCCORE(PSPURV+3)
      I=JCCORE(I+2)
      LOCN(13)=I
      LOCP(16)=I
C             OBTAIN END CELL FOR NETWORK
      CALL NEWCEL(NSPUR,NWKEND,2)
C             SET FLAG TO INDICATE NETWORK NODE
      CALL TOCELL(NSPUR,NWKEND,1,LOCN(7),3)
      N1=0
      LNK=IPS
C             OBTAIN CONTENTS OF NEXT ENTRY ON POLISH STRING.
   10 CALL FRMCEL(JCCORE(PSPURV),LNK,LION,LOCP,4)
      NOP=LION(1)
      JTYPE=LION(2)+1
      NID=LION(3)
      NOIM=LION(4)
      CALL NEWCEL(NSPUR,N2,6)
      GO TO(20,40,100,40,120,110),JTYPE
C             OPERATOR
   20 LION(2)=2
      I=1
   30 CALL POPUP(NSPUR,N1,N1,N3,LCLPTR,1,8)
      CALL COUPLE(NIBLG,N2,N3,-1,0,0,10)
      I=I+1
      IF(I.LE.NOP) GO TO 30
      GO TO 140
C             VARIABLE FOUND
   40 CONTINUE
C                CHECK IF THIS VARIABLE HAS OCCURRED BEFORE IN NETWORK.
      CALL FRMCEL( NSPUR,NWKEND,NPE,LOCNPE,12)
      IF(NPE.EQ.0) GO TO 85
      DO 80 I=1,NPE
      LCHK=LNKBNT(NIBLG,NWKEND,NU,I,14)
      CALL FRMCEL(NSPUR,LCHK,NCHK,LOCN2,16)
      IF(NCHK(1).NE.1) GO TO 80
C             VARIABLE FOUND.  COMPARE WITH CURRENT VARIABLE TO SEE IF
C             SAME.
      CALL FRMCEL(JCCORE(PSPURV),NCHK(2),LION1,LOCP1,18)
C             COMPARE TYPE,NID,AND DIMENSIONALITY.
      DO 50 J=1,3
      IF(LION(J+1).NE.LION1(J)) GO TO 80
```

```
   50 CONTINUE
C                COMPARE INDICES FOR VARIABLES
      IF(NDIM.EQ.0) GO TO 70
      J1=NCHK(2)+ISIZ
      J2=LNK+ISIZ
      DO 60 J=1,NDIM
      IF(JCCORE(J1).NE.JCCORE(J2)) GO TO 80
      J1=J1+1
   60 J2=J2+1
C                VARIABLES ARE THE SAME
   70 LION(3)=NCHK(2)
      GO TO 90
   80 CONTINUE
   85 LION(3)=LNK
   90 IF(JTYPE.EQ.4) NIOVI=LION(3)
      LION(2)=1
      GO TO 130
C                MODULE CONSTANT FOUND.
  100 CALL MATH(101,JCCORE(NID),NU,LION(5),20)
C                LOCAL CONSTANT FOUND.
  110 LION(2)=0
      GO TO 130
C                FUNCTION FOUND (IF FOUND, IT IS THE VARIABLE OF INTEREST
C                ALSO).
  120 LION(3)=LNK
      LION(2)=1
      NIOVI=LNK
C
  130 CALL COUPLE(NIBLG,N2,NWKEND,0,0,0,22)
C                STORE ITEM DATA IN NETWORK NODE.
  140 LION(1)=0
      LION(4)=1
      CALL TOCELL(NSPUR,N2,LION,LOCN,24)
C                OBTAIN NEXT ITEM ON POLISH STRING.
      LNK=LNKFWD(JCCORE(PSPURV),LNK,26)
      IF(LNK.LE.0) GO TO 150
      CALL PUSH(NSPUR,N1,N2,LCLPTR,1,28)
      GO TO 10
C                CHECK IF LAST ELEMENT WAS =.
  150 IF(LION(2).EQ.2.AND.LION(3).EQ.101) GO TO 160
C                ATTACH ZERO,EQUAL TO NETWORK.
      CALL NEWCEL(NSPUR,NZERO,30)
      CALL RDPA(0.D0,NCST(5))
      CALL TOCELL(NSPUR,NZERO,NCST,LOCN,32)
      CALL NEWCEL(NSPUR,NEQUAL,34)
      CALL TOCELL(NSPUR,NEQUAL,NEQ,LOCN,36)
      CALL COUPLE(NIBLG,NZERO,NWKEND,0,0,0,38)
      CALL COUPLE(NIBLG,NEQUAL,N2,0,0,0,40)
      CALL COUPLE(NIBLG,NEQUAL,NZERO,0,0,0,42)
      N2=NEQUAL
C                EXIT ROUTINE
  160 JCCORE(NIBLG)=N2
      CALL SYSEXT
      RETURN
      END
```

```
      FUNCTION JOCPCD(NIBLG,NSPUR,IOP1,IOP2,IOCALL)
      COMMON/ALLOC/JCCORE(1)
      DIMENSION NSPUR(1)
      DIMENSION LST1(5),LST2(5)
C             LOCAL VECTORS REQUIRED.
C             LOCN1  4 ITEMS, TYPE FLAG,NID,NUMBER OF OPERANDS,
C                    VALUE OF CONSTANT.
      DIMENSION LOCN1(13)
      DATA LOCN1/4,9,2,2,7,1,1,5,1,1,8,1,0/
C
      CALL SYSENT(1,3,5,21,IOCALL)
      NIBLG3=NIBLG+2
      JCCORE(NIBLG3)=JCCORE(NIBLG3)+1
      I=JCCORE(17)+1
      LOCN1(13)=JCCORE(I)
      LH1=0
      LH2=0
      JOP1=IOP1
      JOP2=IOP2
C             COMPARE NODES.
   10 CALL FRMCEL(NSPUR,JOP1,LST1,LOCN1,2)
      CALL FRMCEL(NSPUR,JOP2,LST2,LOCN1,4)
      J=JOCPND(LST1,LST2,6)
      IF(J.NE.0) GO TO 20
C             BOTH NODES EQUAL.  GO TO NEXT NODE PAIR.
      JOP1=NEXT(NIBLG,JOP1,JCCORE(NIBLG3), 2,-1, LH1,0,0, 8)
      JOP2=NEXT(NIBLG,JOP2,JCCORE(NIBLG3), 2,-1, LH2,0,0,10)
C             CHECK IF EITHER OPERAND HAS TERMINATED.  IF NOT COMPARE
C             NEXT NODE PAIR. (NOTE - IF EITHER OPERAND TERMINATES,
C             BOTH MUST TERMINATE.)
      IF(JOP1.NE.0) GO TO 10
   20 JOCPCD=J
C             RETURN LISTS LH1 AND LH2.
      CALL JORLSE(NIBLG,LH1,12)
      CALL JORLSE(NIBLG,LH2,14)
      CALL SYSEXT
      RETURN
      END
```

```
      SUBROUTINE JOORCR(NIBLG,NSPUR,IOPR,IDCALL)
      COMMON/ALLOC/JCCORE(1)
      DIMENSION NSPUR(1)
      DIMENSION NODE1(3)
C             LOCAL VECTORS REQUIRED.
C             LOCN1  3 ITEMS, TYPE,NOP, NIO.
      DIMENSION LOCTYP(4),LOCN1(10)
      DATA LOCTYP/1,9,2,2/
      DATA LOCN1/3,9,2,2,5,1,1,7,1,1/
C
      CALL SYSENT(1,3,5,31,IDCALL)
      LHO=0
C             TRACE THE OPERAND FORWARD TO ALL OF ITS TERMINAL NODES.
C             PLACE EACH ONE FOUND AT THE TOP OF THE CONVERGER FOR THE
C             END NODE OF THE NETWORK, NWKEND.
      NIBLG3=NIBLG+2
      JCCORE(NIBLG3)=JCCORE(NIBLG3)+1
      LNK=IOPR
      NWKEND=0
C             CHECK NODE TO SEE IF IT IS A SIMPLE OPERAND.
   10 CALL FRMCEL(NSPUR,LNK,ITYPE,LOCTYP,2)
      IF(ITYPE.GT.1) GO TO 20
C             SIMPLE OPERAND
      NWKEND=0
      CALL BREAK(NIBLG,LNK,NWKEND,1,0,4)
      CALL COUPLE(NIBLG,LNK,NWKEND,0,0,0,6)
C
   20 LNK =NEXT(NIBLG, LNK,JCCORE(NIBLG3), 2,-1, LHO,0,0, 8)
      IF(LNK.NE.0.AND.LNK.NE.NWKEND) GO TO 10
      CALL JORLSE(NIBLG,LHO,11)
C             TRACE BACKWARDS.  AT EACH NODE REACHED, ORDER THE NODES
C             OF THAT OPERAND USING ALGORITHM 1B.
      JCCORE(NIBLG3)=JCCORE(NIBLG3)+1
      KEY=JCCORE(NIBLG3)
      LNK=NWKEND
   30 LNK=NEXT(NIBLG,LNK,KEY,-2,-1,LHO,0,0,10)
C             ALGORITHM 1B.
C             SET N TO THE NUMBER OF OPERANDS TO NODE LNK.
      CALL FRMCEL(NSPUR,LNK,NODE1,LOCN1,12)
      N=NODE1(2)
      IF((NODE1(1).EQ.2.AND.NODE1(3)/100.LT.2).OR.N.EQ.1) GO TO 70
   40 IFLAG=0
C             PICK UP OPERAND 1 OF LNK.
      IOP1=LNKFNT(NIBLG,LNK,NU,1,14)
      DO 60 I=2,N
C             PICK UP OPERAND I FOR NODE LNK.
      IOP2=LNKFNT(NIBLG,LNK,NU,I,16)
C             COMPARE OPERANDS.
      IF(JOCPCD(NIBLG,NSPUR,IOP1,IOP2,18).LE.0) GO TO 50
C             OPERAND I LESS THAN OPERAND I-1.  REVERSE THEIR ORDER IN
C             FORWARD DIVERGER OF OPERAND BEING ORDERED.
      IOP2=0
      CALL BREAK(NIBLG,LNK,IOP2,I,0,20)
      CALL COUPLE(NIBLG,LNK,IOP2,I-1,0,0,22)
      IFLAG=1
      GO TO 60
C             DEFINE OPERAND 1 AS CURRENT OPERAND 2
   50 IOP1=IOP2
   60 CONTINUE
C             TEST IF ORDERING PROCESS IS DONE.  IF NOT START AGAIN.
      IF(N.GT.2.AND.IFLAG.NE.0) GO TO 40
C             END OF ALGORITHM 1B.
```

```
C              CHECK NODE LNK.  IF IT IS NOT OPERAND BEING ORDERED,
C              REPEAT FROM LABEL 30.
   70 IF(LNK.NE.IOPR) GO TO 30
      CALL JORLSE(NIBLG,LHO,24)
      CALL SYSEXT
      RETURN
      END
```

```
      SUBROUTINE JORULS(NIBLG,NSPUR,NWKEND,NIDVI,NRUL,IDCALL)
      COMMON/ALLOC/JCCORE(1)
      DIMENSION NSPUR(1)
      DIMENSION LL(2,3)
      DIMENSION NODE1(6),NODE2(6),NODE3(6),NODE4(6)
      INTEGER RHS,RHSL,RHSR,P1,P2,EQUAL,VIF
      LOGICAL LEX,REX,LMUL,RMUL
C             LOCAL VECTORS REQUIRED
      DIMENSION LOCALL(13),LOCNID(4),LOCVIF(4)
      DATA LOCALL/4,9,1,2,5,1,1,7,1,1,8,1,0/
      DATA LOCNID/1,7,1,1/
      DATA LOCVIF/1,9,1,1/
C             RHS NID TABLE FOR LIST 1/COMP,ARCSIN,ARCCOS,ARCTAN,SIN,
C                                      COS,TAN,EXP,ALOG,RECIP/
      DIMENSION L1RHS(10)
      DATA NL1/10/,L1RHS/1,5,6,7,2,3,4,9,8,10/
C             DATA VECTOR FOR SEARCHING FOR VIF FLAG USING JOMTCH.
      DIMENSION LIST(5)
      DATA LIST/1,0,0,0,0/
C             DATA VECTORS FOR SEARCHING FOR EXP,**, AND *N
      DIMENSION LIST1(5),LION1(4),LION2(4),LION3(4)
      DATA LIST1/1,1,0,1,0/,LION1/1,2,0,9/,LION2/1,2,0,104/
      DATA LION3/1,2,0,202/
C             DATA VECTOR TO CONSTRUCT A NEW OPERATOR NODE
C             LOCN3   3 ITEMS, TYPE FLAG,SEARCH KEY,NID
      DIMENSION LOCN3(10),NOPTV(3)
      DATA LOCN3/3,9,2,2,3,1,1,7,1,1/
      DATA NOPTV/2,1,0/
C             DATA VECTOR TO CONSTRUCT A NEW CONSTANT NODE
C             LOCN4  3 ITEMS, TYPE FLAG, SEARCH KEY, VALUE
      DIMENSION LOCN4(10),NCST(4)
      DATA LOCN4/3,9,2,2,3,1,1,8,1,0/
      DATA NCST/0,1,0,0/
C
      CALL SYSENT(I,3,5,41,IDCALL)
 9999 CONTINUE
      I=JCCORE(17)+1
      I=JCCORE(I)
      LOCN4(10)=I
      LOCALL(13)=I
      NRUL=-1
C             SET VIF FLAGS FOR NETWORK
      CALL JOSVIF(NIBLG,NSPUR,ICNT,NIDVI,1)
      IF(ICNT.EQ.0) GO TO 50
C             OBTAIN RIGHT AND LEFT HAND SIDE OPERANDS OF EQUAL SIGN.
      EQUAL=JCCORE(NIBLG)
      LHS=LNKFNT(NIBLG,EQUAL,NU,1,2)
      RHS=LNKFNT(NIBLG,EQUAL,NU,2,2)
      CALL FRMCEL(NSPUR,LHS,NODE1,LOCALL,4)
      CALL FRMCEL(NSPUR,RHS,NODE2,LOCALL,5)
C             DETERMINE WHICH OPERAND OF EQUAL HAS VIF SET.
      IF(NODE1(1).NE.0) GO TO 30
C             LHS IS Q1, RHS CONTAINS VARIABLE OF INTEREST.
C             CHECK IF RHS IS AN OPERAND (THE VARIABLE OF INTEREST ONLY)
      IF(NODE2(2).EQ.1) GO TO 700
C             RHS IS AN OPERATOR
      J=NODE2(4)/100+1
      GO TO (100,20,200),J
C             RHS IS A BINARY OPERATOR.
C             RETRIEVE RIGHT AND LEFT HAND SIDE OPERANDS FOR IT.
   20 RHSL=LNKFNT(NIBLG,RHS,NU,1,3)
      RHSR=LNKFNT(NIBLG,RHS,NU,2,4)
```

```
      CALL FRMCEL(NSPUR,RHSL,NODE3,LOCALL,8)
      CALL FRMCEL(NSPUR,RHSR,NODE4,LOCALL,9)
      J=NODE2(4)-101+3*NODE3(1)+6*NODE4(1)
C           NEITHER        LHSVIF         RHSVIF            BOTH
C        -    1              4              7                10
C        /    2              5              8                11
C       **    3              6              9                12
      GO TO(50,50,50,400,400,600,300,300,500,2900,2100,40),J
C             LHS CONTAINS VARIABLE OF INTEREST.  CHECK RHS.
   30 IF(NODE2(1).EQ.1) GO TO 900
      GO TO 800
C             USE THIS EXIT IF RULES U NEEDED BUT THEY FAIL.
   40 NRUL=0
   50 CALL SYSEXT
      RETURN
C             ***      **** RULES S SORTING DONE ***     ***
C
C     RULE  S-1
C
C        =,Q1,L1(COMP'SIN'COS'TAN'ARCSIN'ARCCOS'ARCTAN'ALOG'EXP'RECIP)
C        ,P1      BECOMES
C
C        =,(L1(COMP'ARCSIN'ARCCOS'ARCTAN'SIN'COS'TAN'EXP'ALOG'RECIP),
C        Q1),P1
C
C             =                EQUAL
C             Q1               LHS,NODE1
C             OP               RHS,NODE2
C             P1               WILL BE IN P1
  100 P1=0
      CALL BREAK(NIBLG,RHS,P1,1,0,10)
      LHS=0
      CALL BREAK(NIBLG,EQUAL,LHS,1,0,11)
      CALL COUPLE(NIBLG,EQUAL,P1,2,1,0,12)
      CALL COUPLE(NIBLG,RHS,LHS,0,0,0,13)
C             REPLACE UNARY OPERATOR WITH NEW UNARY OPERATOR.
      I=NODE2(4)
      CALL TOCELL(NSPUR,RHS,L1RHS(I),LOCNID,14)
      NRUL=1
      GO TO 50
C
C     RULE  S-2
C
C        =,Q1,L1(*N'+N),LO1(N,ANY=P1)  BECOMES
C
C        =,L1(/'-),L1(*N-1'+N-1),LO1(N-1,DELETE P1),Q1,P1
C
C             =                EQUAL
C             Q1               LHS,NODE1
C             OP               RHS,NODE2
C             P1               WILL BE IN RHSL
  200 NRUL=2
      CALL JOMTCH(NIBLG,NSPUR,RHS,0,NODE2(3),LIST,1,IL,RHSL,15)
C             CHECK IF MORE THAN 1 OPERAND CONTAINS VARIABLE OF INTEREST
      CALL JOMTCH(NIBLG,NSPUR,RHS,IL,NODE2(3),LIST,1,IR,RHSR,16)
      IF(IR.NE.0) GO TO 2200
      P1=0
      CALL BREAK(NIBLG,RHS,P1,IL,0,17)
      LHS=0
      CALL BREAK(NIBLG,EQUAL,LHS,1,1,18)
      RHS=0
      CALL BREAK(NIBLG,EQUAL,RHS,1,1,19)
```

```
C             ATTACH P1 TO EQUAL NODE
      CALL COUPLE(NIBLG,EQUAL,P1,0,0,1,20)
C             SET UP A NEW NODE TO HOLD LHS TOP OPERATOR FOR EQUAL.
      CALL NEWCEL(NSPUR,NLHS,21)
      NOPTV(3)=103
      IF(NODE2(4).EQ.201) NOPTV(3)=102
      CALL TOCELL(NSPUR,NLHS,NOPTV,LOCN3,22)
C             ATTACH NEW LHS TO EQUAL.
      CALL COUPLE(NIBLG,EQUAL,NLHS,1,1,0,23)
C             ATTACH Q1 AS RHS AND MULTIOPERATOR AS LHS OPERAND OF NEW
C             LHS.
      CALL COUPLE(NIBLG,NLHS,LHS,1,1,1,24)
      CALL COUPLE(NIBLG,NLHS,RHS,1,1,0,25)
      GO TO 50
C
C     RULE  S-3
C
C        =,Q1,L1(/*-),Q2,P1   BECOMES
C
C        =,(L1(*2'+2),Q2,Q1),P1
C
C             =               EQUAL
C             Q1              LHS,NODE1
C             (/*-)           RHS,NODE2
C             Q2              RHSL,NODE3
C             P1              RHSR,NODE4,THEN P1
  300 NRUL=3
      LHS=0
      CALL BREAK(NIBLG,EQUAL,LHS,1,1,26)
      P1=0
      CALL BREAK(NIBLG,RHS,P1,2,1,27)
C             ALTER OPERATOR
      I=NODE2(4)+99
      CALL TOCELL(NSPUR,RHS,I,LOCNIO,28)
C             COUPLE Q1,OPERATOR AND P1,EQUAL
      CALL COUPLE(NIBLG,RHS,LHS,2,1,0,29)
      CALL COUPLE(NIBLG,EQUAL,P1,2,1,0,30)
      GO TO 50
C
C     RULE  S-4
C
C        =,Q1,L1(/*-),P1,Q2   BECOMES
C
C        =,(L1(/*-),Q1,Q2),P1
C
C             =               EQUAL
C             Q1              LHS,NODE1
C             (/*-)           RHS,NODE2
C             P1              RHSL,NODE3
C             Q2              RHSR,NODE4
  400 NRUL=4
      LHS=0
      CALL BREAK(NIBLG,EQUAL,LHS,1,1,31)
      P1=0
      CALL BREAK(NIBLG,RHS,P1,1,1,32)
      CALL COUPLE(NIBLG,RHS,LHS,1,1,0,33)
      CALL COUPLE(NIBLG,EQUAL,P1,2,1,0,34)
      GO TO 50
C
C     RULE  S-5
C
C        =,Q1,**,Q2,P1    BECOMES
```

```
C
C       =,(**,RECIP,Q2,Q1),P1
C
C             =                EQUAL
C             Q1               LHS,NODE1
C             **               RHS,NODE2
C             Q2               RHSL,NODE3
C             P1               RHSR,NODE4
  500 NRUL=5
      LHS=0
      CALL BREAK(NIBLG,EQUAL,LHS,1,1,35)
      P1=0
      CALL BREAK(NIBLG,RHS,P1,2,0,36)
      RHSL=0
      CALL BREAK(NIBLG,RHS,RHSL, 1,0,37)
C           OBTAIN A NEWCELL TO PUT RECIP OPERATOR INTO.
      CALL NEWCEL(NSPUR,NOPR,37)
      NOPTV(3)=10
      CALL TOCELL(NSPUR,NOPR,NOPTV,LOCN3,38)
C           COUPLE **,COMP AND COMP,Q1 AND EQUAL,P1
      CALL COUPLE(NIBLG,RHS,LHS,0,0,0,38)
      CALL COUPLE(NIBLG,RHS,NOPR,0,0,0,39)
      CALL COUPLE(NIBLG,NOPR,RHSL,0,0,0,40)
      CALL COUPLE(NIBLG,EQUAL,P1,2,0,0,41)
      GO TO 50
C
C     RULE  S-6
C
C       =,Q1,**,P1,Q2  BECOMES
C
C       =,(/,ALOG,Q2,ALOG,Q1),P1
C
C             =                EQUAL
C             Q1               LHS,NODE1
C             **               RHS,NODE2
C             P1               RHSL,NODE3,THEN P1
C             Q2               RHSR,NODE4
  600 NRUL=6
      LHS=0
      CALL BREAK(NIBLG,EQUAL,LHS,1,1,42)
      P1=0
      CALL BREAK(NIBLG,RHS,P1,1,0,43)
      RHSR=0
      CALL BREAK(NIBLG,RHS,RHSR,1,0,44)
C           OBTAIN 2 NEWCELLS AND STORE ALOG OPERATORS IN BOTH.
      NOPTV(3)=8
      DO 610 I=1,2
      NOPL=NOPR
      CALL NEWCEL(NSPUR,NOPR,45)
  610 CALL TOCELL(NSPUR,NOPR,NOPTV,LOCN3,45)
C           CHANGE ** TO /
      CALL TOCELL(NSPUR,RHS,103,LOCNID,46)
C           COUPLE NOPL,Q2 AND NOPR,Q1 AND /,NOPR AND /,NOPL AND =,P1
      CALL COUPLE(NIBLG,NOPL,RHSR,0,0,0,47)
      CALL COUPLE(NIBLG,NOPR,LHS,0,0,0,48)
      CALL COUPLE(NIBLG,RHS,NOPR,0,1,0,49)
      CALL COUPLE(NIBLG,RHS,NOPL,0,0,0,50)
      CALL COUPLE(NIBLG,EQUAL,P1,2,0,0,51)
      GO TO 50
C
C     RULE  S-7
C
```

```
C        =,Q1,X   BECOMES
C
C        EXIT
  700 NRUL=7
      GO TO 50
C
C     RULE  S-8
C
C        =,P1,Q1   BECOMES
C
C        =,Q1,P1
C
C             =                EQUAL
C             P1               LHS,NODE1
C             Q1               RHS,NODE2
  800 NRUL=8
      RHS=0
      CALL BREAK(NIBLG,EQUAL,RHS,2,1,52)
      CALL COUPLE(NIBLG,EQUAL,RHS,1,1,0,53)
      GO TO 50
C
C     RULE  S-9
C
C        =,P1,P2   BECOMES
C
C        =,0.0,-,P1,P2
C
C             =                EQUAL
C             P1               LHS,NODE1
C             P2               RHS,NODE2
  900 NRUL=9
C             BREAK P1 AND P2 LINKS TO EQUAL.
      DO 910 I=1,2
      NU=0
  910 CALL BREAK(NIBLG,EQUAL,NU,1,0,54)
C             OBTAIN A NEWCELL AND FILL IT WITH THE CONSTANT ZERO.
      CALL RDPA(0.000,NCST(3))
      CALL NEWCEL( NSPUR,NCL,55)
      CALL TOCELL(NSPUR,NCL,NCST,LOCN4,56)
C             OBTAIN NEWCELL AND FILL IT WITH BINARY -.
      CALL NEWCEL(NSPUR,NCL2,57)
      NOPTV(3)=102
      CALL TOCELL(NSPUR,NCL2,NOPTV,LOCN3,57)
C             COUPLE EQUAL TO MINUS AND ZERO.  COUPLE - TO P2 AND P1.
      CALL COUPLE(NIBLG,EQUAL,NCL2,0,0,1,58)
      CALL COUPLE(NIBLG,EQUAL,NCL,0,0,0,59)
      CALL COUPLE(NIBLG,NCL2,RHS,0,0,1,60)
      CALL COUPLE(NIBLG,NCL2,LHS,0,0,0,61)
      GO TO 50
C
C     ***************************************************************
C
C     RULE  U-1 (MAYBE)
C
C        /,L1(EXP,P1'*N,LO1(N,ANY=(EXP,P1)),L2(EXP,P2'*M,LO2(M,
C        ANY=(EXP,P2))   BECOMES
C
C        L1(NULL'/,*N-1,LO1(N-1,DELETE(EXP,P1))),L2(EXP,-,P1,P2'
C        *M,LO2(M,DELETE(EXP,P2),ADD(EXP,-,P1,P2))
C
C             /                RHS,NODE2
C             L1               RHSL,NODE3
```

```
C              L2                  RHSR,NODE4
 2100 IF(NODE3(2).NE.2.OR.NODE4(2).NE.2) GO TO 40
      LEX=NODE3(4).EQ.9
      REX=NODE4(4).EQ.9
      LMUL=NODE3(4).EQ.202
      RMUL=NODE4(4).EQ.202
      IF(.NOT.((LEX.OR.LMUL).AND.(REX.OR.RMUL))) GO TO 2300
      L1PTR=RHSL
      IF(LEX) GO TO 2110
C             LHS IS *N
      CALL JOMTCH(NIBLG,NSPUR,RHSL,0,NODE3(3),LIST1,LION1,L1,L1PTR,62)
      IF(L1.EQ.0) GO TO 2300
 2110 L2PTR=RHSR
      IF(REX) GO TO 2120
C             RHS IS *N
      CALL JOMTCH(NIBLG,NSPUR,RHSR,0,NODE4(3),LIST1,LION1,L2,L2PTR,63)
      IF(L2.EQ.0) GO TO 2300
C             SUCCESS.  RULE  U-1  APPLIES.
 2120 NU=0
      CALL BREAK(NIBLG,NU,L1PTR,1,0,64)
      L3=0
      CALL BREAK(NIBLG,L2PTR,L3,1,0,70)
      CALL COUPLE(NIBLG,L2PTR,L1PTR,1,1,0,71)
      CALL COUPLE(NIBLG,L1PTR,L3,2,1,0,72)
      CALL TOCELL(NSPUR,L1PTR,102,LOCNID,73)
      NRUL=101
      GO TO 50
C
C     RULE  U-2 (MAYBE)
C
C        *N,LO1(N,ANY=(EXP,P1),ANY=(EXP,P2))  BECOMES
C
C        *N-1,LO1(N-1,DELETE(EXP,P1),DELETE(EXP,P2),ADD(EXP(EXP,+2,P1,P2))
C
C             *N              RHS,NODE2
C             EXP             RHSL
C             EXP             RHSR
C
C             (CONTROL COMES FROM S-2.)
 2200 IF(NODE2(4).EQ.201) GO TO 2500
      CALL JOMTCH(NIBLG,NSPUR,RHS,IL,NODE2(3),LIST1,LION1,IL,RHSL,74)
      CALL JOMTCH(NIBLG,NSPUR,RHS,IL,NODE2(3),LIST1,LION1,IR,RHSR,75)
      IF(IR.EQ.0) GO TO 2400
C             SUCCESS.  RULE U-2 APPLIES.
      RHSL=0
      CALL BREAK(NIBLG,RHS,RHSL,1,0,76)
      P2=0
      CALL BREAK(NIBLG,RHSR,P2,1,0,77)
      CALL COUPLE(NIBLG,RHSR,RHSL,1,1,0,78)
      CALL COUPLE(NIBLG,RHSL,P2,2,1,0,79)
      CALL TOCELL(NSPUR,RHSL,201,LOCNID,80)
      NRUL=102
      GO TO 50
C
C     RULE  U-3 (MAYBE)
C
C        /,L1(**,P1,RQ1'*N,LO1(N,ANY=(**,P1,RQ1)),L2(**,P2,RQ1'
C        *M,LO2(M,ANY=(**,P2,RQ1))  BECOMES
C
C        L1(NULL'/,*N-1,LO1(N-1,DELETE(**,P1,RQ1)),L2(**,-,P1,P2,Q1'
C        *M,LO2(M,DELETE(**,P2,Q1),ADD(**,-,P1,P2,Q1))
C
```

```
C             /                RHS,NODE2
C             L1               RHSL,NODE3
C             L2               RHSR,NODE4
C
C             (ENTRY IS FROM RULE U-1.)
 2300 LEX=NODE3(4).EQ.104
      REX=NODE4(4).EQ.104
      IF(.NOT.((LEX.OR.LMUL).AND.(REX.OR.RMUL))) GO TO 40
C             ENTRY IS ** OR *N IN BOTH OPERANDS.
      L1PTR=RHSL
      IF(LEX) GO TO 2310
C             LHS IS *N.
      CALL JOMTCH(NIBLG,NSPUR,RHSL,0,NODE3(3),LIST1,LION2,L1,L1PTR,80)
      IF(L1.EQ.0) GO TO 40
 2310 LQ1L=LNKFNT(NIBLG,L1PTR,NU,2,30)
      CALL FRMCEL(NSPUR,LQ1L,LVIF,LOCVIF,82)
      IF(LVIF.EQ.1) GO TO 40
      L2PTR=RHSR
      IF(REX) GO TO 2320
C             RHS IS *N.
      CALL JOMTCH(NIBLG,NSPUR,RHSR,0,NODE4(3),LIST1,LION2,L2,L2PTR,83)
      IF(L2.EQ.0) GO TO 40
 2320 LQ1R=LNKFNT(NIBLG,L2PTR,NU,2,31)
C             PUT TWO OPERANDS IN FUNDAMENTAL FORM AND COMPARE THEM.
      CALL JOORDR(NIBLG,NSPUR,LQ1L,85)
      CALL JOORDR(NIBLG,NSPUR,LQ1R,86)
      IF(JOCPCO(NIBLG,NSPUR,LQ1L,LQ1R,87).NE.0) GO TO 40
C             SUCCESS.  RULE U-3 APPLIES.
C             LQ1L    CONTAINS Q1 LEFT.
C             LQ1R    CONTAINS Q1 RIGHT.
C             L1PTR   CONTAINS ** LEFT.
C             L2PTR   CONTAINS ** RIGHT.
C             LP2     WILL CONTAIN P2.
      IR=0
      CALL BREAK(NIBLG,IR,L1PTR,1,0,88)
      LQ1L=0
      CALL BREAK(NIBLG,L1PTR,LQ1L,2,1,89)
      LP2=0
      CALL BREAK(NIBLG,L2PTR,LP2,1,1,90)
      CALL COUPLE(NIBLG,L2PTR,L1PTR,1,1,0,91)
      CALL COUPLE(NIBLG,L1PTR,LP2,2,1,0,92)
      CALL TOCELL(NSPUR,L1PTR,102,LOCNID,93)
      CALL JORTRN(NIBLG,NSPUR,LQ1L,94)
      NRUL=103
      GO TO 50
C
C     RULE  U-4  (MAYBE)
C
C        *N,LO1(N,ANY=(**,P1,RQ1),ANY=(**,P1,RQ1))   BECOMES
C
C        *N-1,LO1(N-1,DELETE(**,P1,RQ1),DELETE(**,P2,RQ1),ADD(**,
C        +2,P1,P2,Q1))
C
C             =               EQUAL
C                             LHS,NODE1
C                             RHS,NODE2
C             **              WILL BE IN RHSL
C             **              WILL BE IN RHSR
C             Q1              WILL BE IN LQ1L
C             Q1              WILL BE IN LQ1R
C
C             (CONTROL COMES FROM RULE U-2.)
```

```
 2400 CALL JOMTCH(NIBLG,NSPUR,RHS,0,NODE2(3),LIST1,LION2,L1,RHSL,99)
      CALL JOMTCH(NIBLG,NSPUR,RHS,L1,NODE2(3),LIST1,LION2,L2,RHSR,100)
      IF(L2.EQ.0) GO TO 40
      LQ1L=LNKFNT(NIBLG,RHSL,NU,2,34)
      LQ1R=LNKFNT(NIBLG,RHSR,NU,2,35)
      CALL FRMCEL(NSPUR,LQ1L,LVIF,LOCVIF,103)
      IF(LVIF.EQ.1) GO TO 40
      CALL JOORDR(NIBLG,NSPUR,LQ1L,104)
      CALL JOORDR(NIBLG,NSPUR,LQ1R,105)
      IF(JOCPCO(NIBLG,NSPUR,LQ1L,LQ1R,106).NE.0) GO TO 40
C           SUCCESS.  RULE U-4 APPLIES.
      RHSL=0
      CALL BREAK(NIBLG,RHS,RHSL,L1,0,107)
      LP2=0
      CALL BREAK(NIBLG,RHSR,LP2,1,1,108)
      LQ1L=0
      CALL BREAK(NIBLG,RHSL,LQ1L,2,1,109)
      CALL COUPLE(NIBLG,RHSR,RHSL,1,1,1,110)
      CALL  COUPLE(NIBLG,RHSL,LP2,2,1,1,111)
      CALL JORTRN(NIBLG,NSPUR,LQ1L,112)
      CALL TOCELL(NSPUR,RHSL,201,LOCNID,113)
      NRUL=104
      GO TO 50
C
C     RULE  U-5  (MAYBE)
C
C        +N,LO1(N,ANY=RP1 AND ANY=RP1)  BECOMES
C
C        +N-1,LO1(N-1,DELETE P1,DELETE P1,ADD *2,2.0,P1)
C
C             +N              RHS,NODE2
C             P1              RHSL(DIVERGER POSITION IL)
C             P1              RHSR(DIVERGER POSITION IR)
C
C             (CONTROL COMES FROM RULE U-2.)
 2500 P1=RHSL
      P2=RHSR
      CALL JOORDR(NIBLG,NSPUR,RHSL,114)
      CALL JOORDR(NIBLG,NSPUR,RHSR,115)
      IF(JOCPCO(NIBLG,NSPUR,RHSL,RHSR,116).NE.0) GO TO 2600
C           SUCCESS.  RULE U-5 APPLIES.
      RHSR=0
      CALL BREAK(NIBLG,RHS,RHSR,IR,0,117)
      RHSL=0
      CALL BREAK(NIBLG,RHS,RHSL,IL,0,118)
      CALL JORTRN(NIBLG,NSPUR,RHSR,119)
C           CREATE NEW OPERAND STRING WITH * AND 2.0 IN IT.
      CALL NEWCEL(NSPUR,RHSR,120)
      NOPTV(3)=202
      CALL TOCELL(NSPUR,RHSR,NOPTV,LOCN3,121)
      CALL NEWCEL(NSPUR,NCC,122)
      CALL RDPA(2.000,NCST(3))
      CALL TOCELL(NSPUR,NCC,NCST,LOCN4,123)
      CALL COUPLE(NIBLG,NCC,NWKEND,1,1,0,124)
      CALL COUPLE(NIBLG,RHSR,NCC,1,1,0,125)
      CALL COUPLE(NIBLG,RHS,RHSR,1,1,0,126)
      CALL COUPLE(NIBLG,RHSR,RHSL,2,1,0,127)
      NRUL=105
      GO TO 50
C
C     RULE  U-6  (MAYBE)
C
```

```
C        +N,*N,LO1(N,ANY=RP1),RP1    BECOMES
C
C        +N-1,*2,+2,*N-1,LO1(*N-1,DELETE P1),1.0,P1
C
C             +N              RHS,NODE2
C             *N              RHSL. WILL BE IN NODE3 (DIVERGER POSITION
C                             IL)
C             P1              RHSR (DIVERGER POSITION IR)
C
C             (CONTROL COMES FROM U-5.)
 2600 K=0
 2610 CALL FRMCEL(NSPUR,RHSL,NODE3,LOCALL,128)
      IF(NODE3(2).NE.2.OR.NODE3(4).NE.202) GO TO 2700
C             OPERAND RHSL IS *N.  LOCATE OPERAND OF *N WITH VIF SET.
      CALL JOMTCH(NIBLG,NSPUR,RHSL,0,NODE3(3),LIST,1,IL1,P1,129)
C             DOES THIS OPERAND MATCH RHSR
      IF(JOCPCO(NIBLG,NSPUR,P1,RHSR,130).NE.0) GO TO 2700
C             SUCCESS.  RULE U-6 APPLIES.
      CALL BREAK(NIBLG,RHS,RHSR,NU,0,131)
      P1=0
      CALL BREAK(NIBLG,RHSL,P1,IL1,0,132)
      CALL JORTRN(NIBLG,NSPUR,RHSR,133)
      CALL BREAK(NIBLG,RHS,RHSL,NU,0,134)
C             CREATE A *,+,1.0 OPERAND.
      CALL NEWCEL( NSPUR,RHSR,135)
      NOPTV(3)=202
      CALL TOCELL(NSPUR,RHSR,NOPTV,LOCN3,136)
      CALL NEWCEL(NSPUR,NCC,137)
      NOPTV(3)=201
      CALL TOCELL(NSPUR,NCC,NOPTV,LOCN3,138)
      CALL NEWCEL(NSPUR,NC2,139)
      CALL RDPA(1.000,NCST(3))
      CALL TOCELL(NSPUR,NC2,NCST,LOCN4,140)
      CALL COUPLE(NIBLG,NCC,NC2,1,1,1,141)
      CALL COUPLE(NIBLG,RHSR,NCC,1,1,1,142)
      CALL COUPLE(NIBLG,NCC,RHSL,1,1,0,143)
      CALL COUPLE(NIBLG,RHS,RHSR,1,1,0,144)
      CALL COUPLE(NIBLG,RHSR,P1,2,1,0,145)
      NRUL=K+106
      GO TO 50
C
C     RULE  U-7 (MAYBE)
C
C        +N,RP1,*N,LO1(N,ANY=RP1)   BECOMES
C
C        +N-1,*2,*N-1,LO1(*N-1,DELETE P1),1.0,P1
C
C             +N              RHS,NODE2
C             *N              WILL BE IN RHSL
C             P1              WILL BE IN RHSR
C
C             (CONTROL COMES FROM U-6.)
 2700 IF(K.EQ.1) GO TO 2800
      P2=P1
      IL2=IL1
      K=RHSR
      RHSR=RHSL
      RHSL=K
      K=1
      GO TO 2610
C
C     RULE  U-8
```

```
C
C       +N,*N,LO1(N,ANY=RP1),*M,LO2(M,ANY=RP1)  BECOMES
C
C       +N-1,*2,+2,*N-1,LO1(N-1,DELETE P1),*M-1,LO2(M-1,
C       DELETE P1),P1
C
C            +N              RHS
C            *N              RHSR
C            *M              RHSL
C            RP1 OF *N       P2
C            RP1 OF *M       P1
C
C            (CONTROL COMES FROM U-7.)
 2800 IF(P1.EQ.RHSL.OR.P2.EQ.RHSR) GO TO 40
C            COMPARE P1 AND P2.
      IF(JOCPCO(NIBLG,NSPUR,P1,P2,146).NE.0) GO TO 40
C            OPERANDS ARE THE SAME.  RULE U-8 APPLIES.
      CALL BREAK(NIBLG,RHS,RHSR,NU,0,147)
      CALL BREAK(NIBLG,RHS,RHSL,NU,0,148)
      P1=0
      CALL BREAK(NIBLG,RHSL,P1,IL1,0,149)
      P2=0
      CALL BREAK(NIBLG,RHSR,P2,IL2,0,150)
      CALL JORTRN(NIBLG,NSPUR,P2,160)
C            CONSTRUCT  *,+ OPERAND.
C            * IS IN NCC, +IN NC1.
      DO 2810 I=1,2
      CALL NEWCEL(NSPUR,NCC,161)
      NOPTV(3)=200+I
      CALL TOCELL(NSPUR,NCC,NOPTV,LOCN3,162)
      IF(I.EQ.1) NC1=NCC
 2810 CONTINUE
      CALL CCUPLE(NIBLG,NCC,P1,1,1,1,163)
      CALL COUPLE(NIBLG,NCC,NC1,1,1,0,164)
      CALL CCUPLE(NIBLG,NC1,RHSL,1,1,1,165)
      CALL COUPLE(NIBLG,NC1,RHSR,1,1,0,166)
      NRUL=108
      GO TO 50
C
C     RULE  U-9 (MAYBE)
C
C        -,P1,P1   BECOMES
C        0.0
C
C        -,P1,+,P1   BECOMES
C        -1,+
C
C        -,P1,+,*N,P1   BECOMES
C        -1,+,*2,(-,1.0,*N-1),P1
C
C        -,(+N,P1),P1   BECOMES
C        COMP,+N-1
C
C        -,(+N,P1),(+M,P1)   BECOMES
C        -,(+N-1,DELETE P1),(+M-1,DELETE P1)
C
C        -,(+N,P1),(+M,*M,P1)   BECOMES
C        -,(+N-1,DELETE P1),(+M,(*2,(-,1.0,*M),P1))
C
C        -,(+N,*N,P1),P1   BECOMES
C        COMP,+N,(*2,(-,1.0,*N-1),P1)
C
```

```
C        -,(+N,*N,P1),(+M,P1)   BECOMES
C        -,(+N,(*2,(-,1.0,*N-1),P1),+M-1
C
C        -,(+N,*N,P1),(+M,*M,P1)   BECOMES
C        -,(+N-1),(+M,*2,(-,*N-1,*M-1),P1)
C
C                -                RHS,NODE2
C        LEFT OPERAND             RHSL,NODE3
C        RIGHT OPERAND            RHSR,NODE4
 2900 LL(1,1)=RHSL
      LL(2,1)=RHSR
      DO 2910 I=1,2
      DO 2910 J=2,3
 2910 LL(I,J)=0
      DO 2940 I=1,2
      IF(NODE3(2).NE.2.OR.NODE3(4).NE.201) GO TO 2920
      CALL JOMTCH(NIBLG,NSPUR,LL(I,1),0,NODE3(3),LIST1,LION3,NU,LL(I,2)
     1           ,167)
      IF(NU.EQ.0) GO TO 2920
      CALL JOMTCH(NIBLG,NSPUR,LL(I,2),0,NODE3(3),LIST,1,NU,LL(I,3),168)
 2920 DO 2930 J=1,4
 2930 NODE3(J)=NODE4(J)
 2940 CALL JOCROR(NIBLG,NSPUR,LL(I,1),169)
C             COMPARE OPERANDS AT EACH LEVEL
      DO 2960 I=1,3
      IF(LL(1,I).EQ.0) GO TO 40
      DO 2950 J=1,3
      IF(LL(2,J).EQ.0) GO TO 2960
      IF(JOCPCO(NIBLG,NSPUR,LL(1,I),LL(2,J),170).NE.0) GO TO 2950
      IREM=I
      JREM=J
      GO TO 2970
 2950 CONTINUE
 2960 CONTINUE
      GO TO 40
C
C             SUCCESS.  RULE  U-8  APPLIES.
 2970 I=IREM
      J=JREM
      NRUL=108
C             DELETE P1 ON LEFT HAND SIDE.
      NU=0
      CALL BREAK(NIBLG,NU,LL(1,I),1,0,171)
      CALL JORTRN(NIBLG,NSPUR,LL(1,I),172)
C             BREAK INTO-P1 LINK ON RIGHT HAND SIDE.
      NU=0
      CALL BREAK(NIBLG,NU,LL(2,J),I,0,173)
      IF(I.EQ.3.OR.J.EQ.3) GO TO 2990
C             RETURN P1 FROM RIGHT HAND SIDE.
      CALL JORTRN(NIBLG,NSPUR,LL(2,J),174)
      IF(J.EQ.2) GO TO 50
      IF(I.EQ.2) GO TO 2980
C             CHANGE - IN RHS TO 0.0-NWKEND AND EXIT.
      CALL RDPA(0.0D0,NCST(3))
      CALL TOCELL(NSPUR,RHS,NCST,LOCN4,175)
      CALL COUPLE(NIBLG,RHS,NWKEND,0,0,0,176)
      GO TO 50
C             CHANGE - IN RHS TO COMP, THEN EXIT.
 2980 CALL TOCELL(NSPUR,RHS,1,LOCNID,177)
      GO TO 50
C             CREATE * WITH TWO OPERANDS OF - AND P1.
 2990 CALL NEWCEL(NSPUR,LASTR,177)
```

```
      NOPTV(3)=202
      CALL TOCELL(NSPUR,LASTR,NOPTV,LOCN3,178)
      CALL NEWCEL(NSPUR,MINUS,179)
      NOPTV(3)=102
      CALL TOCELL(NSPUR,MINUS,NOPTV,LOCN3,180)
      CALL COUPLE(NIBLG,LASTR,LL(2,J),0,0,0,181)
      CALL COUPLE(NIBLG,LASTR,MINUS,0,0,0,182)
      IF(J.NE.3) GO TO 2992
C             BREAK +,* LINK OF ORIGINAL RHS AND THEN COUPLE +,LASTR
C             AND MINUS,*.
      NU=0
      CALL BREAK(NIBLG,NU,LL(2,2),1,0,183)
      CALL COUPLE(NIBLG,MINUS,LL(2,2),0,0,0,184)
      CALL COUPLE(NIBLG,LL(2,1),LASTR,0,0,0,185)
      IF(I.NE.3) GO TO 2994
C             BREAK +,* LINK OF ORIGINAL LHS.  COUPLE MINUS,*.
 2992 NU=0
      CALL BREAK(NIBLG,NU,LL(1,2),1,0,186)
      CALL COUPLE(NIBLG,MINUS,LL(1,2),0,0,0,186)
C             IF APPROPRIATE, COUPLE + TO LASTR.
      IF(J.LT.3) CALL COUPLE(NIBLG,LL(1,1),LASTR,0,0,0,187)
      IF(I.EQ.J) GO TO 50
C             CREATE A 1.0-NWKEND AND COUPLE TO MINUS, TOP POSITION.
 2994 CALL NEWCEL(NSPUR,NONE,188)
      CALL ROPA(1.000,NCST(3))
      CALL TOCELL(NSPUR,NONE,NCST,LOCN4,189)
      CALL COUPLE(NIBLG,NONE,NWKEND,0,0,0,190)
      CALL COUPLE(NIBLG,MINUS,NONE,0,0,0,191)
C             IF J=1, CHANGE - TO COMP IN RHS.
      IF(J.EQ.1) CALL TOCELL(NSPUR,RHS,1,LOCNIO,192)
      GO TO 50
      END
```

```
      SUBROUTINE JORULT(NIBLG,NSPUR,IOOPR,IOCALL)
      COMMON/ALLOC/JCCORE(1)
      DIMENSION NSPUR(1)
      DIMENSION NODE1(6),NODE2(6),NODE3(6)
      INTEGER RHS,RHSR
      DOUBLE PRECISION ZERO,ONE
      LOGICAL RHSC,LHSC,RHSZ,LHSZ,RHSO,LHSO
C             LOCAL VECTORS REQUIRED
C             LOCALL   5 ITEMS,VIF,TYPE FLAG,NOP,NID,VALUE.
      DIMENSION LOCALL(13),LOCVIF(4),LOCNID(4),LOCVAL(4)
      DATA LOCALL/4,9,1,2,5,1,1,7,1,1,8,1,0/
      DATA LOCVIF/1,9,1,1/
      DATA LOCNID/1,7,1,1/
      DATA LOCVAL/1,8,1,0/
C
      CALL SYSENT(1,3,5,42,IOCALL)
      I=JCCORE(17)+1
      LOCALL(13)=JCCORE(I)
      LOCVAL(4)=LOCALL(13)
      LHOCL=0
      NIBLG3=NIBLG+2
      CALL RDPA(0.D0,ZERO)
      CALL RDPA(1.D0,ONE)
    5 IFLAG=1
C             CLEAR VIF'S
      CALL JOSVIF(NIBLG,NSPUR,ICNT,0,1)
   10 IF(IFLAG.EQ.0) GO TO 7000
   15 CALL JORLSE(NIBLG,LHOCL,1)
      IFLAG=0
C             OBTAIN FIRST NODE IN NETWORK
      LNK=IOOPR
      JCCORE(NIBLG3)=JCCORE(NIBLG3)+1
      GO TO 25
C             NEXT ENTRY ON OPERAND TO BE CHECKED FOR SIMPLIFICATION.
   20 LNK =NEXT(NIBLG, LNK,JCCORE(NIBLG3), 2,-1,LHOCL,0,0,2)
      IF(LNK.EQ.0) GO TO 10
C             DETERMINE CONTENTS OF NODE.
   25 CALL FRMCEL(NSPUR,LNK,NODE1,LOCALL,3)
C             DETERMINE IF ANY SIMPLIFICATION RULES SHOULD BE APPLIED.
C             IF AN OPERAND OR CONSTANT, GO TO NEXT NODE.
C             IF VIF FLAG IS SET, GO TO NEXT NODE. (THE VIF FLAG IS
C             USED TO INDICATE WHICH NODES HAVE BEEN ALTERED DURING
C             THE LAST PASS THROUGH THE NETWORK)
      IF(NODE1(1).NE.0.OR.NODE1(2).LT.2) GO TO 6000
      LNB=LNKBNT(NIBLG,LNK,NU,1,4)
      J=NODE1(4)/100+1
      IF(J.GT.1.AND.NODE1(3).EQ.1) GO TO 500
      GO TO(30,40,90),J
C             UNARY OPERATOR. OBTAIN OPERAND.
   30 LHS=LNKFNT(NIBLG,LNK,NU,1,5)
      CALL FRMCEL(NSPUR,LHS,NODE2,LOCALL,6)
C             IS IT A CONSTANT, APPLY RULE T-20 IF SO.
      IF(NODE2(2).EQ.0) GO TO 2000
C             CHECK IF UNARY OPERATOR IS COMP FOLLOWED BY -.
      IF(NODE1(4).NE.1.OR.NODE2(2).NE.2.OR.NODE2(4).NE.102) GO TO 6000
      GO TO 1800
C             BINARY OPERATOR.  OBTAIN TWO OPERANDS.
   40 IF(NODE1(4).EQ.101) GO TO 6000
      LHS=LNKFNT(NIBLG,LNK,NU,1,7)
      RHS=LNKFNT(NIBLG,LNK,NU,2,8)
      CALL FRMCEL(NSPUR,LHS,NODE2,LOCALL,9)
      CALL FRMCEL(NSPUR,RHS,NODE3,LOCALL,10)
```

```
C             SET INDICATORS AS TO WHETHER OPERANDS ARE
C             CONSTANTS, ZERO, OR ONE.
      LHSC=NODE2(2).EQ.0
      LHSZ=LHSC.AND.MATHCP(303,ZERO,NODE2(5),11).EQ.0
      LHSC=LHSC.AND.MATHCP(303,ONE,NODE2(5),12).EQ.0
      RHSC=NODE3(2).EQ.0
      RHSZ=RHSC.AND.MATHCP(303,ZERO,NODE3(5),13).EQ.0
      RHSC=RHSC.AND.MATHCP(303,ONE,NODE3(5),14).EQ.0
      J=NODE1(4)-101
      IF(LHSC.AND.RHSC) GO TO 1900
      GO TO(50,70,80),J
C             BINARY OPERATOR IS -
   50 IF(LHSZ) GO TO 600
      IF(RHSZ) GO TO 700
C             NEITHER IS ZERO.  IS EITHER OPERAND COMP
      IF(NODE2(2).EQ.2.AND.NODE2(4).EQ.1) GO TO 800
      IF(NODE3(2).EQ.2.AND.NODE3(4).EQ.1) GO TO 900
   60 L=NODE1(4)+99
      IF(NODE2(2).EQ.2.AND.NODE3(2).EQ.2.AND.NODE2(4).EQ.L.AND.
     1    NODE3(4).EQ.L) GO TO 2100
      IF(NODE3(2).EQ.2.AND.NODE3(4).EQ.NODE1(4)) GO TO 2400
      IF(NODE2(2).EQ.2.AND.NODE2(4).EQ.NODE1(4)) GO TO 2500
      GO TO 6000
C             BINARY OPERATOR IS /
   70 IF(LHSO) GO TO 1200
      IF(RHSO) GO TO 1300
      IF(RHSZ) GO TO 1400
      IF(LHSZ) GO TO 1500
C             IS EITHER OPERAND RECIP
      IF(NODE3(2).EQ.2.AND.NODE3(4).EQ.10) GO TO 1600
      IF(NODE2(2).EQ.2.AND.NODE2(4).EQ.10) GO TO 1700
      GO TO 60
C             BINARY OPERATOR IS **
   80 IF(RHSZ.OR.RHSO) GO TO 1000
      IF(LHSZ.OR.LHSO) GO TO 1100
      GO TO 6000
C             MULTIOPERAND OPERATOR
   90 N=NODE1(3)
      K=NODE1(4)-99
      RHS=0
C             SEARCH OPERANDS FOR APPROPRIATE PATTERNS
      DO 130 I=1,N
      IOX=I
      LHS=LNKFNT(NIBLG,LNK,NU,I,15)
      CALL FRMCEL(NSPUR,LHS,NODE2,LOCALL,16)
      LHSC=NODE2(2).EQ.0
      LHSZ=LHSC.AND.MATHCP(303,ZERO,NODE2(5),17).EQ.0
      LHSO=LHSC.AND.MATHCP(303,ONE,NODE2(5),18).EQ.0
      IF(NODE2(2).EQ.2.AND.NODE2(4).EQ.K) GO TO 2300
      IF(NODE1(4).EQ.201) GO TO 120
C             OPERATOR IN LNK IS *N
      IF(LHSZ.OR.LHSO) GO TO 150
C             IS LHS OPERAND HEADED BY RECIP
      IF(NODE2(2).EQ.2.AND.NODE2(4).EQ.10) GO TO 200
C             IS OPERAND A CONSTANT
  100 IF(.NOT.LHSC) GO TO 130
      IF(RHS.NE.0) GO TO 2200
      DO 110 J=1,5
  110 NODE3(J)=NODE2(J)
      RHS=LHS
      GO TO 130
C             OPERATOR IS +N
```

```
  120 IF(LHSZ) GO TO 300
      IF(NODE2(2).EQ.2.AND.NODE2(4).EQ.1) GO TO 400
      GO TO 100
C             END OF DO LOOP TO LOOK AT OPERANDS
  130 CONTINUE
      GO TO 6000
C
C*********************************************************************
C
C     RULE  T-1
C
C        *N,LO1(N,ANY=L1(0.0'1.0))  BECOMES
C
C        L1(0.0'*N-1,DELETE 1.0)
C
C             *N              LNK,NODE1
C             0.0'1.0         LHS,NODE2
C             LHSZ OR LHSO    INDICATES WHETHER LHS IS ZERO OR ONE
C             LNB             LINK BACK FROM *N
  150 CALL BREAK(NIBLG,LNK,LHS,NU,0,19)
      IF(LHSO) GO TO 160
C             LHS IS ZERO.
      CALL BREAK(NIBLG,LNB,LNK,LNBPOS,0,20)
      CALL JORTRN(NIBLG,NSPUR,LNK,21)
      CALL COUPLE(NIBLG,LNB,LHS,LNBPOS,0,0,22)
      LNK=LHS
      GO TO 4990
C             LHS IS ONE.
  160 CALL JORTRN(NIBLG,NSPUR,LHS,23)
      GO TO 5000
C
C     RULE  T-2
C
C        *N,LO1(N,ANY=RECIP,01)   BECOMES
C
C        /,01,*N-1,LO1(N-1,DELETE RECIP,01)
C
C             *N              LNK,NODE1
C             RECIP           LHS,NODE2
C             LNB             LINK BACK FROM *N
C
C             BREAK LNB,*N AND *N,RECIP LINKS
  200 K=103
  210 CALL BREAK(NIBLG,LNB,LNK,LNBPOS,0,24)
      CALL BREAK(NIBLG,LNK,LHS,NU,0,25)
C             CHANGE RECIP TO / OR -
      CALL TOCELL(NSPUR,LHS,K,LOCNID,26)
C             COUPLE LNB TO /, / TO *N-1
      CALL COUPLE(NIBLG,LNB,LHS,LNBPOS,0,0,27)
      CALL COUPLE(NIBLG,LHS,LNK,2,0,0,28)
      LNK=LHS
      GO TO 4990
C
C     RULE  T-3
C
C        +N,LO1(N,ANY=0.0)   BECOMES
C
C        +N-1,LO1(N-1,DELETE 0.0)
C
C             +N              LNK,NODE1
C             0.0             LHS,NODE2
  300 CALL BREAK(NIBLG,LNK,LHS,NU,0,29)
```

```
      CALL JORTRN(NIBLG,NSPUR,LHS,30)
      GO TO 5000
C
C     RULE  T-4
C
C        +N,LO1(N,ANY=COMP,02)   BECOMES
C
C        -,02,+N-1,LO1(N-1,DELETE COMP,02)
C
C             +N              LNK,NODE1
C             COMP            LHS,NODE2
C             LNB             LINK BACK FROM +N
  400 K=102
      GO TO 210
C
C     RULE  T-5
C
C        (*1'+1'-1'/1'**1),01   BECOMES
C
C        01
C
C             OP              LNK
C             01              WILL BE IN LHS
  500 NU=1
      LHS=0
C             BREAK LNB,LNK AND LNK,01 LINKS
  510 CALL BREAK(NIBLG,LNB,LNK,LNBPOS,0,31)
      CALL BREAK(NIBLG,LNK,LHS,NU,0,32)
C             RETURN OPERATOR
      CALL RETURN(NSPUR,LNK,LNK,33)
C             COUPLE LNB,01
      CALL COUPLE(NIBLG,LNB,LHS,LNBPOS,0,0,34)
      LNK=LHS
      GO TO 4990
C
C     RULE  T-6
C
C        -,0.0,01   BECOMES
C
C        01
C
C             -               LNK,NODE1
C             0.0             LHS,NODE2
C             01              RHS,NODE3
  600 CALL BREAK(NIBLG,LNB,LNK,LNBPOS,0,35)
      NU=0
      CALL BREAK(NIBLG,LNK,NU,2,0,36)
      CALL JORTRN(NIBLG,NSPUR,LNK,37)
      CALL COUPLE(NIBLG,LNB,RHS,LNBPOS,0,0,38)
      LNK=RHS
      GO TO 4990
C
C     RULE  T-7
C
C        -,01,0.0   BECOMES
C
C        COMP,01
C
C             -               LNK,NODE1
C             01              LHS,NODE2
C             0.0             RHS,NODE3
  700 K=1
```

```
  710 NU=0
      CALL BREAK(NIBLG,LNK,NU,2,0,39)
C             CHANGE - TO COMP
      CALL TOCELL(NSPUR,LNK,K,LOCNID,40)
      CALL JORTRN(NIBLG,NSPUR,RHS,41)
      GO TO 5000
C
C     RULE  T-8
C
C        -,COMP,01,02    BECOMES
C
C        +2,01,02
C
C              -                LNK
C             COMP              LHS
C             02                RHS
C             01                WILL BE IN LHS1
  800 K=201
  810 LHS1=LNKFNT(NIBLG,LHS,NU,1,42)
      CALL BREAK(NIBLG,LNK,LHS,NU,1,43)
      CALL BREAK(NIBLG,LHS,LHS1,NU,0,44)
      CALL COUPLE(NIBLG,LNK,LHS1,1,1,0,45)
C             CHANGE - TO +
      CALL TOCELL(NSPUR,LNK,K,LOCNID,46)
      CALL RETURN(NSPUR,LHS,LHS,47)
      GO TO 5000
C
C     RULE  T-9
C
C        -,01,COMP,02    BECOMES
C
C        COMP,+2,01,02
C
C              -                LNK
C             01                LHS
C             COMP              RHS
  900 K=201
  910 CALL BREAK(NIBLG,LNK,LHS,NU,0,48)
      CALL TOCELL(NSPUR,LNK,NODE3(4),LOCNID,49)
      CALL TOCELL(NSPUR,RHS,K,LOCNID,50)
      CALL COUPLE(NIBLG,RHS,LHS,1,1,0,51)
      GO TO 5000
C
C     RULE  T-10
C
C        **,01,L1(0.0'1.0)    BECOMES
C
C        L1(0.0'1.0)
C
C              **               LNK
C             01                LHS
C             (0.0'1.0)         RHS
 1000 GO TO 600
C
C     RULE  T-11
C
C        **,L1(0.0'1.0),01    BECOMES
C
C        L1(1.0'01)
C
C              **               LNK
C             (0.0'1.0)         LHS
```

```
C              01                RHS
 1100 CALL BREAK(NIBLG,LNB,LNK,LNBPOS,0,52)
      NU=0
      IF(LHSZ) GO TO 1110
C             LHS IS ONE
      CALL BREAK(NIBLG,LNK,NU,2,0,53)
      LHS=RHS
      GO TO 1120
C             LHS IS ZERO
 1110 CALL BREAK(NIBLG,LNK,NU,1,0,54)
      CALL TOCELL(NSPUR,LHS,ONE,LOCVAL,55)
 1120 CALL JORTRN(NIBLG,NSPUR,LNK,56)
      CALL COUPLE(NIBLG,LNB,LHS,LNBPOS,1,0,57)
      LNK=LHS
      GO TO 4990
C
C     RULE  T-12
C
C        /,1.0,01   BECOMES
C
C        01
C
C             /                LNK
C             1.0              LHS
C             01               RHS
 1200 GO TO 600
C
C     RULE  T-13
C
C        /,01,1.0   BECOMES
C
C        RECIP,01
C
C             /                LNK
C             01               LHS
C             1.0              RHS
 1300 K=10
      GO TO 710
C
C     RULE  T-14
C
C        /,01,0.0   BECOMES
C
C        0.0
C
C             /                LNK
C             01               LHS
C             0.0              RHS
 1400 GO TO 600
C
C     RULE  T-15
C
C        /,0.0,01   BECOMES
C
C        =,0.0,01      (ERASE ENTIRE OLD EQUATION)
C
C             /                LNK
C             0.0              LHS
C             01               RHS
 1500 CALL BREAK(NIBLG,LNB,LNK,NU,0,58)
      CALL TOCELL(NSPUR,LNK,101,LOCNID,59)
      CALL JORTRN(NIBLG,NSPUR,JCCORE(NIBLG),60)
```

```
      JCCORE(NIBLG)=LNK
      IDOPR=LNK
      GO TO 5
C
C     RULE  T-16
C
C        /,01,RECIP,02   BECOMES
C
C        RECIP,*2,01,02
C
C             /              LNK
C             01             LHS
C             RECIP          RHS
 1600 K=202
      GO TO 910
C
C     RULE  T-17
C
C        /,RECIP,01,02   BECOMES
C
C        *2,01,02
C
C             /              LNK
C             RECIP          LHS
C             02             RHS
 1700 K=202
      GO TO 810
C
C     RULE  T-18
C
C        COMP,-,01,02   BECOMES
C
C        -,02,01
C
C             COMP           LNK
C             -              LHS
C             01             WILL BE IN RHS
 1800 RHS=0
      CALL SYSDMP
      CALL BREAK(NIBLG,LHS,RHS,1,1,61)
      CALL COUPLE(NIBLG,LHS,RHS,2,0,0,62)
      GO TO 510
C
C     RULE  T-19
C
C        (-'/'**),C1,C2   BECOMES
C
C        E
C
C             OP             LNK
C             C1             LHS
C             C2             RHS
 1900 CALL MATH(NODE1(4),NODE3(5),NODE2(5),NODE2(5),63)
      CALL TOCELL(NSPUR,RHS,NODE2(5),LOCVAL,64)
      GO TO 600
C
C     RULE  T-20
C
C        (COMP'SIN'COS'TAN'ARCSIN'ARCCOS'ARCTAN'ALOG'EXP'RECIP),C1
C        BECOMES
C
C        E
```

```
C
C             OP              LNK,NODE1
C             C1              LHS,NODE2
 2000 CALL MATH(NODE1(4),NODE2(5),NU,NODE2(5),65)
      CALL TOCELL(NSPUR,LHS,NODE2(5),LOCVAL,66)
      GO TO 510
C
C     RULE  T-21
C
C        L1(/'-),L1(*N'+N),LO1(N,ANY=RO1),L1(*M'+M),
C        LO2(M,ANY=RO1)   BECOMES
C
C        L1(*N-1'+N-1),LO1(N-1,DELETE O1),L1(*M-1'+M-1),LO2(M-1,
C        DELETE O1)
C
C             (/'-)           LNK,NODE1
C             (*N'+N)         LHS,NODE2
C             (*M'+M)         RHS,NODE3
C
C             J=1 FOR - AND 2 FOR /.
C             CHECK IF AN OPERAND IN THE LHS IS THE SAME AS AN OPERAND
C             ON THE RHS.  FIRST PUT BOTH OPERANDS INTO FUNDAMENTAL
C             FORM.
 2100 CALL JOORDR(NIBLG,NSPUR,LHS,67)
      CALL JOORDR(NIBLG,NSPUR,RHS,68)
C             SEARCH BOTH OPERAND LISTS TO FIND EQUIVALENT OPERANDS.
      NI=1
      MI=1
      LNI=LNKFNT(NIBLG,LHS,NU,NI,69)
 2110 LMI=LNKFNT(NIBLG,RHS,NU,MI,70)
 2120 IF(JOCPCO(NIBLG,NSPUR,LNI,LMI,71)) 2130,2150,2140
C             OPERAND LNI IS LESS THAN OPERAND LMI
 2130 IF(NI.GE.NODE2(3)) GO TO 6000
      NI=NI+1
      LNI=LNKFNT(NIBLG,LHS,NU,NI,72)
      GO TO 2120
C             OPERAND LMI IS LESS THAN OPERAND LNI
 2140 IF(MI.GE.NODE3(3)) GO TO 6000
      MI=MI+1
      GO TO 2110
C             SEARCH IS SUCCESSFUL.  OPERAND LNI IS SAME AS LMI.
 2150 NU=0
      CALL BREAK(NIBLG,LHS,NU,NI,0,73)
      NU=0
      CALL BREAK(NIBLG,RHS,NU,MI,0,74)
      CALL JORTRN(NIBLG,NSPUR,LNI,75)
      CALL JORTRN(NIBLG,NSPUR,LMI,76)
      GO TO 5000
C
C     RULE  T-22
C
C        L1(*N'+N),LO1(N,ANY=C1 AND ANY=C2)   BECOMES
C
C        L1(*N-1'+N-1),LO1(N-1,ADD L1(C1*C2'C1+C2),DELETE C1 AND C2)
C
C             (*N'+N)         LNK,NODE1
C             C1              RHS,NODE3
C             C2              LHS,NODE2
 2200 CALL MATH(NODE1(4),NODE3(5),NODE2(5),NODE2(5),77)
      CALL TOCELL(NSPUR,RHS,NODE2(5),LOCVAL,78)
      GO TO 300
C
```

```
C      RULE  T-23
C
C         L1(*N'+N),LO1(N,ANY=(L1(/'-),O1),O2)  BECOMES
C
C         L1(/'-),O1,L1(*N'+N),LO1(N,DELETE(L1(/'-),O1,O2),ADD O2)
C
C              (*N'+N)          LNK,NODE1
C              (/'-)            LHS,NODE2
C              O2               WILL BE IN RHS
 2300 LHS=0
      CALL BREAK(NIBLG,LNK,LHS,IDX,79)
      CALL BREAK(NIBLG,LNB,LNK,LNBPOS,80)
      RHS=0
      CALL BREAK(NIBLG,LHS,RHS,2,81)
      CALL COUPLE(NIBLG,LNB,LHS,LNBPOS,1,0,82)
      CALL COUPLE(NIBLG,LHS,LNK,2,1,0,83)
      CALL COUPLE(NIBLG,LNK,RHS,IDX,0,84)
      LNK=LHS
      GO TO 4990
C
C      RULE  T-24
C
C         L1(/'-),O1,L1(/'-),O2,O3   BECOMES
C
C         L1(/'-),L1(*'+),O1,O2,O3
C
C              (/'-)            LNK,NODE1
C              O1               LHS,NODE2
C              (/'-)            RHS,NODE3
C              O3               WILL BE IN RHSR
 2400 RHSR=0
      CALL BREAK(NIBLG,RHS,RHSR,2,1,85)
      LHS=0
      CALL BREAK(NIBLG,LNK,LHS,1,1,86)
      CALL COUPLE(NIBLG,RHS,LHS,1,1,0,87)
      CALL  COUPLE(NIBLG,LNK,RHSR,2,1,0,88)
C           CHANGE /'- TO *'+.
      CALL TOCELL(NSPUR,RHS,NODE1(4)+99,LOCNID,89)
      GO TO 5000
C
C      RULE  T-25
C
C         L1(/'-),L1(/'-),O1,O2,O3
C
C         L1(/'-),O2,L1(*,+),O1,O3
C
C              (/'-)            LNK,NODE1
C              (/'-)            LHS,NODE2
C              O3               RHS,NODE3
C              O2               WILL BE IN LHSR
 2500 LHSR=0
      CALL BREAK(NIBLG,LHS,LHSR,2,1,90)
      RHS=0
      CALL BREAK(NIBLG,LNK,RHS,2,1,91)
      CALL COUPLE(NIBLG,LNK,LHSR,1,1,0,92)
      CALL COUPLE(NIBLG,LHS,RHS,2,1,0,93)
      CALL TOCELL(NSPUR,LHS,NODE1(4)+99,LOCNID,94)
      GO TO 5000
C
C*****************************************************************
C
 4990 IFLAG=2
```

```
C             INCREMENT KEY
 5000 JCCORE(NIBLG3)=JCCORE(NIBLG3)+1
C             CLEAR FLAGS OF EARLIER OPERATORS TO INDICATE OPERAND
C             HAS BEEN ALTERED AND MAY REQUIRE FURTHER SIMPLIFICATION
C             AT A LATER PASS.
      CALL TOCELL(NSPUR,LNK,0,LOCVIF,95)
      NLEV=0
 5020 CALL TOCELL(NSPUR,LNB,0,LOCVIF,96)
      IF(NLEV.GE.1) GO TO 5030
      NLEV=NLEV+1
      LNB=LNKBNT(NIBLG,LNB,NU,1,97)
      IF(LNB.GT.0) GO TO 5020
 5030 CONTINUE
      IF(IFLAG.EQ.2) GO TO 15
      IFLAG=1
      GO TO 20
C             SET VIF TO INDICATE OPERAND LNK UNCHANGED THIS PASS.
 6000 CALL TOCELL(NSPUR,LNK,1,LOCVIF,98)
      GO TO 20
C
 7000 CALL SYSEXT
      RETURN
      END
```

```
      SUBROUTINE SOLVE(PSPUR,IPS,NIBLG,IDCALL)
      COMMON/ALLOC/JCCORE(1)
      DATA INIT/0/
      INTEGER PSPUR
      DIMENSION ISPUR(5)
      DATA ISPUR/0,0,1,13,6/
C
      CALL SYSENT(1,3,5,51,IDCALL)
C             OBTAIN CORE FOR THE NETWORK INFORMATION BLOCK IF IT IS
C             NOT ALREADY SET UP.
      IF(INIT.NE.0) GO TO 30
      INIT=1
      N=JCCORE(17)+19
      LEN=JCCORE(N)+4
      IF(JCCORE(N+1).GT.LEN)LEN=JCCORE(N+1)
      ISPUR(4)=5*LEN+5
      CALL NEWCEL( ISPUR,NIBLG,2)
      K=NIBLG+5
      JCCORE(NIBLG+1)=K
      JCCORE(NIBLG+3)=JCCORE(N)*5+NIBLG
      JCCORE(NIBLG+4)=JCCORE(N+1)*5+NIBLG
      DO 20 I=1,LEN
C             USER MASK
      JCCORE(K)=ISPUR(1)
C             NUMBER OF LIST ENTRIES TO ALLOCATE PER REQUEST.
      JCCORE(K+2)=1
C             SIZE OF A LIST ENTRY.
      JCCORE(K+3)=I
C             LIST TYPE
      JCCORE(K+4)=6
   20 K=K+5
      NSPUR=JCCORE(NIBLG+4)
C
C             CONVERT POLISH STRING TO NETWORK FORM.
   30 CONTINUE
      CALL JOPSTN(NIBLG,JCCORE(NSPUR),PSPUR,IPS,NWKEND,NIOVI,4)
C             APPLY RULES S OR U TO EQUATION.
   40 CALL JORULS(NIBLG,JCCORE(NSPUR),NWKEND,NIOVI,NRUL,6)
      IF(NRUL.GT.0.AND.NRUL.NE.7) GO TO 40
C             APPLY RULES T TO SIMPLIFY THE EQUATION.
      CALL JORULT(NIBLG,JCCORE(NSPUR),JCCORE(NIBLG),8)
      CALL JONTPS(NIBLG,JCCORE(NSPUR),PSPUR,IPS,NWKEND,10)
      CALL SYSEXT
      RETURN
      END
```

```
C
C*****************************************************************
C                          ***************
C                          *   REMOTE    *
C                          *   GENERAL   *
C                          * INFORMATION *
C                          ***************
C
C       SPECIAL INSTRUCTIONS/LIST TYPE
C           ALL LIST TYPES EMPLOYED WITH REMOTE MUST BE FORWARD-
C           BACKWARD.
C
C       SPECIAL INSTRUCTIONS/LEND
C           THE FIRST DATA TYPE MUST HAVE A LEND OF (1,0,1,0).  THE
C           SECOND DATA TYPE MUST HAVE A LEND OF (2,0,1,0).  THE THIRD
C           AND FOURTH DATA TYPES INDICATE THE NUMBER OF FILE ENTRIES
C           AND THE NUMBER OF WORDS PER FILE ENTRY RESPECTIVELY.  THE
C           LEND FOR THE THIRD DATA TYPE MUST HAVE 1 AS THE SMALLER
C           WORD INDEX.  THE FOURTH DATA TYPE MUST HAVE AS THE LARGER
C           WORD INDEX OF ITS LEND THE NUMBER OF WORDS IN THE PRE-FILE
C           PARAMETER SET.
C
C       SPECIAL INSTRUCTIONS/DEBUG
C           THE FOLLOWING INDICATES THE ORDER FOR OPERATIONAL CHECKS
C           OF THE REMOTE SYSTEM.  THE PROGRAMS OF LEVEL THREE, FOR
C           INSTANCE, CALL ONLY SUBPROGRAMS FROM LEVELS ONE AND TWO.
C           SIMILARLY FOR ALL OTHER LEVELS.
C               SUBPROGRAM          LEVEL          PROGRAM ID. NUMBER
C               SUPPLY              0              4
C               C4STUP              1              10
C               C4ADR1              2              6
C               C4ADR3              3              8
C               C4NEXT              3              9
C               C4AOR2              4              7
C               C4ADRS              5              5
C               C4TRAN              6              11
C               FETCH               7              1
C               FIND                7              2
C               STORE               7              3
C
C       SPECIAL INSTRUCTIONS/COAST
C           REMOTE MUST BE SUPPORTED BY THE COAST SYSTEM, AS REMOTE IS
C           ESSENTIALLY A LIST STRUCTURE COMPOSED OF LIST ENTRIES WHICH
C           ARE EQUAL TO OR LARGER THAN A CORE SEGMENT.  THE CATALOGUE
C           AND REGISTER SEGMENTS ARE EQUAL IN SIZE TO A CORE SEGMENT.
C           THE SIZE OF THE RECORD SEGMENTS IS SPECIFIED BY JCSPUR(4).
C           THE LIST TYPE INDICATED BY JCSPUR(5) MUST BE FORWARD-
C           BACKWARD.
C
C       SPECIAL INSTRUCTIONS/OFF-LINE STORAGE
C           ONLY RECORD SEGMENTS MAY RESIDE ON OFF-LINE STORAGE.  THE
C           RECORD SEGMENT ADDRESS CONTAINED IN THE REGISTER WILL BE
C           NEGATIVE FOR ALL RECORD SEGMENTS ON OFF-LINE STORAGE.  THE
C           ABSOLUTE VALUE OF THESE NEGATIVE INTEGERS MAY BE UTILIZED
C           AS AN AIDE TO INFORMATION RETRIEVAL.  FOR THE PROGRAM
C           SUPPLY, A PROGRAM MUST BE SUBSTITUTED TO PERFORM THE
C           RETRIEVAL OPERATION IF OFF-LINE STORAGE IS TO BE EMPLOYED.
C           SUPPLY IS THE ONLY MACHINE DEPENDENT ROUTINE WITHIN REMOTE.
C           JCSPUR(4) SHOULD BE SET EQUAL TO THE RECORD LENGTH IN WORDS
C           FOR THE OFF-LINE STORAGE DEVICE UTILIZED.  THE CORE SEGMENT
C           SIZE AND JCSPUR(4) NEED NOT BE EQUAL, BUT JCSPUR(4) MUST
C           NOT BE LARGER THAN THE LARGEST ALLOWED CORE BUDDY.
```

```
C
C          CURRENT STATUS
C              OPERATIONAL
C
C*****************************************************************
C
```

```
C
C*****************************************************************
C                          **********
C                          * C4AORS *
C                          **********
C
C       PURPOSE
C           C4AORS COMPUTES THE ABSOLUTE ADDRESS OF A FILE ENTRY
C           SPECIFIED BY A LISTER VECTOR.  C4AORS ALSO COMPUTES THE
C           MODEL AND LOESS VECTORS.
C
C       USAGE
C           CALL C4AORS(JCSPUR,LISTER,LOCAL,LOESS,MODEL,IODINE,JCFINE,
C           JCLOAD,JCSORF,JCCLNO)
C
C       DATA FORMAT
C           N/A
C
C       DESCRIPTION OF PARAMETERS
C           JCSPUR = A VECTOR CONTAINING THE POINTER TO THE USER,S CORE
C                    ASSIGNMENT MASK, THE POINTER TO THE USER'S
C                    AVAILABLE SPACE LIST HEADCELL, THE NUMBER OF
C                    SEGMENTS REQUESTED, THE NUMBER OF WORDS PER
C                    SEGMENT, AND LASTLY THE INDEX SPECIFYING THE LIST
C                    TYPE.  THE JCSPUR VECTOR MUST CONTAIN THESE FIVE
C                    ENTRIES IN THE INDICATED ORDER.
C           LISTER = AN N DIMENSIONAL VECTOR SUCH THAT, LISTER(1)=N,
C                    LISTER(2) IS THE RECORD NUMBER.  THE NEXT N-3
C                    WORDS OF LISTER CONTAIN THE INTEGERS FOR DIRECTORY
C                    SELECTION.  LISTER(N) IS THE FILE ENTRY NUMBER.
C                    IF LISTER(N)=0, THE ADDRESS COMPUTED WILL BE THAT
C                    OF THE FIRST FILE WORD OF THE FILE SELECTED BY
C                    LISTER(N-1).  NO OTHER LISTER WORDS MAY BE ZERO.
C                    IF LISTER(1)=1, LISTER(2) CONTAINS THE NUMBER OF
C                    FILE ENTRIES FROM THE CURRENT ONE TO ADVANCE.  THE
C                    DIRECTION OF MOVEMENT IS INDICATED BY THE SIGN OF
C                    LISTER(2), + FOR FORWARD AND - FOR BACKWARD.
C           LOCAL  = A USER SUPPLIED VECTOR INDICATING THE DATA TYPES
C                    FOR TRANSFER.  THE FIRST WORD INDICATES THE NUMBER
C                    OF DIFFERENT DATA TYPES, AND IS FOLLOWED BY THAT
C                    NUMBER OF SETS OF THREE WORDS EACH.  THE FIRST OF
C                    A THREE WORD SET IS THE DATA TYPE NUMBER FOR THIS
C                    LIST TYPE, WHILE THE REMAINING TWO WORDS ARE
C                    INDICES INDICATING THE FIRST AND LAST OCCURRENCE
C                    OF THE DATA TYPE FOR TRANSFER.  IF THE THIRD WORD
C                    OF THE SET IS ZERO, THE LAST ITEM TRANSFERRED WILL
C                    BE THE LAST ITEM INDICATED IN THE LEND COMPONENT
C                    (DATA LOCATION ARRAY) FOR THE CURRENT DATA TYPE.
C                    IF LOCAL(1)=-1 NO DATA TRANSFER OCCURS, AND IF
C                    LOCAL(1)=0 ALL DATA IN THE FILE ENTRY IS
C                    TRANSFERRED.
C           LOESS  = A FOUR WORD VECTOR PATTERNED AFTER LOCAL.  LOESS
C                    CONTAINS THE INFORMATION SPECIFYING THE DATA ITEM
C                    FOR TRANSFER IN SUCCEEDING FILE ENTRY PARTITIONS.
C           MODEL  = A FOUR WORD VECTOR PATTERNED AFTER LOCAL.  MODEL
C                    APPLIES TO THE DATA ITEM FOR TRANSFER IN THE
C                    CURRENT FILE ENTRY PARTITION.
C           IODINE = A PARAMETER INDICATING THE THREE WORD SET IN LOCAL
C                    FROM WHICH LOESS AND MODEL ARE TO BE FORMED.
C           JCFINE = A VECTOR CONTAINING (LISTER(1)-1) COMPONENTS.  THE
C                    FIRST (LISTER(1)-3) WORDS CONTAIN THE NUMBER OF
C                    FILE ENTRIES IN THE RESPECTIVE (LISTER(1)-3)
```

```
C               DIRECTORIES.  THE LAST TWO WORDS OF JCFINE CONTAIN
C               RESPECTIVELY THE NUMBER OF FILE ENTRIES AND THE
C               NUMBER OF WORDS PER FILE ENTRY OF THE FILE
C               ULTIMATELY SELECTED BY LISTER.  THIS IS A USER
C               PROVIDED VECTOR, WHICH NEED CONTAIN MEANINGFUL
C               INFORMATION ONLY FOR THOSE DIRECTORIES AND FILES
C               WHICH HAVE NOT BEEN PREVIOUSLY ESTABLISHED.
C       JCLOAD = A THIRTY WORD VECTOR.  THE WORDS OF JCLOAD HAVE
C               THE FOLLOWING USAGE.
C                 1 = ADDRESS OF CATALOGUE OR REGISTER SEGMENT WORD
C                 2 = ADDRESS OF CATALOGUE OR REGISTER SEGMENT
C                 3 = SEQUENCE NUMBER OF CATALOGUE OR REGISTER
C                     SEGMENT
C                 4 = ENTRY NUMBER IN CATALOGUE OR REGISTER SEGMENT
C                 5 = ADDRESS OF RECORD SEGMENT WORD
C                 6 = ADDRESS OF RECORD  SEGMENT
C                 7 = WORD NUMBER IN RECORD SEGMENT
C                 8 = FILE ENTRY NUMBER
C                 9 = NUMBER OF FILE ENTRIES IN FILE
C                10 = NUMBER OF WORDS PER FILE ENTRY
C                11 = ADDRESS OF FIRST WORD OF FILE
C                12 = NUMBER OF FILE ENTRIES OR SEGMENTS TO ADVANCE
C                13 = VARIOUS
C                14 = + FOR DATA TRANSFER INTO A FILE ENTRY, AND
C                     - FOR DATA TRANSFER FROM A FILE ENTRY
C                15 = MODE INDICATOR
C                16 = FIRST WORD OF LOCAL FOR NUMBER OF FILE
C                     ENTRIES
C                20 = BASE ERROR CODE
C                21 = RELATIVE ADDRESS OF FIRST EMPTY WORD IN
C                     RECORD
C                22 = NUMBER OF WORDS IN PRE-FILE PARAMETER SET
C                23 = FIRST WORD OF LOCAL FOR NUMBER OF WORDS PER
C                     FILE ENTRY
C                27 = RELATIVE ADDRESS OF CURRENT FILE ENTRY
C                28 = ADDRESS OF FIRST RECORD WORD
C                29 = 0 FOR FILE ENTRY BOUNDARY ENFORCEMENT, AND -1
C                     FOR ALLOWED BOUNDARY VIOLATION
C                30 = ADDRESS OF POINTER TO FIRST CATALOGUE SEGMENT
C       JCSORF = SEE FETCH
C       JCCLNO = A USER ASSIGNED CALL STATEMENT ID. PARAMETER
C
C   REMARKS
C       THE NUMBER REPRESENTED BY JCTCIA MUST BE SUPPLIED IN THE
C       COMMON STATEMENT FOR SOME MACHINES.
C       IF JCLOAD HAS NOT BEEN INITIALIZED, JCLOAD(15) MUST BE SET
C       TO -1 BY THE USER PRIOR TO CALL FOR ANY REMOTE SUBROUTINE.
C       JCLOAD(29) MUST NOT EQUAL -1 UNLESS VIOLATION OF FILE ENTRY
C       BOUNDARIES IS TO BE PERMITTED.
C       THE FIRST RECORD WORD CONTAINS THE RELATIVE ADDRESS OF THE
C       FIRST EMPTY RECORD WORD.
C       THIS ROUTINE IS NOT INTENDED FOR DIRECT USER CALL.
C
C   SUBROUTINE AND FUNCTION SUBPROGRAMS REQUIRED
C       C4ADR1
C       C4ADR2
C       C4ADR3
C       SYSENT
C       SYSERR
C       SYSEXT
C       TOCELL
C
```

```
C       ERROR CODES FOR THIS PROGRAM
C           CODE FATAL  ERROR(DATA PROVIDED)
C           1    F      ILLEGAL LISTER VECTOR (JCLOAD(5))
C           2    F      ILLEGAL LOCAL VECTOR (JCLOAD(11))
C           3    F      EXCEEDED BOUNDS OF FILE ENTRY (JCLOAD(5))
C           4    F      ZERO DIRECTORY ENTRY (JCLOAD(11))
C
C       METHOD
C           C4ADRS USES LISTER TO COMPUTE THE ABSOLUTE ADDRESS OF THE
C           SELECTED FILE ENTRY.  JCLOAD(5) IS THIS ADDRESS.
C
C                                  **********
C
      SUBROUTINE C4ADRS(JCSPUR,LISTER,LOCAL,LOESS,MODEL,IODINE,JCFINE,
     1 JCLOAD,JCSORF,JCCLNO)
      COMMON/ALLOC/JCCORE(1)
      DIMENSION JCSPUR(1),LISTER(1),LOCAL(1),LOESS(1),MODEL(1),
     1 JCFINE(1),JCLOAD(1)
C                     ENTER
      CALL SYSENT(1,2,1,5,JCCLNO)
C                     SELECTING PROPER MODE
      IF(LISTER(1).GT.1)GO TO 1
      IF(LISTER(1).LE.0)CALL SYSERR(-1,JCLOAD(5))
C                     ADVANCING FROM CURRENT FILE ENTRY
      MODEL(1)=-1
      LOESS(1)=-1
      JCLOAD(12)=LISTER(2)
      JCLOAD(15)=4
      CALL C4ADR2(JCSPUR,JCLOAD,JCSORF,1)
      IF(JCSORF.EQ.-1)GO TO 104
      M1=JCLOAD(28)
      JCCORE(M1)=JCLOAD(21)
      GO TO 9
C                     FURTHER MODE SELECTION
    1 IF(JCLOAD(15).GT.0)GO TO 3
C                     INITIALIZATION OF JCLOAD
      DO 2 I=1,13
      JCLOAD(I)=0
      JCDUM1=I+14
    2 JCLOAD(JCDUM1)=0
      M1=JCSPUR(5)+20
      JCLOAD(22)=JCCORE(M1)+13
      M1=JCLOAD(22)
      JCDUM1=JCCORE(M1)
      JCLOAD(22)=JCCORE(M1+1)
      JCDUM1=JCLOAD(22)-JCDUM1+1
      JCLOAD(16)=1
      JCLOAD(17)=3
      JCLOAD(18)=1
      JCLOAD(19)=1
      JCLOAD(21)=2
      JCLOAD(23)=1
      JCLOAD(24)=4
      JCLOAD(25)=JCDUM1
      JCLOAD(26)=JCDUM1
      JCLOAD(28)=0
C                     ACCESSING CATALOGUE SEGMENT POINTER TO REGISTER
    3 JCLOAD(1)=JCLOAD(30)
      JCLOAD(4)=LISTER(2)
      JCLOAD(12)=0
      JCLOAD(15)=1
      CALL C4ADR1(JCSPUR,JCLOAD,JCSORF,1)
```

```
      IF(JCSORF.EQ.-1)GO TO 104
      IF(LISTER(1).LE.2)GO TO 104
C                     PREPARING TO STEP THROUGH DIRECTORIES
      JCDUM2=3
      JCDUM3=0
      JCLOAD(15)=1
      JCLOAD(27)=2
      JCDUM1=LISTER(1)-2
C                     WORKING THROUGH DIRECTORIES
      DO 8 I=1,JCDUM1
      IF(JCSORF.EQ.-1)GO TO 8
      IF(JCDUM2.NE.2)GO TO 4
C                     ADD NEW FILE TO END OF RECORD
      M1=JCLOAD(5)
      JCCORE(M1)=JCLOAD(21)
      JCLOAD(27)=JCLOAD(21)
C                     SAVING ADDRESS OF POINTER
    4 JCDUM3=JCLOAD(5)
C                     COMPUTING ABSOLUTE ADDRESS OF SELECTED FILE
    5 CALL C4ADR3(JCSPUR,JCLOAD,JCSORF,1)
      IF(JCSORF.EQ.-1)GO TO 8
      IF(JCDUM2.NE.2)GO TO 6
      JCDUM4=JCSPUR(4)-JCLOAD(7)+1
      IF(JCDUM4.GE.JCLOAD(22))GO TO 6
C                     INSUFFICIENT SPACE FOR PRE-FILE PARAMETER SET
      JCLOAD(27)=JCLOAD(27)+JCDUM4
      JCCORE(JCDUM3)=JCLOAD(27)
      JCDUM2=1
      JCLOAD(15)=2
      GO TO 5
C                     PREPARING TO COMPUTE ADDRESS OF FILE ENTRY
    6 JCLOAD(11)=JCLOAD(5)
      IF(I.GT.1)GO TO 13
      JCLOAD(28)=JCLOAD(11)-1
      M1=JCLOAD(28)
      JCLOAD(21)=JCCORE(M1)
      IF(JCLOAD(21).LE.1)JCLOAD(21)=2
      IF(JCLOAD(21).EQ.2)JCDUM2=1
   13 JCLOAD(8)=0
      JCLOAD(15)=3
      IF(JCDUM2.EQ.3)GO TO 7
C                     INITIALIZING PRE-FILE PARAMETER SET
      CALL TOCELL(JCSPUR,JCLOAD(11),JCFINE(I),JCLOAD(16),1)
      JCLOAD(9)=JCFINE(I)
      JCLOAD(10)=1
      IF(I.EQ.JCDUM1)JCLOAD(10)=JCFINE(I+1)
      CALL TOCELL(JCSPUR,JCLOAD(11),JCLOAD(10),JCLOAD(23),2)
      JCLOAD(21)=JCLOAD(27)+JCLOAD(22)+JCLOAD(9)*JCLOAD(10)
      JCLOAD(15)=4
C                     COMPUTING ABSOLUTE ADDRESS OF FILE ENTRY
    7 IF((LISTER(I+2).LT.0).OR.((LISTER(I+2).EQ.0).AND.(JCDUM1.GT.I)))
     1 CALL SYSERR(-2,JCLOAD(11))
      JCLOAD(12)=LISTER(I+2)
      CALL C4ADR2(JCSPUR,JCLOAD,JCSORF,2)
      IF((I.EQ.JCDUM1).OR.(JCSORF.EQ.-1))GO TO 8
C                     RETRIEVING RELATIVE ADDRESS FROM DIRECTORY
      M1=JCLOAD(5)
      JCLOAD(27)=JCCORE(M1)
      JCDUM2=3
      IF(JCLOAD(27).GT.0)GO TO 71
      JCDUM2=2
      IF(JCLOAD(14).GT.0)GO TO 71
```

```
      JCSORF=-JCSORF
      IF(JCSORF.EQ.0)CALL SYSERR(-4,JCLOAD(11))
   71 JCLOAD(15)=2
    8 CONTINUE
      IF(JCSORF.EQ.-1)GO TO 104
      M1=JCLOAD(28)
      JCCORE(M1)=JCLOAD(21)
      IF(LISTER(JCOUM1+2).LE.0)GO TO 104
C                   PREPARING LOESS AND MODEL
    9 MODEL(1)=-1
      LOESS(1)=-1
      IF(LOCAL(1).LE.0)GO TO 104
      MODEL(1)=1
      JCOUM1=3*(IODINE-1)+2
      M1=JCSPUR(5)+20
      JCLOAD(13)=JCCORE(M1)+4*(LOCAL(JCOUM1)-1)+1
      MODEL(2)=LOCAL(JCOUM1)
      MODEL(3)=LOCAL(JCOUM1+1)
      M1=JCLOAD(13)
      JCOUM2=JCSPUR(4)-JCLOAD(7)-JCCORE(M1)-MODEL(3)+3
      MODEL(4)=LOCAL(JCOUM1+2)
      IF(MODEL(4).GT.0)GO TO 10
      MODEL(4)=JCCORE(M1+1)
      IF(MODEL(4).EQ.0)MODEL(4)=JCLOAD(10)
      MODEL(4)=MODEL(4)-JCCORE(M1)+1
   10 JCOUM3=JCLOAD(10)-JCCORE(M1)-MODEL(4)+1
      IF((JCOUM3.LT.0).AND.(JCLOAD(29).NE.-1))CALL SYSERR(-3,JCLOAD(5))
      JCOUM3=MODEL(4)-MODEL(3)+1
      IF(JCOUM2.GE.JCOUM3)GO TO 104
C                   PARTITIONED FILE ENTRY
      IF(JCOUM2.GT.0)GO TO 11
      MODEL(1)=-1
      LOESS(3)=1-JCOUM2
      GO TO 12
   11 LOESS(3)=1
   12 LOESS(1)=1
      LOESS(2)=MODEL(2)
      LOESS(4)=MODEL(4)-JCOUM2-MODEL(3)+1
      MODEL(4)=MODEL(3)+JCOUM2-1
C                   EXIT
  104 CALL SYSEXT
      RETURN
      END
```

```
C
C*****************************************************************
C                              **********
C                              * C4ADR1 *
C                              **********
C
C       PURPOSE
C           C4ADR1 PROVIDES THE CALLING PROGRAM WITH THE ABSOLUTE
C           ADDRESS OF A POINTER WITHIN A CATALOGUE OR REGISTER
C
C       USAGE
C           CALL C4ADR1(JCSPUR,JCLOAD,JCSORF,JCCLNO)
C
C       DATA FORMAT
C           N/A
C
C       DESCRIPTION OF PARAMETERS
C           JCSPUR = A VECTOR DESCRIBED IN THE C4ADRS DOCUMENTATION
C           JCLOAD = A VECTOR DESCRIBED IN THE C4ADRS DOCUMENTATION
C           JCSORF = SEE FETCH
C           JCCLNO = A USER ASSIGNED CALL STATEMENT ID. PARAMETER
C
C       REMARKS
C           THE NUMBER REPRESENTED BY JCTCIA MUST BE SUPPLIED IN THE
C           COMMON STATEMENT FOR SOME MACHINES.
C           THIS ROUTINE IS NOT INTENDED FOR DIRECT USER CALL.
C
C       SUBROUTINE AND FUNCTION SUBPROGRAMS REQUIRED
C           C4STUP
C           SYSENT
C           SYSERR
C           SYSEXT
C
C       ERROR CODES FOR THIS PROGRAM
C           CODE FATAL  ERROR(DATA PROVIDED)
C           1    F      BACKWARD LINK FROM POINTER (JCLOAD(1))
C           2    F      BACKWARD LINK BEYOND FIRST SEGMENT (JCLOAD(2))
C           3    F      SEGMENT CONTAINING POINTER TO DESIRED
C                       INFORMATION NOT IN CORE (ADDRESS OF LAST
C                       POINTER)
C           4    F      NO LINK FROM POINTER (JCLOAD(1))
C           5    F      BACKWARD LINK TO DESIRED SEGMENT CANNOT BE
C                       ACCOMPLISHED (ADDRESS OF LAST POINTER)
C
C       METHOD
C           TWO MODES OF OPERATION ARE ALLOWED, JCLOAD(15)=1 AND
C           JCLOAD(15)=2.  THE FORMER REQUIRES COMPONENTS 1,4,12,14 AND
C           15.  FROM JCLOAD(4) AND JCLOAD(12), THE NUMBER OF SEGMENTS
C           TO ADVANCE AND THE CORRECT ENTRY NUMBER WITHIN THAT SEGMENT
C           ARE COMPUTED.  JCLOAD COMPONENTS 1,2,3,4 AND 12 CARRY THE
C           RESULT OF THE ADDRESS CALCULATION.  WHEN JCLOAD(15)=2, THE
C           ADDRESS CALCULATION BEGINS WITH THE REGISTER (OR CATALOGUE)
C           SEGMENT SPECIFIED BY JCLOAD(2) AND JCLOAD(3).
C
C                              **********
C
      SUBROUTINE C4ADR1(JCSPUR,JCLOAD,JCSORF,JCCLNO)
      COMMON/ALLOC/JCCORE(1)
      DIMENSION JCSPUR(1),JCLOAD(15
C                 ENTER
      CALL SYSENT(1,2,1,6,JCCLNO)
C                   COMPUTING NUMBER OF SEGMENTS TO ADVANCE
```

```
      JCDUM6=JCSPUR(4)
      JCSPUR(4)=JCCORE(3)
      JCDUM1=(JCLOAD(4)-1)/(JCCORE(3)-2)
      JCDUM2=JCDUM1+JCLOAD(12)+2-JCLOAD(15)
      JCDUM3=IABS(JCDUM2)
      IF(JCDUM3.NE.0)GO TO 1
      IF(JCLOAD(15).NE.1)GO TO 5
      JCSCRF=-JCSCRF
      IF(JCSCRF.EQ.-1)GO TO 6
      CALL SYSERR(-4,JCLOAD(1))
      GO TO 5
    1 JCDUM2=(JCDUM3-JCDUM2)/(2*JCDUM3)
      IF(JCDUM2.EQ.0)GO TO 2
      IF(JCLOAD(15).EQ.1)CALL SYSERR(-1,JCLOAD(1))
      IF(JCDUM3.GE.JCLOAD(3))CALL SYSERR(-2,JCLOAD(2))
    2 JCLOAD(12)=JCDUM3
      JCDUM5=JCLOAD(1)
      IF(JCLOAD(15).EQ.2)JCDUM5=JCLOAD(2)
C                   ACCESSING PROPER SEGMENT
      DO 4 I=1,JCDUM3
      IF(JCSCRF.EQ.-1)GO TO 4
      JCLOAD(2)=JCCORE(JCDUM5+JCDUM2)
      IF(JCLOAD(2).GT.0)GO TO 3
      JCSCRF=-JCSCRF
      IF(JCSCRF.EQ.-1)GO TO 6
      IF(JCLOAD(14).LT.0)CALL SYSERR(-3,JCDUM5)
      IF(JCDUM2.EQ.1)CALL SYSERR(-5,JCDUM5)
      JCLOAD(2)=JCDUM5
      CALL C4STUP(JCSPUR,JCLOAD,1)
      JCLOAD(2)=JCCORE(JCDUM5+JCDUM2)
    3 JCLOAD(3)=JCLOAD(3)+(JCLOAD(15)-1)+1-2*JCDUM2
      JCDUM5=JCLOAD(2)
      JCLOAD(12)=JCDUM3-I
    4 JCLOAD(15)=2
      IF(JCSCRF.EQ.-1)GO TO 6
C                   COMPUTING THE ENTRY NUMBER
    5 JCLOAD(4)=JCLOAD(4)-JCDUM1*(JCCORE(3)-2)
      JCLOAD(1)=JCLOAD(2)+JCLOAD(4)
      IF(JCLOAD(4).GT.0)JCLOAD(1)=JCLOAD(1)+1
C                   EXIT
    6 JCSPUR(4)=JCDUM6
      CALL SYSEXT
      RETURN
      END
```

```
C
C*********************************************************************
C                                **********
C                                * C4ADR2 *
C                                **********
C
C       PURPOSE
C           C4ADR2 PROVIDES THE CALLING PROGRAM WITH THE ABSOLUTE
C           ADDRESS OF A FILE ENTRY.
C
C       USAGE
C           CALL C4ADR2(JCSPUR,JCLOAD,JCSORF,JCCLNO)
C
C       DATA FORMAT
C           N/A
C
C       DESCRIPTION OF PARAMETERS
C           JCSPUR = A VECTOR DESCRIBED IN THE C4ADRS DOCUMENTATION
C           JCLOAD = A VECTOR DESCRIBED IN THE C4ADRS DOCUMENTATION
C           JCSORF = SEE FETCH
C           JCCLNO = A USER ASSIGNED CALL STATEMENT ID. PARAMETER
C
C       REMARKS
C           THE NUMBER REPRESENTED BY JCTCIA MUST BE SUPPLIED IN THE
C           COMMON STATEMENT FOR SOME MACHINES.
C           THIS ROUTINE IS NOT INTENDED FOR DIRECT USER CALL.
C
C       SUBROUTINE AND FUNCTION SUBPROGRAMS REQUIRED
C           C4ADR3
C           FRMCEL
C           SYSENT
C           SYSERR
C           SYSEXT
C           TOCELL
C
C       ERROR CODES FOR THIS PROGRAM
C           CODE FATAL  ERROR(DATA PROVIDED)
C           1    F      IMPROPER MODE (JCLOAD(11))
C           2    F      NEGATIVE FILE ENTRY NUMBER (JCLOAD(11))
C           3    F      DESIRED FILE ENTRY BEYOND LIMITS OF FILE
C                       (JCLOAD(11))
C           4    F      FILE ENTRY CANNOT BE ADDED TO FILE (JCLOAD(11))
C
C       METHOD
C           TWO MODES OF OPERATION ARE ALLOWED, JCLOAD(15)=3 AND
C           JCLOAD(15)=4.  JCLOAD COMPONENTS 1-7,11-12, AND 14-27 ARE
C           REQUIRED FOR MODE 3.  JCLOAD COMPONENTS 9 AND 10 ARE
C           RETRIVED FROM THE PRE-FILE PARAMETER SET.  THE ABSOLUTE
C           ADDRESS OF THE FILE ENTRY NUMBER SPECIFIED IN JCLOAD(12) IS
C           COMPUTED.  FOR JCLOAD(14) 0, ADDITIONAL FILE ENTRIES WILL
C           AUTOMATICALLY BE ADDED TO THE FILE PROVIDED THE FILE IS THE
C           LAST ON THE RECORD.  MODE 4 REQUIRES COMPONENTS 8,9, AND 10
C           AS WELL AS THOSE REQUIRED FOR MODE 3.  JCLOAD (12) IS THE
C           NUMBER OF FILE ENTRIES TO MOVE FROM THE CURRENT FILE ENTRY.
C
C                                **********
C
      SUBROUTINE C4ADR2(JCSPUR,JCLOAD,JCSORF,JCCLNO)
      COMMON/ALLOC/JCCORE(1)
      DIMENSION JCSPUR(1),JCLOAD(1)
C                 ENTER
      CALL SYSENT(1,2,1,7,JCCLNO)
```

```
C                   SELECTING PROPER MODE
      JCDUM2=JCLOAD(15)
      GO TO (1,1,2,3),JCDUM2
    1 CALL SYSERR(-1,JCLOAD(11))
C                   MODE = 3
    2 CALL FRMCEL(JCSPUR,JCLOAD(11),JCLOAD(10),JCLOAD(23),1)
      CALL FRMCEL(JCSPUR,JCLOAD(11),JCLOAD(9),JCLOAD(16),2)
      JCLOAD(8)=0
C                   MODE = 4
    3 IF(JCLOAD(12).EQ.0)GO TO 104
      JCDUM3=0
      JCDUM4=JCLOAD(12)
      JCDUM5=1
      IF(JCLOAD(8).GT.0)GO TO 4
      JCDUM3=JCLOAD(22)
      JCDUM5=0
      JCLOAD(12)=JCLOAD(12)-1
C                   COMPUTING THE FILE ENTRY NUMBER
    4 JCDUM6=JCLOAD(8)+JCDUM4
      IF(JCDUM6.GT.0)GO TO 5
      JCSORF=-JCSORF
      IF(JCSORF.EQ.-1)GO TO 104
      CALL SYSERR(-2,JCLOAD(11))
    5 IF(JCDUM6.LE.JCLOAD(9))GO TO 8
C                   FILE ENTRY BEYOND LAST FILE ENTRY
      IF(JCLOAD(14).GT.0)GO TO 6
      JCSORF=-JCSORF
      IF(JCSORF.EQ.-1)GO TO 104
      CALL SYSERR(-3,JCLOAD(11))
    6 JCDUM7=JCLOAD(27)+JCDUM3+(JCLOAD(9)-JCLOAD(8)+JCDUM5)*JCLOAD(10)
      IF(JCDUM7.EQ.JCLOAD(21))GO TO 7
C                   FILE NOT LAST IN RECORD
      CALL SYSERR(-4,JCLOAD(11))
C                   INCREASING SIZE OF FILE
    7 JCLOAD(21)=JCLOAD(21)+(JCDUM6-JCLOAD(9))*JCLOAD(10)
      JCLOAD(9)=JCDUM6
      CALL TOCELL(JCSPUR,JCLOAD(11),JCLOAD(9),JCLOAD(16),1)
C                   COMPUTING RELATIVE ADDRESS OF FILE ENTRY
    8 JCLOAD(27)=JCLOAD(27)+JCDUM3+JCLOAD(12)*JCLOAD(10)
      JCLOAD(8)=JCDUM6
C                   COMPUTING ABSOLUTE ADDRESS OF FILE ENTRY
      JCLOAD(15)=2
      CALL C4ADR3(JCSPUR,JCLOAD,JCSORF,1)
C                   EXIT
  104 CALL SYSEXT
      RETURN
      END
```

```
C
C*****************************************************************
C                              **********
C                              * C4ADR3 *
C                              **********
C
C       PURPOSE
C           C4ADR3 CONVERTS THE RELATIVE ADDRESS FROM JCLOAD(27) INTO
C           AN ABSOLUTE ADDRESS, UPDATING JCLOAD COMPONENTS 1-7.
C
C       USAGE
C           CALL C4ADR3(JCSPUR,JCLOAD,JCSORF,JCCLNO)
C
C       DATA FORMAT
C           N/A
C
C       DESCRIPTION OF PARAMETERS
C           JCSPUR = A VECTOR DESCRIBED IN THE C4ADRS DOCUMENTATION
C           JCLOAD = A VECTOR DESCRIBED IN THE C4ADRS DOCUMENTATION
C           JCSORF = SEE FETCH
C           JCCLNO = A USER ASSIGNED CALL STATEMENT ID. PARAMETER
C
C       REMARKS
C           THE NUMBER REPRESENTED BY JCTCIA MUST BE SUPPLIED IN THE
C           COMMON STATEMENT FOR SOME MACHINES.
C           THIS ROUTINE IS NOT INTENDED FOR DIRECT USER CALL.
C
C       SUBROUTINE AND FUNCTION SUBPROGRAMS REQUIRED
C           C4ADR1
C           C4STUP
C           SYSENT
C           SYSEXT
C
C       ERROR CODES FOR THIS PROGRAM
C           1    F       RECORD SEGMENT ABSENT (ADDRESS OF POINTER)
C
C       METHOD
C           SELF EXPLANATORY
C
C                              **********
C
      SUBROUTINE C4ADR3(JCSPUR,JCLOAD,JCSORF,JCCLNO)
      COMMON/ALLOC/JCCORE(1)
      DIMENSION JCSPUR(1),JCLOAD(1)
C                 ENTER
      CALL SYSENT(1,2,1,8,JCCLNO)
C                 COMPUTING RECORD AND REGISTER SEGMENT NUMBERS
      JCDUM1=(JCLOAD(27)-1)/JCSPUR(4)
      JCDUM2=JCDUM1/(JCSPUR(4)-2)+1
      JCDUM3=JCLOAD(3)
      IF(JCLOAD(15).EQ.1)JCDUM3=1
    1 JCLOAD(12)=JCDUM2-JCDUM3
      JCLOAD(4)=JCDUM1-(JCDUM2-1)*(JCCORE(3)-2)+1
C                 ACCESSING POINTER TO RECORD SEGMENT
      CALL C4ADR1(JCSPUR,JCLOAD,JCSORF,1)
      IF(JCSORF.EQ.-1)GO TO 2
      JCLOAD(15)=3
C                 ACCESSING RECORD-SEGMENT
      JCDUM2=JCLOAD(1)
      IF(JCCORE(JCDUM2).GT.0)GO TO 3
      IF(JCCORE(JCDUM2).LT.0)GO TO 4
      IF(JCLOAD(14).GT.0)GO TO 4
```

```
      JCSORF=-JCSORF
      IF(JCSORF.EQ.-1)GO TO 2
      CALL SYSERR(-1,JCDUM2)
    4 CALL C4STUP(JCSPUR,JCLOAD,1)
    3 JCLOAD(6)=JCCORE(JCDUM2)
C                   COMPUTING ABSOLUTE ADDRESS
      JCLOAD(7)=JCLOAD(27)-JCDUM1+JCSPUR(4)
      JCLOAD(5)=JCLOAD(6)+JCLOAD(7)-1
C                   EXIT
    2 CALL SYSEXT
      RETURN
      END
```

```
C
C*****************************************************************
C                                 **********
C                                 * C4NEXT *
C                                 **********
C
C       PURPOSE
C           C4NEXT RETURNS IN JCLOAD(5) THE ABSOLUTE ADDRESS OF THE
C           NEXT RECORD SEGMENT.
C
C       USAGE
C           CALL C4NEXT(JCSPUR,LOESS,MODEL,JCLOAD,JCCLNO)
C
C       DATA FORMAT
C           N/A
C
C       DESCRIPTION OF PARAMETERS
C           JCSPUR = A VECTOR DESCRIBED IN THE C4ADRS DOCUMENTATION
C           LOESS  = A VECTOR DESCRIBED IN THE C4ADRS DOCUMENTATION
C           MODEL  = A VECTOR DESCRIBED IN THE C4ADRS DOCUMENTATION
C           JCLOAD = A VECTOR DESCRIBED IN THE C4ADRS DOCUMENTATION
C           JCCLNO = A USER ASSIGNED CALL STATEMENT ID. PARAMETER
C
C       REMARKS
C           THE NUMBER REPRESENTED BY JCTCIA MUST BE SUPPLIED IN THE
C           COMMON STATEMENT FOR SOME MACHINES.
C           THIS ROUTINE IS NOT INTENDED FOR DIRECT USER CALL.
C
C       SUBROUTINE AND FUNCTION SUBPROGRAMS REQUIRED
C           C4ADR1
C           C4STUP
C           J5TST1
C           SYSENT
C           SYSERR
C           SYSEXT
C
C       ERROR CODES FOR THIS PROGRAM
C           CODE FATAL  ERROR(DATA PROVIDED)
C           1    F      RECORD SEGMENT NOT ASSIGNED (JCLOAD(1))
C
C       METHOD
C           ADDRESSING NEXT RECORD SEGMENT IS SELF-EXPLANATORY.  FROM
C           THE LOESS PASSED TO C4NEXT, THE MODEL FOR THE NEW FILE
C           ENTRY PARTITION IS CALCULATED.  LOESS IS THEN UPDATED.
C
C                                 **********
C
      SUBROUTINE C4NEXT(JCSPUR,LOESS,MODEL,JCLOAD,JCCLNO)
      COMMON/ALLOC/JCCORE(1)
      DIMENSION JCSPUR(1),LOESS(1),MODEL(1),JCLOAD(1)
C                 ENTER
      CALL SYSENT(1,2,1,9,JCCLNO)
C                 ADDRESSING NEXT REGISTER SEGMENT ENTRY
      JCLOAD(4)=JCLOAD(4)+1
      JCLOAD(12)=0
      JCLOAD(15)=2
      CALL C4ADR1(JCSPUR,JCLOAD,0,1)
      M1=JCLOAD(1)
      JCLOAD(5)=JCCORE(M1)
      IF(JCLOAD(5).GT.0)GO TO 2
C                 RECORD NOT IN CORE
      IF(JCLOAD(5).LT.0)GO TO 1
```

```
      IF(JCLOAD(14).LT.0)CALL SYSERR(-1,JCLOAD(1))
C                     ACQUIRING RECORD SEGMENT
    1 JCLOAD(15)=3
      CALL C4STUP(JCSPUR,JCLOAD,1)
      M1=JCLOAD(1)
      JCLOAD(5)=JCCORE(M1)
C                     TESTING RECORD SEGMENT ADDRESS
    2 CALL J5TST1(JCLOAD(5),1)
C                     UPDATING MODEL AND LOESS
      DO 3 I=1,3
    3 MODEL(I)=LOESS(I)
      IF(LOESS(4).GT.JCSPUR(4))GO TO 4
C                     LAST PARTITION
      LOESS(1)=-1
      MODEL(4)=LOESS(4)
      GO TO 104
C                     NOT LAST PARTITION
    4 LOESS(4)=LOESS(4)-JCSPUR(4)
      IF(MODEL(3).GT.JCSPUR(4))GO TO 5
      MODEL(4)=JCSPUR(4)
      LOESS(3)=1
      GO TO 104
    5 MODEL(1)=-1
      LOESS(3)=LOESS(3)-JCSPUR(4)
C                     EXIT
  104 CALL SYSEXT
      RETURN
      END
```

```
C
C*****************************************************************
C                              **********
C                              * C4STUP *
C                              **********
C
C       PURPOSE
C           C4STUP ADDS EMPTY CORE SEGMENTS TO A CATALOGUE, REGISTER OR
C           RECORD.
C
C       USAGE
C           CALL C4STUP(JCSPUR,JCLOAD,JCCLNO)
C
C       DATA FORMAT
C           N/A
C
C       DESCRIPTION OF PARAMETERS
C           JCSPUR = A VECTOR DESCRIBED IN THE C4ADRS DOCUMENTATION
C           JCLOAD = A VECTOR DESCRIBED IN THE C4ADRS DOCUMENTATION
C           JCCLNO = A USER ASSIGNED CALL STATEMENT ID. PARAMETER
C
C       REMARKS
C           THE NUMBER REPRESENTED BY JCTCIA MUST BE SUPPLIED IN THE
C           COMMON STATEMENT FOR SOME MACHINES.
C           THIS ROUTINE IS NOT INTENTED FOR DIRECT USER CALL.
C
C       SUBROUTINE AND FUNCTION SUBPROGRAMS REQUIRED
C           NEWCEL
C           SUPPLY
C           SYSENT
C           SYSERR
C           SYSEXT
C
C       ERROR CODES FOR THIS PROGRAM
C           CODE FATAL  ERROR(DATA PROVIDED)
C           1    NF     NOT AN EMPTY LOCATION (ADDRESS OF POINTER)
C
C       METHOD
C           THREE MODES OF OPERATION ARE POSSIBLE.  JCLOAD(15)=1 AND
C           JCLOAD(15)=2 CORRESPOND TO THE TWO MODES FOR C4ADR1.  THE
C           THIRD MODE, JCLOAD(15)=3, IS FOR THE ADDITION OF A RECORD
C           SEGMENT TO A RECORD.  FOR MODES 1 AND 3, JCLOAD(1) IS THE
C           ADDRESS OF THE POINTER FOR WHICH CORE IS REQUESTED.  IN
C           MODE 2, CORE SEGMENT(S) ARE LINKED TO THE SEGMENT NAMED BY
C           JCLOAD(2).  C4STUP MAKES NO CHANGES IN JCLOAD.  JCLOAD(20)
C           IS THE BASE ERROR CODE.  JCLOAD(12)=NUMBER OF SEGMENTS.
C
C                              **********
C
      SUBROUTINE C4STUP(JCSPUR,JCLOAD,JCCLNO)
      COMMON/ALLOC/JCCORE(1)
      DIMENSION JCSPUR(1),JCLOAD(1)
C                 ENTER
      CALL SYSENT(1,2,1,IO,JCCLNO)
C                 SELECTING PROPER MODE
      JCDUM8=JCSPUR(2)
      IF(JCLOAD(15).NE.3)JCSPUR(2)=0
      JCDUM9=JCSPUR(3)
      JCDUM1=JCLOAD(15)
      JCDUM2=JCLOAD(1)
      JCDU10=JCCORE(JCDUM2)
      JCDUM4=JCLOAD(12)
```

```
      GO TO (3,1,2),JCDUM1
    1 JCDUM2=JCLOAD(2)
      JCDUM3=JCDUM2
      GO TO 3
    2 JCDUM4=1
C                    VACANCY TEST
    3 IF(JCCORE(JCDUM2).LE.0)GO TO 4
      CALL SYSERR(1,JCDUM2)
      GO TO 104
    4 JCSPUR(3)=JCDUM4
      DO 8 I=1,JCDUM4
C                    SEGMENT ACQUISITION
      CALL NEWCEL(JCSPUR,JCDUM5,1)
C                    SETTING LINKS
      JCCORE(JCDUM2)=JCDUM5
      GO TO (7,6,8),JCDUM1
    6 JCDUM6=JCDUM5+1
      JCCORE(JCDUM6)=JCDUM3
    7 JCDUM2=JCDUM5
      JCDUM3=JCDUM5
    8 CONTINUE
      JCCORE(JCDUM5)=0
      IF((JCDUM1.NE.3).OR.(JCDU10.GE.0))GO TO 104
      JCDUM1=IABS(JCDU10)
      CALL SUPPLY(JCDUM1,JCDUM5,JCSPUR(4),1)
  104 IF(JCLOAD(15).NE.3)JCSPUR(2)=JCDUM8
      JCSPUR(3)=JCDUM9
C                    EXIT
      CALL SYSEXT
      RETURN
      END
```

```
C
C*****************************************************************
C                              **********
C                              * C4TRAN *
C                              **********
C
C       PURPOSE
C           C4TRAN PERFORMS THE DATA TRANSFER FOR FETCH AND STORE.
C
C       USAGE
C           CALL C4TRAN(JCSPUR,LISTER,LION,LOCAL,JCFINE,JCLOAD,JCSORF,
C           JCCLNO)
C
C       DATA FORMAT
C           N/A
C
C       DESCRIPTION OF PARAMETERS
C           JCSPUR = A VECTOR DESCRIBED IN THE C4ADRS DOCUMENTATION
C           LISTER = A VECTOR DESCRIBED IN THE C4ADRS DOCUMENTATION
C           LION   = A VECTOR WHICH CONTAINS THE DATA ASSOCIATED WITH
C                    A TRANSFER OPERATION.  THE DATA ITEMS ARE PLACED
C                    OR REMOVED FROM LION IN THE ORDER DICTATED BY
C                    LOCAL.
C           LOCAL  = A VECTOR DESCRIBED IN THE C4ADRS DOCUMENTATION
C           JCFINE = A VECTOR DESCRIBED IN THE C4ADRS DOCUMENTATION
C           JCLOAD = A VECTOR DESCRIBED IN THE C4ADRS DOCUMENTATION
C           JCSORF = SEE FETCH
C           JCCLNO = A USER SPECIFIED CALL STATEMENT ID. NUMBER
C
C       REMARKS
C           THE NUMBER REPRESENTED BY JCTCIA MUST BE SUPPLIED IN THE
C           COMMON STATEMENT FOR SOME MACHINES.
C           THIS PROGRAM IS NOT INTENDED FOR DIRECT USER CALL.
C
C       SUBROUTINE AND FUNCTION SUBPROGRAMS REQUIRED
C           C4ADRS
C           C4NEXT
C           FRMCEL
C           SYSENT
C           SYSEXT
C           TOCELL
C
C       ERROR CODES FOR THIS PROGRAM
C           NONE
C
C       METHOD
C           IF JCLOAD(14)=-1, FRMCEL IS EMPLOYED.  IF JCLOAD(14)=1,
C           TOCELL IS EMPLOYED.
C
C                              **********
C
      SUBROUTINE C4TRAN(JCSPUR,LISTER,LION,LOCAL,JCFINE,JCLOAD,JCSORF,
     1 JCCLNO)
      COMMON/ALLOC/JCCORE(1)
      DIMENSION JCSPUR(1),LISTER(1),LION(1),LOCAL(1),JCFINE(1),JCLOAD(1)
     1 ,LOESS(4),MODEL(4),JCDUMY(5)
C                   ENTER
      CALL SYSENT(1,2,1,11,JCCLNO)
C                   PREPARING FOR DATA TRANSFER
      JCDUM1=LOCAL(1)
      JCDUM3=LISTER(1)
      JCDUM6=LISTER(2)
```

```
      JCDUM7=JCDUM1
      IF(JCDUM1.GT.0)GO TO 3
C                     EITHER NO DATA TRANSFER OR EVERYTHING IN FILE ENTRY
      JCDUM1=1
      IF(LOCAL(1).LT.0)GO TO 3
C                     ENTIRE FILE ENTRY WORD FOR WORD
      IF(LISTER(1).LE.1)GO TO 1
C                     RETRIEVING NUMBER OF WORDS PER FILE ENTRY
      LOCAL(1)=-1
      CALL C4ADRS(JCSPUR,LISTER,LOCAL,LOESS,MODEL,1,JCFINE,JCLOAD,JCSORF
     1 ,1)
      IF(JCSORF.EQ.-1)GO TO 16
C                     RESETTING LISTER TO AVOID REPEATING ADDRESS CALC.
      LISTER(1)=1
      LISTER(2)=0
C                     SETTING UP LOCAL TO TRANSFER EACH FILE ENTRY WORD
    1 DO 2 I=1,3
    2 LOCAL(I)=1
      LOCAL(4)=JCLOAD(10)
C                     SERVICING DATA ITEMS IN LOCAL
    3 JCDUM2=1
      DO 15 I=1,JCDUM1
      IF(JCSORF.EQ.-1)GO TO 15
      JCDUM4=1
C                     ADDRESSING FILE ENTRY
      CALL C4ADRS(JCSPUR,LISTER,LOCAL,LOESS,MODEL,I,JCFINE,JCLOAD,JCSORF
     1 ,2)
      IF(JCSORF.EQ.-1)GO TO 15
    4 IF(MODEL(1).LE.0)GO TO 7
C                     TRANSFERRING DATA ITEM
      IF(JCLOAD(14).LT.0)GO TO 5
      CALL TOCELL(JCSPUR,JCLOAD(5),LION(JCDUM2),MODEL,1)
      GO TO 6
    5 CALL FRMCEL(JCSPUR,JCLOAD(5),LION(JCDUM2),MODEL,1)
    6 JCDUM2=JCDUM2+MODEL(4)-MODEL(3)+1
    7 IF(LOESS(1).LE.0)GO TO 11
C                     PARTITIONED FILE ENTRY
      GO TO (8,10),JCDUM4
C                     SECOND PARTITION
    8 JCDUM4=2
      M1=JCLOAD(13)
      JCDUM5=JCCORE(M1)
      JCCORE(M1)=1
      DO 9 J=1,5
    9 JCDUMY(J)=JCLOAD(J)
C                     ADDRESSING NEXT FILE ENTRY PARTITION
   10 CALL C4NEXT(JCSPUR,LOESS,MODEL,JCLOAD,1)
      GO TO 4
C                     END OF TRANSFER OPERATION
   11 GO TO (14,12),JCDUM4
   12 DO 13 J=1,5
   13 JCLOAD(J)=JCDUMY(J)
      M1=JCLOAD(13)
      JCCORE(M1)=JCDUM5
   14 LISTER(1)=1
   15 LISTER(2)=0
   16 LISTER(1)=JCDUM3
      LISTER(2)=JCDUM6
      LOCAL(1)=JCDUM7
C                     EXIT
      CALL SYSEXT
      RETURN
```

```
END
```

```
C
C*****************************************************************
C                              *********
C                              * FETCH *
C                              *********
C
C       PURPOSE
C           THE PURPOSE OF FETCH IS TO RETRIEVE DATA ITEMS FROM A FILE
C           ENTRY.
C
C       USAGE
C           CALL FETCH(JCSPUR,LISTER,LION,LOCAL,JCFINE,JCLOAD,JCSORF,
C           JCCLNO)
C
C       DATA FORMAT
C           N/A
C
C       DESCRIPTION OF PARAMETERS
C           JCSPUR = A VECTOR DESCRIBED IN THE C4ADRS DOCUMENTATION
C           LISTER = A VECTOR DESCRIBED IN THE C4ADRS DOCUMENTATION
C           LION   = A VECTOR DESCRIBED IN THE C4TRAN DOCUMENTATION
C           LOCAL  = A VECTOR DESCRIBED IN THE C4ADRS DOCUMENTATION
C           JCFINE = A VECTOR DESCRIBED IN THE C4ADRS DOCUMENTATION
C           JCLOAD = A VECTOR DESCRIBED IN THE C4ADRS DOCUMENTATION
C           JCSORF = +1 OR -1 AS THE FETCH OPERATION IS OR IS NOT
C                    SUCCESSFUL WHEN JCSORF IS PASSED AS +1.  IF THE
C                    TEST FOR SUCCESS OR FAILURE IS NOT DESIRED, JCSORF
C                    IS PASSED AS 0.
C           JCCLNO = A USER ASSIGNED CALL STATEMENT ID. PARAMETER
C
C       REMARKS
C           THE NUMBER REPRESENTED BY JCTCIA MUST BE SUPPLIED IN THE
C           COMMON STATEMENT FOR SOME MACHINES.
C           JCLOAD(29) MUST NOT EQUAL -1 UNLESS VIOLATION OF FILE ENTRY
C           BOUNDARIES IS TO BE PERMITTED.
C           CARE MUST BE EXERCISED IN USING THE SECOND MODE OF
C           OPERATION (SEE METHOD) TO INSURE THAT THE JCLOAD VECTOR HAS
C           BEEN UNALTERED SINCE THE STORE OR FETCH CALL ADDRESSING THE
C           CURRENT FILE ENTRY.
C           THE LIST TYPE SPECIFIED BY JCSPUR(5) MUST BE FORWARD-
C           BACKWARD.
C           THIS PROGRAM, FIND AND STORE ARE THE ONLY REMOTE ROUTINES
C           RECOMMENDED FOR DIRECT USER CALL.
C           JCLOAD(30) MUST CONTAIN THE ADDRESS OF THE POINTER TO THE
C           CATALOGUE.
C
C       SUBROUTINE AND FUNCTION SUBPROGRAMS REQUIRED
C           C4TRAN
C           SYSENT
C           SYSEXT
C
C       ERROR CODES FOR THIS PROGRAM
C           NONE
C
C       METHOD
C           LISTER MAY BE USED TO ADDRESS THE DESIRED FILE ENTRY VIA
C           CATALOGUE AND/OR DIRECTORY SELECTIONS, OR MAY SPECIFY THE
C           FILE ENTRY VIA LISTER(1)=1 AND LISTER(2)=THE NUMBER OF FILE
C           ENTRIES TO MOVE FROM THE CURRENT FILE ENTRY.  IF LOCAL(1)
C           IS -1, ONLY THE ADDRESSING OPERATION IS PERFORMED.  IF
C           LOCAL(1) IS 0 OR LARGER, FETCH PERFORMS AS WOULD FRMCEL.
C           A RECORD SEGMENT MAY BE RETRIEVED FROM OFF-LINE STORAGE IF
```

```
C           REQUIRED.  NO OTHER ABSENCES OF SEGMENTS FROM ALLOCATABLE
C           CORE ARE TOLERATED.
C
C                              **********
C
      SUBROUTINE FETCH(JCSPUR,LISTER,LION,LOCAL,JCFINE,JCLOAD,JCSORF,
     1 JCCLNO)
      COMMON/ALLOC/JCCORE(1)
      DIMENSION JCSPUR(1),LISTER(1),LION(1),LOCAL(1),JCFINE(1),JCLOAD(1)
C                 ENTER
      CALL SYSENT(1,2,1,1,JCCLNO)
C                 PREPARING FOR DATA TRANSFER
      JCLOAD(14)=-1
C                 TRANSFERING DATA
      CALL C4TRAN(JCSPUR,LISTER,LION,LOCAL,JCFINE,JCLOAD,JCSORF,1)
C                 EXIT
      CALL SYSEXT
      RETURN
      END
```

```
C
C*****************************************************************
C                              ********
C                              * FIND *
C                              ********
C
C       PURPOSE
C           FIND SEARCHES A FILE IN A SPECIFIED DIRECTION FOR A FILE
C           ENTRY CONTAINING A SPECIFIED PATTERN OF DATA VALUES.
C
C       USAGE
C           CALL FIND(JCSPUR,LISTER,LION,LOCAL,JCFINE,JCLOAD,JCDOFS,
C           JCCLNO)
C
C       DATA FORMAT
C           N/A
C
C       DESCRIPTION OF PARAMETERS
C           JCSPUR = A VECTOR DESCRIBED IN THE C4ADRS DOCUMENTATION
C           LISTER = A VECTOR DESCRIBED IN THE C4ADRS DOCUMENTATION
C           LION   = A VECTOR DESCRIBED IN THE C4TRAN DOCUMENTATION
C           LOCAL  = A VECTOR DESCRIBED IN THE C4ADRS DOCUMENTATION
C           JCFINE = A VECTOR DESCRIBED IN THE C4ADRS DOCUMENTATION
C           JCLOAD = A VECTOR DESCRIBED IN THE C4ADRS DOCUMENTATION
C           JCDOFS = A PARAMETER EQUAL IN MAGNITUDE TO 1, 2, OR 3 AS
C                    RESPICTIVELY IT IS DESIRED THE SEARCH PROCEED IN
C                    THE FORWARD DIRECTION ONLY, BACKWARD DIRECTION
C                    ONLY OR THE SEARCH IS TO ENCOMPASS THE ENTIRE
C                    FILE.  IF JCFOFS IS NEGATIVE, THEN THE SEARCH IS
C                    SUCCESSFUL IF ANY DATA TYPE DESCRIBED IN THE LOCAL
C                    VECTOR MATCHES THE CORRESPONDING VALUES INDICATED
C                    IN LION.  WHEN JCDOFS IS NEGATIVE, THE PARTICULAR
C                    DATA TYPE FOUND TO MATCH IS RETURNED TO THE USER
C                    VIA THE JCDOFS PARAMETER.  FOR A POSITIVE JCDOFS
C                    ALL DATA TYPES IN LOCAL MUST MATCH.  JCDOFS IS SET
C                    EQUAL TO ZERO IN THE CASE OF FAILURE.
C           JCCLNO = A USER ASSIGNED CALL STATEMENT ID. PARAMETER
C
C       REMARKS
C           THE NUMBER REPRESENTED BY JCTCIA MUST BE SUPPLIED IN THE
C           COMMON STATEMENT FOR SOME MACHINES.
C           JCLOAD(29) MUST NOT EQUAL -1 UNLESS VIOLATION OF FILE ENTRY
C           BOUNDARIES IS TO BE PERMITTED.
C           CARE MUST BE EXERCISED IN USING THE SECOND MODE OF
C           OPERATION (SEE METHOD) TO INSURE THAT THE JCLOAD VECTOR HAS
C           BEEN UNALTERED SINCE THE FIND, FETCH OR STORE CALL
C           ADDRESSING THE CURRENT FILE ENTRY.
C           THE LIST TYPE SPECIFIED BY JCSPUR(5) MUST BE FORWARD-
C           BACKWARD.
C           THIS PROGRAM, FETCH AND STORE ARE THE ONLY REMOTE ROUTINES
C           RECOMMENDED FOR DIRECT USER CALL.
C           JCLOAD(30) MUST CONTAIN THE ADDRESS OF THE POINTER TO THE
C           CATALOGUE.
C           THE FILE ENTRY NUMBER OF THE FILE ENTRY CONTAINING THE DATA
C           PATTERN IS RECORDED IN JCLOAD(8), AND ITS ABSOLUTE ADDRESS
C           IN JCLOAD(5).  IF JCDOFS IS 0, THE SEARCH HAS FAILED.
C
C       SUBROUTINE AND FUNCTION SUBPROGRAMS REQUIRED
C           C4TRAN
C           SYSENT
C           SYSERR
C           SYSEXT
```

```
C
C       ERROR CODES FOR THIS PROGRAM
C           CODE FATAL  ERROR(DATA PROVIDED)
C           1    NF     LOCAL(1) = -1 (JCLOAD(5))
C
C       METHOD
C           LISTER MAY BE USED TO ADDRESS THE DESIRED FILE ENTRY VIA
C           CATALOGUE AND/OR DIRECTORY SELECTIONS, OR MAY SPECIFY THE
C           FILE ENTRY VIA LISTER(1)=1 AND LISTER(2)=THE NUMBER OF FILE
C           ENTRIES TO MOVE FROM THE CURRENT FILE ENTRY.  FIND BEHAVES
C           ESSENTIALLY AS DOES LOCATE.  A RECORD SEGMENT MAY BE
C           RETRIEVED FROM OFF-LINE STORAGE.  NO OTHER ABSENCES OF
C           SEGMENTS FROM ALLOCATABLE CORE ARE TOLERATED.
C
C                                  **********
C
      SUBROUTINE FIND(JCSPUR,LISTER,LION,LOCAL,JCFINE,JCLOAD,JCDOFS,
     1 JCCLNO)
      COMMON/ALLOC/JCCORE(1)
      DIMENSION JCSPUR(1),LISTER(1),LION(1),LOCAL(1),JCFINE(1),JCLOAD(1)
     1,JCDUMY(15),JCLIMT(15)
C                   ENTER
      CALL SYSENT(1,2,1,2,JCCLNO)
C                   PREPARING FOR DATA TRANSFER
      JCSORF=1
      JCLOAD(14)=-1
      JCDUM2=0
      JCDUM7=LISTER(1)
      JCDUM8=LISTER(2)
      JCDUM9=-(-1)**IABS(JCDOFS)
C                   TRANSFERRING DATA
    1 CALL C4TRAN(JCSPUR,LISTER,JCDUMY,LOCAL,JCFINE,JCLOAD,JCSORF,1)
      IF(JCSORF.EQ.-1)GO TO 82
      IF(JCDUM2.GT.0)GO TO 6
      JCDUM3=LOCAL(1)
      JCDU10=JCLOAD(8)
      LISTER(1)=1
      LISTER(2)=JCDUM9
      IF(JCDUM2.LT.0)GO TO 6
      IF(JCDUM3.GT.0)GO TO 2
      IF(JCDUM3.LT.0)CALL SYSERR(1,JCLOAD(5))
      JCDUM3=1
      JCLIMT(1)=JCLOAD(10)
      GO TO 5
    2 DO 4 I=1,JCDUM3
      JCDUM4=3*I+1
      IF(LOCAL(JCDUM4).GT.0)GO TO 3
      JCDUM5=JCSPUR(5)+20
      JCDUM5=JCCORE(JCDUM5)+4*(LOCAL(JCDUM4-2)-1)+1
      JCLIMT(I)=JCCORE(JCDUM5+1)
      IF(JCLIMT(I).EQ.0)JCLIMT(1)=JCLOAD(10)
      JCLIMT(I)=JCLIMT(I)-JCCORE(JCDUM5)+1-LOCAL(JCDUM4-1)+1
      GO TO 4
    3 JCLIMT(I)=LOCAL(JCDUM4)-LOCAL(JCDUM4-1)+1
    4 CONTINUE
    5 JCDUM2=1
    6 JCDUM4=0
      JCDUM5=0
      JCDU11=JCDOFS
      DO 8 I=1,JCDUM3
      IF((JCDUM5*JCDU11).GT.0)GO TO 8
      JCDUM5=0
```

```
      JCDUM6=JCLIMT(I)
      DO 7 J=1,JCDUM6
      JCDUM4=JCDUM4+1
    7 IF(JCDUMY(JCDUM4).NE.LION(JCDUM4))JCDUM5=1
      IF((JCDOFS.GT.0).OR.(JCDUM5.EQ.1))GO TO 8
      JCDUM5=-1
      JCDOFS=I
    8 CONTINUE
      IF(JCDUM5.LE.0)GO TO 10
      JCDU11=JCLOAD(8)+JCDUM9
      IF((JCDU11.GT.0).AND.(JCDU11.LE.JCLOAD(9)))GO TO 1
   81 IF((IABS(JCDOFS).LT.3).OR.(JCDUM9.EQ.-1))GO TO 9
      JCDUM9=-1
      JCDUM2=-1
      LISTER(2)=JCDU10-JCLOAD(8)-1
      GO TO 1
   82 JCDU10=JCLOAD(9)+1
      IF(LISTER(1).EQ.1)GO TO 81
    9 JCDOFS=0
   10 LISTER(1)=JCDUM7
      LISTER(2)=JCDUM8
C                    EXIT
      CALL SYSEXT
      RETURN
      END
```

```
C
C*****************************************************************
C                              *********
C                              * STORE *
C                              *********
C
C       PURPOSE
C           THE PURPOSE OF STORE IS THE STORAGE OF DATA ITEMS IN A FILE
C           ENTRY.
C
C       USAGE
C           CALL STORE(JCSPUR,LISTER,LION,LOCAL,JCFINE,JCLOAD,JCCLNO)
C
C       DATA FORMAT
C           N/A
C
C       DESCRIPTION OF PARAMETERS
C           JCSPUR = A VECTOR DESCRIBED IN THE C4ADRS DOCUMENTATION
C           LISTER = A VECTOR DESCRIBED IN THE C4ADRS DOCUMENTATION
C           LION   = A VECTOR DESCRIBED IN THE C4TRAN DOCUMENTATION
C           LOCAL  = A VECTOR DESCRIBED IN THE C4ADRS DOCUMENTATION
C           JCFINE = A VECTOR DESCRIBED IN THE C4ADRS DOCUMENTATION
C           JCLOAD = A VECTOR DESCRIBED IN THE C4ADRS DOCUMENTATION
C           JCCLNO = A USER SPECIFIED CALL STATEMENT ID. NUMBER
C
C       REMARKS
C           THE NUMBER REPRESENTED BY JCTCIA MUST BE SUPPLIED IN THE
C           COMMON STATEMENT FOR SOME MACHINES.
C           TO FACILITATE THE PROPER INITIALIZATION OF THE JCLOAD
C           VECTOR, JCLOAD(15) MUST EQUAL -1 FOR THE FIRST CALL TO
C           SUPPLY.
C           JCLOAD(29) MUST NOT EQUAL -1 UNLESS VIOLATION OF FILE ENTRY
C           BOUNDARIES IS TO BE PERMITTED.
C           CARE MUST BE EXERCISED IN USING THE SECOND MODE OF
C           OPERATION (SEE METHOD) TO INSURE THAT THE JCLOAD VECTOR HAS
C           BEEN UNALTERED SINCE THE STORE OR FETCH CALL ADDRESSING THE
C           CURRENT FILE ENTRY.
C           THE LIST TYPE SPECIFIED BY JCSPUR(5) MUST BE FORWARD-
C           BACKWARD.
C           THIS PROGRAM, FETCH AND FIND ARE THE ONLY REMOTE ROUTINES
C           RECOMMENDED FOR DIRECT USER CALL.
C           JCLOAD(30) MUST CONTAIN THE ADDRESS OF THE POINTER TO THE
C           CATALOGUE.
C
C       SUBROUTINE AND FUNCTION SUBPROGRAMS REQUIRED
C           C4TRAN
C           SYSENT
C           SYSEXT
C
C       ERROR CODES FOR THIS PROGRAM
C           NONE
C
C       METHOD
C           LISTER MAY BE USED TO ADDRESS THE DESIRED FILE ENTRY VIA
C           CATALOGUE AND/OR DIRECTORY SELECTIONS, OR MAY SPECIFY THE
C           FILE ENTRY VIA LISTER(1)=1 AND LISTER(2)=THE NUMBER OF FILE
C           ENTRIES TO MOVE FROM THE CURRENT FILE ENTRY.  IF LOCAL(1)
C           IS -1, ONLY THE ADDRESSING OPERATION IS PERFORMED.  IF
C           LOCAL(1) IS 0 OR LARGER, STORE PERFORMS AS WOULD TOCELL.
C           CATALOGUE, REGISTER AND RECORD SEGMENTS ARE ACQUIRED FROM
C           ALLOCATABLE CORE AS REQUIRED TO PERFORM THE ADDRESS
C           COMPUTATION.  RECORD SEGMENTS ON OFF-LINE MEMORY WILL BE
```

```
C           RETRIEVED AS REQUIRED.
C
C                              **********
C
      SUBROUTINE STORE(JCSPUR,LISTER,LION,LOCAL,JCFINE,JCLOAD,JCCLNO)
      COMMON/ALLOC/JCCORE(1)
      DIMENSION JCSPUR(1),LISTER(1),LION(1),LOCAL(1),JCFINE(1),JCLOAD(1)
C                ENTER
      CALL SYSENT(1,2,1,3,JCCLNO)
C                PREPARING FOR DATA TRANSFER
      JCLOAD(14)=1
      M1=0
C                TRANSFERRING DATA
      CALL C4TRAN(JCSPUR,LISTER,LION,LOCAL,JCFINE,JCLOAD,M1,1)
C                EXIT
      CALL SYSEXT
      RETURN
      END
```

```
C
C*****************************************************************
C                              **********
C                              * SUPPLY *
C                              **********
C
C       PURPOSE
C           THE PURPOSE OF SUPPLY IS THE RETRIEVAL OF DATA
C           CORRESPONDING TO A RECORD SEGMENT.
C
C       USAGE
C           CALL SUPPLY(JCPKEY,JCFWOR,JCNWPR,JCCLNO)
C
C       DATA FORMAT
C           N/A
C
C       DESCRIPTION OF PARAMETERS
C           JCPKEY = A POSITIVE INTEGER KEY IDENTIFYING THE INFORMATION
C                    TO BE PLACED IN A RECORD SEGMENT
C           JCFWOR = THE ABSOLUTE ADDRESS OF THE FIRST WORD OF THE
C                    RECORD SEGMENT
C           JCNWPR = THE NUMBER OF WORDS PER RECORD SEGMENT
C           JCCLNO = A USER ASSIGNED CALL STATEMENT ID. PARAMETER
C
C       REMARKS
C           THE NUMBER REPRESENTED BY JCTCIA MUST BE SUPPLIED IN THE
C           COMMON STATEMENT FOR SOME MACHINES.
C           THE CURRENT VERSION OF SUPPLY DOES NOT RETRIEVE INFORMATION
C           FROM OFF-LINE STORAGE.  THIS VERSION ONLY PRODUCES AN ERROR
C           SUBROUTINE CALL WHICH IS NORMALLY FATAL.  IF A USER WISHES
C           TO EMPLOY OFF-LINE STORAGE, SUPPLY MUST BE MODIFIED TO
C           OPERATE AS DESCRIBED IN METHOD.
C
C       SUBROUTINE AND FUNCTION SUBPROGRAMS REQUIRED
C           SYSENT
C           SYSERR
C           SYSEXT
C
C       ERROR CODES FOR THIS PROGRAM
C           CODE FATAL  ERROR(DATA PROVIDED)
C           1    F      ATTEMPTED RETRIEVAL FROM MASS MEMORY (JCFWOR)
C
C       METHOD
C           SUPPLY RETRIEVES AN ENTIRE RECORD SEGMENT FROM OFF-LINE
C           STORAGE.  THE USER MAY UTILIZE ANY STORAGE DEVICE DESIRED.
C           THE INFORMATION IS TRANSFERRED TO ON-LINE MEMORY AS WHOLE
C           WORDS.  PARTIAL RETRIEVAL OF RECORD SEGMENTS IS NOT
C           PERMITTED.
C
C                              **********
C
      SUBROUTINE SUPPLY(JCPKEY,JCFWOR,JCNWPR,JCCLNO)
      COMMON/ALLOC/JCCORE(1)
C                 ENTER
      CALL SYSENT(1,2,1,4,JCCLNO)
      CALL SYSERR(-1,JCFWOR)
C                 EXIT
      CALL SYSEXT
      RETURN
      END
```

```
C
C*****************************************************************
C                          **************
C                          *    COAST     *
C                          *   GENERAL    *
C                          * INFORMATION  *
C                          **************
C
C       SPECIAL INSTURCTIONS/COIN
C           THE COAST SUBROUTINE COIN MUST BE CALLED PRIOR TO USING ANY
C           OTHER OF THE COAST SUBROUTINES.
C
C       SPECIAL INSTRUCTIONS/DEBUG
C           THE FOLLOWING INDICATES THE ORDER FOR OPERATIONAL CHECKS
C           OF THE COAST SYSTEM.  THE PROGRAMS FOR LEVEL TWO, FOR
C           INSTANCE, CALL ONLY SUBPROGRAMS FROM LEVELS ONE OR A.
C           SIMILARLY FOR ALL OTHER LEVELS.
C               SUBPROGRAM          LEVEL       PKG./PROG. ID. NUMBERS
C               LAND                A           -
C               LEOR                A           -
C               LOR                 A           -
C               SYSDMP              A           -
C               SYSENT              A           -
C               SYSERR              A           -
C               SYSEXT              A           -
C               J5INDX              1           3/16
C               J5IPOT              1           3/17
C               J5NUMB              1           3/20
C               J5SAVE              1           3/23
C               J5TST4              1           3/27
C               J5TST6              1           3/29
C               J5MUPD              2           3/18
C               J5NEXT              2           3/19
C               J5SAME              2           3/22
C               J5TST1              2           3/24
C               J5TST3              2           3/26
C               J5TST7              2           3/30
C               SETLIM              2           3/12
C               CORN                3           1/2
C               BSTLNK              3           3/1
C               FSTLNK              3           3/4
C               J5COMP              3           3/15
C               J5REDY              3           3/21
C               J5TST2              3           3/25
C               J5TST5              3           3/28
C               LNKBWD              3           3/5
C               LNKFWD              3           3/6
C               C8COMB              4           2/1
C               COPY                4           3/2
C               FRMCEL              4           3/3
C               LOCATE              4           3/7
C               RETURN              4           3/11
C               SETLNK              4           3/13
C               TOCELL              4           3/14
C               COIN                5           1/1
C               C8GIVE              5           2/2
C               POPUP               5           3/9
C               LIST                6           1/3
C               NEWCEL              7           3/8
C               PUSH                8           3/10
C
C       SPECIAL INSTRUCTIONS/ERROR CODES
```

```
C
C          AN ERROR MESSAGE IS OF THE FOLLOWING FORM.
C              ERROR N ( , , , , ) E  D
C          WHERE EACH   INDICATES IN ORDER OF OCCURRENCE
C          1  THE NUMBER OF SUBROUTINES CURRENTLY ENTERED
C          2  THE SYSTEM ID. NUMBER
C          3  THE LEVEL ID. NUMBER
C          4  THE PACKAGE ID. NUMBER
C          5  THE PROGRAM ID. NUMBER
C          6  THE CALL NUMBER
C          7  THE ERROR CODE
C          8  THE DATA PROVIDED TO AID IN DETERMINING THE NATURE OR
C             CAUSE OF THE ERROR
C          FIELDS 2 THROUGH 6 UNIQUELY IDENTIFY A PROGRAM CALLING THE
C          SYSERR ROUTINE.  FIELDS 7 AND 8 ARE PASSED TO SYSERR.  A
C          NEGATIVE ERROR CODE INDICATES A FATAL ERROR, TERMINATING
C          EXECUTION.  EXECUTION CONTINUES FOR A POSITIVE ERROR CODE.
C          JCCORE(14) PROVIDES THE MEANS FOR OVER-RIDING THE SIGN OF
C          AN ERROR CODE.  WHEN SET AT ZERO, JCCORE(14) HAS NO EFFECT.
C          IF -1 IS THE VALUE OF JCCORE(14), THEN ANY ERROR COMMITTED
C          IS FATAL.  IF +1 IS THE VALUE, ANY ERROR WILL BE NON-FATAL.
C
C        SPECIAL INSTRUCTIONS/DUMP
C            IF A DUMP OF JCCORE IS DESIRED WHEN AN ERROR CONDITION IS
C            ENCOUNTERED, JCCORE(19) MUST BE SET EQUAL TO A NON-ZERO
C            INTEGER.  IF POSITIVE, ONLY PRINTED OUTPUT WILL BE
C            PROVIDED.  FOR BOTH PRINTED AND PUNCHED OUTPUT, JCCORE(19)
C            MUST BE SET TO -N, WHERE N IS THE DATA SET REFERENCE NUMBER
C            FOR THE CARD PUNCH.  JCCORE(19)=0 SUPPRESSES THE DUMP.
C
C        SPECIAL INSTRUCTIONS/LAND, LEOR, LOR
C            LAND, LEOR, AND LOR ARE RESPECTIVELY THE FUNCTION NAMES FOR
C            THE LOGICAL OPERATIONS 'AND', 'EXCLUSIVE OR', AND 'OR'.
C            THESE FUNCTIONS MUST BE PROVIDED IN OBJECT DECK FORM.  THE
C            CURRENTLY INSTALLED OBJECT DECK IS COMPATABLE WITH THE
C            FOLLOWING MACHINES.
C                IBM 360 H LEVEL
C                IBM 370
C            THE TWO ARGUMENTS REQUIRED BY EACH FUNCTION ARE PROVIDED AS
C            PASSED PARAMETERS.
C
C        SPECIAL INSTRUCTIONS/TRACE
C            THE CODING REQUIRED FOR TRACING THROUGH EACH CALLED
C            SUBPROGRAM IS PROVIDED.  THIS OPTION MAY BE INSTALLED OR
C            REMOVED BY REMOVING OR INSTALLING THE COMMENT C ON EACH OF
C            THE AFFECTED CARDS.  WHEN INSTALLED, THE TRACE IS
C            CONTROLLED BY JCCORE(20) AS FOLLOW.
C                JCCORE(20) = 0  NO TRACE
C                             N  TRACE MESSAGE FOR ALL SUBPROGRAMS
C                                ENTERED UP TO THE POINT AT WHICH N
C                                SUBPROGRAMS ARE CURRENTLY ENTERED.  THE
C                                TRACE RESUMES UPON EXITING A SUFFICIENT
C                                NUMBER OF SUBPROGRAMS SO THAT THE
C                                NUMBER OF CURRENTLY ENTERED IS N OR
C                                LESS.
C            WHEN TURNING A TRACE OFF (BY SETTING JCCORE(20) TO 0), SET
C            JCCORE(13) TO 6.
C
C        CURRENT STATUS
C            OPERATIONAL
C
C*************************************************************************
C
```

```
C
C*****************************************************************
C                              ********
C                              * COIN *
C                              ********
C
C
C       PURPOSE
C           THIS SUBPROGRAM IS THE COAST SYSTEM INITIALIZATION ROUTINE.
C
C       USAGE
C           CALL COIN(JCINPT,JCOTPT)
C
C       DATA FORMAT
C           JCNBPW,JCTCIA,JCCSSZ,JCMAXB     CARD TYPE 1     4I20
C           JCNOLT                                  2       4I20
C           JCNODI,JCFOFB                           3       4I20
C           JCDFWI,JCDLWI,JCSHFT,JCMASK             4       4I20
C
C       DESCRIPTION OF PARAMETERS
C           JCINPT = INPUT DATA SET REFERENCE NUMBER (SEE REMARKS)
C           JNOTPT = OUTPUT DATA SET REFERENCE NUMBER (SEE REMARKS)
C           JCNBPW = NUMBER OF BITS PER WORD
C           JCTCIA = TOTAL WORDS OF CORE INITIALLY AVAILABLE, SET ASIDE
C                    FOR ALLOCATION BY A COMMON STATEMENT.
C           JCCSSZ = CORE SEGMENT SIZE IN WORDS
C           JCMAXB = MAXIMUM NUMBER OF SEGMENTS FOR EACH BUDDY.
C           JCNOLT = NUMBER OF LIST TYPES
C           JCNODI = NUMBER OF DATA TYPES FOR A LIST TYPE
C           JCFOFB = PARAMETER EQUAL TO 1 OR 2 ACCORDING TO A LIST TYPE
C                    BEING FORWARD ONLY OR FORWARD-BACKWARD THREADED
C                    RESPECTIVELY
C           JCDFWI = THE INDEX FOR THE FIRST WORD IN WHICH A DATA TYPE
C                    OCCURS
C           JCDLWI = THE INDEX FOR THE LAST WORD IN WHICH A DATA TYPE
C                    OCCURS WHERE 0 INDICATES EACH WORD FROM JCDFWI TO
C                    THE LAST WORD OF A LIST ENTRY
C           JCSHFT = THE SHIFT CORRESPONDING TO A DATA TYPE
C           JCMASK = THE MASK CORRESPONDING TO A DATA TYPE
C
C       REMARKS
C           THE NUMBER REPRESENTED BY JCTCIA MUST BE SUPPLIED IN THE
C           COMMON STATEMENT.
C           SYNONYMOUS TERMS FOR  DATA SET REFERENCE NUMBER  ARE
C            SYMBOLIC UNIT REFERENCE NUMBER,   SYMBOLIC UNIT NUMBER,
C            LOGICAL UNIT,  AND  LOGICAL FILE.
C           THERE WILL BE ONE DATA CARD FOR EACH DATA TYPE FOR EACH
C           LIST TYPE.  THAT IS, THE CARD SUPPLYING JCNODI MUST BE
C           FOLLOWED IMMEDIATELY BY JCNODI CARDS DESCRIBING THE DATA
C           TYPES.  ALSO THERE MUST BE JCNOLT SETS OF CARDS DESCRIBING
C           THE LIST TYPES.  THE FIRST AND SECOND DATA TYPES OF EACH
C           LIST TYPE MUST BE RESPECTIVELY THE FORWARD LINK AND IF
C           APPLICABLE THE BACKWARD LINK.
C
C       SUBROUTINE AND FUNCTION SUBPROGRAMS REQUIRED
C           C8COMB
C           J5IPOT
C           J5NUMB
C
C       ERROR CODES FOR THIS SUBPROGRAM
C           NONE
C           SEE SUBPROGRAM ERROR FOR A COMPLETE SYSTEM ERROR LISTING.
C
```

```
C
C       METHOD
C           SELF EXPLANATORY
C
C                                   ********
C
      SUBROUTINE COIN(JCINPT,JCOTPT)
      COMMON/ALLOC/JCCORE(1)
      JCCORE(10)=JCINPT
      JCCORE(11)=JCOTPT
      DO 3 I=12,20
    3 JCCORE(I)=0
      JCCORE(13)=6
    1 FORMAT(4I20)
C                     INPUT OF PARAMETERS
      READ(JCINPT,1)(JCCORE(I),I=1,4)
      READ(JCINPT,1)JCNOLT
      JCDUM1=21+JCNOLT
C                     INPUT OF DATA LOCATION ARRAY
      DO 2 I=1,JCNOLT
      READ(JCINPT,1)JCNODI,JCCORE(JCDUM1)
      JCCORE(20+I)=JCDUM1
      JCDUM2=JCDUM1+1
      JCDUM1=JCDUM1+4*JCNODI
      READ(JCINPT,1)(JCCORE(J),J=JCDUM2,JCDUM1)
    2 JCDUM1=JCDUM1+1
C                     CALCULATION OF LOCATIONS OF AVAILABLE SPACE LIST AND
C                     RETURN MASK
      JCCORE(5)=JCDUM1
      JCCORE(6)=J5IPOT(JCCORE(4),1,0)
      JCCORE(4)=2**JCCORE(6)
      JCCORE(7)=JCCORE(5)+JCCORE(6)+1
      JCDUM1=JCCORE(2)/JCCORE(3)
      JCDUM2=JCCORE(1)-1
      JCCORE(8)=J5NUMB(JCDUM1,JCDUM2,0)
      JCCORE(9)=JCCORE(7)+JCCORE(8)
C                     PREPARATION OF AVAILABLE SPACE LIST
      JCDUM1=JCCORE(7)
      JCCORE(JCDUM1)=-1
      JCDUM1=JCCORE(9)
      JCDUM2=JCCORE(2)
      DO 4 I=JCDUM1,JCDUM2
    4 JCCORE(I)=-I
      CALL C8COMB(0)
      RETURN
      END
```

```
C
C*****************************************************************
C                                 ********
C                                 * CORN *
C                                 ********
C
C       PURPOSE
C           THE PURPOSE OF THIS SUBPROGRAM IS TO RETURN TO THE SYSTEM
C           CORE SPECIFIED BY A USER SUBPROGRAM.
C
C       USAGE
C           CALL CORN(IAUASL,JCCLNO)
C
C       DATA FORMAT
C           N/A
C
C       DESCRIPTION OF PARAMETERS
C           SEE THE DOCUMENTATION FOR LIST
C
C       REMARKS
C           THE NUMBER REPRESENTED BY JCTCIA MUST BE SUPPLIED IN THE
C           COMMON STATEMENT.
C           AN IAUASL VECTOR MAY BE CREATED FOR THE PURPOSE OF PARTIAL
C           CORE RETURN.  THE ARTIFICIAL USER CORE ASSIGNMENT MASK
C           WOULD INDICATE THE CORE SEGMENTS TO BE RETURNED.
C
C       SUBROUTINE AND FUNCTION SUBPROGRAMS REQUIRED
C           J5TST1
C           LOR
C           SYSENT
C           SYSEXT
C
C       ERROR CODES FOR THIS SUBPROGRAM
C           NONE
C
C       METHOD
C           ALL CORE RETURNED TO CORN IS INDICATED BY A RETURN MASK.
C           THIS CORE REMAINS UNUSED UNTIL ALL OF THE AVAILABLE CORE
C           HAS BEEN EXHAUSTED.  THE BUDDY SYSTEM SUBPROGRAM C8COMB
C           WOULD THEN BE AUTOMATICALLY CALLED TO MAKE THE CORE
C           INDICATED BY THE RETURN MASK AVAILABLE FOR ALLOCATION.
C
C                                 ********
C
      SUBROUTINE CORN(IAUASL,JCCLNO)
      COMMON/ALLOC/JCCORE(1)
      DIMENSION IAUASL(5)
      CALL SYSENT(1,1,1,2,JCCLNO)
      JCDUM1=JCCORE(8)
      JCDUM2=IAUASL(1)
      IF(JCDUM2.EQ.0)GO TO 2
C                 IS MASK LEGAL CORE ASSIGNMENT
      CALL J5TST1(JCDUM2,1)
      JCDUM3=JCCORE(7)
C                 UPDATING RETURN MASK AND CLEARING USER MASK
      DO 1 I=1,JCDUM1
      JCCORE(JCDUM3)=LOR(JCCORE(JCDUM3),JCCORE(JCDUM2))
      JCCORE(JCDUM2)=0
      JCDUM2=JCDUM2+1
    1 JCDUM3=JCDUM3+1
C                 SETTING MASK AND HEADCELL POINTERS TO ZERO
      IAUASL(1)=0
```

```
2 IAUASL(2)=0
  CALL SYSEXT
  RETURN
  END
```

```
C
C*****************************************************************
C                              ********
C                              * LIST *
C                              ********
C
C       PURPOSE
C           THE PURPOSE OF THIS SUBPROGRAM IS TO FURNISH THE CALLING
C           PROGRAM WITH THE REQUESTED CORE.
C
C       USAGE
C           CALL LIST(IAUASL,JCCLNO)
C
C       DATA FORMAT
C           N/A
C
C       DESCRIPTION OF PARAMETERS
C           IAUASL = A VECTOR CONTAINING THE POINTER TO THE USER'S CORE
C                    ASSIGNMENT MASK, THE POINTER TO THE USER'S
C                    AVAILABLE SPACE LIST HEADCELL, THE NUMBER OF LIST
C                    ENTRIES REQUESTED, THE NUMBER OF WORDS PER LIST
C                    ENTRY, AND LASTLY THE INDEX SPECIFYING THE LIST
C                    TYPE.  THE IAUASL ARRAY MUST CONTAIN THESE FIVE
C                    ENTRIES IN THE INDICATED ORDER. IAUASL MAY BE A
C                    COLUMN FROM A MATRIX STORING THE ABOVE INFORMATION
C                    FOR EACH USER MAINTAINED AVAILABLE SPACE LIST.  IN
C                    THIS CASE THE PASSED PARAMETER WOULD BE THE MATRIX
C                    ELEMENT AT THE TOP OF THE COLUMN DESCRIBING THE
C                    PARAMETERS PERTAINING TO THE SELECTED USER ASL.
C           JCCLNO = A USER SPECIFIED CALL STATEMENT ID. NUMBER
C
C       REMARKS
C           THE NUMBER REPRESENTED BY JCTCIA MUST BE SUPPLIED IN THE
C           COMMON STATEMENT.
C
C       SUBROUTINE AND FUNCTION SUBPROGRAMS REQUIRED
C           C8GIVE
C           J5MUPD
C           J5NUM8
C           J5REDY
C           SYSENT
C           SYSERR
C           SYSEXT
C
C       ERROR CODES FOR THIS SUBPROGRAM
C           CODE FATAL  ERROR(DATA PROVIDED)
C           1    F      BUDDY TOO SMALL TO USE AS A MASK (ADDRESS OF
C                       BUDDY)
C           2    F      BUDDY TOO SMALL FOR USER (ADDRESS OF BUDDY)
C           3    F      NEGATIVE MASK WORD INDEX (ADDRESS OF MASK)
C           4    F      ILLEGAL FIRST WORD OF BUDDY (ADDRESS OF BUDDY)
C
C       METHOD
C           THE BUDDY SYSTEM IS UTILIZED TO SECURE CORE FOR THE
C           SATISFACTION OF REQUESTS BY CALLING PROGRAMS.  THE CORE
C           ACQUIRED BY LIST IS THREADED USING THE FORWARD LINK OF THE
C           SPECIFIED LIST TYPE.  THE CREATION AND UPDATING OF USER
C           CORE ASSIGNMENT MASKS ARE AUTOMATIC FUNCTIONS PERFORMED BY
C           LIST.
C
C                              ********
C
```

```
      SUBROUTINE LIST(IAUASL,JCCLNO)
      COMMON/ALLOC/JCCORE(1)
      DIMENSION IAUASL(5)
      CALL SYSENT(1,1,1,3,JCCLNO)
C                   IS THIS FIRST CALL OF LIST
      IF(IAUASL(1).NE.0)GO TO 1
C                   PREPARATION OF REQUEST FOR CORE ASSIGNMENT MASK
      JCDUM1=1
      JCDUM2=J5NUMB(JCCORE(8),JCCORE(3),1)
      JCDUM3=JCDUM2
      JCERRC=-01
      GO TO 3
C                   PREPARATION OF REQUEST FOR CORE
    1 JCNOCT=0
      JCERRC=-02
      IAUASL(2)=0
      JCDUM1=2
      JCDUM2=J5NUMB(IAUASL(4),JCCORE(3),2)
    2 JCDUM4=(IAUASL(3)-JCNOCT)*IAUASL(4)
C                   HAS USER REQUEST FOR CORE BEEN SATISFIED
      IF(JCDUM4.LE.0)GO TO 104
      JCDUM3=J5NUMB(JCDUM4,JCCORE(3),3)
C                   REQUEST FOR CORE
    3 CALL C8GIVE(JCHDCL,JCNSIB,JCDUM3,JCDUM2,1)
      IF(JCNSIB.LT.JCDUM2)CALL SYSERR(JCERRC,JCHDCL)
      GO TO (4,6),JCDUM1
C                   ESTABLISHMENT OF USER CORE ASSIGNMENT MASK
    4 IAUASL(1)=JCHDCL
      JCDUM5=JCCORE(8)
      DO 5 I=1,JCDUM5
      JCDUM6=JCHDCL+I-1
    5 JCCORE(JCDUM6)=0
C                   UPDATING OF THE USER CORE ASSIGNMENT MASK
    6 JCDUM3=(JCHDCL-JCCORE(9))/JCCORE(3)
      IF(JCDUM3.LT.0)CALL SYSERR(-3,JCHDCL)
      JCDUM4=JCDUM3*JCCORE(3)+JCCORE(9)
      IF(JCDUM4.NE.JCHDCL)CALL SYSERR(-4,JCHDCL)
      JCDUM3=JCDUM3+1
      CALL J5MUPD(JCDUM3,JCNSIB,IAUASL(1),1)
      IF(JCDUM1.EQ.1)GO TO 1
C                   THREADING OF THE CORE PER USER REQUEST
      CALL J5REDY(JCNOCT,JCNSIB,IAUASL(4),IAUASL(5),JCHDCL,IAUASL(2),1)
      GO TO 2
  104 CALL SYSEXT
      RETURN
      END
```

```
C
C*****************************************************************
C                              **********
C                              * C8COMB *
C                              **********
C
C        PURPOSE
C            THE PURPOSE OF C8COMB IS TO ESTABLISH AN AVAILABLE SPACE
C            LIST STRUCTURE WHEN CALLED EITHER BY COIN OR BY C8GIVE.
C
C        USAGE
C            CALL C8COMB(JCCLNO)
C
C        DATA FORMAT
C            N/A
C
C        DESCRIPTION OF PARAMETERS
C            JCCLNO = A USER SPECIFIED CALL STATEMENT ID. NUMBER
C
C        REMARKS
C            THE NUMBER REPRESENTED BY JCTCIA MUST BE SUPPLIED IN THE
C            COMMON STATEMENT.
C            THERE ARE NO PASSED PARAMETERS AS ALL INFORMATION REQUIRED
C            BY C8COMB IS AVAILABLE FROM THE COMMON CORE.
C
C        SUBROUTINE AND FUNCTION SUBPROGRAMS REQUIRED
C            J5IPOT
C            J5MUPD
C            J5NEXT
C            J5REDY
C            J5SAME
C            J5SAVE
C            J5TST7
C            SYSENT
C            SYSEXT
C
C        ERROR CODES FOR THIS SUBPROGRAM
C            NONE
C
C        METHOD
C            WHEN CALLED BY COIN, C8COMB PREPARES THE CONTINUOUS CORE
C            FROM THE FIRST ALLOCATABLE WORD TO THE LAST FOR ALLOCATION.
C            AN AVAILABLE SPACE LIST IS PREPARED FOR EACH BUDDY SIZE UP
C            TO THE USER SPECIFIED MAXIMUM.  A BUDDY WILL ALWAYS BE
C            COMPOSED OF AN INTEGER POWER OF TWO NUMBER OF SEGMENTS.
C            WHEN CALLED BY C8GIVE, THE AVAILABLE SPACE LISTS ARE
C            PREPARED FROM THE RETURN MASK.
C
C                              **********
C
      SUBROUTINE C8COMB(JCCLNO)
      COMMON/ALLOC/JCCORE(1)
      CALL SYSENT(1,1,2,1,JCCLNO)
      JCDUM1=(JCCORE(2)-JCCORE(9)+1)/JCCORE(3)
      JCDUM2=JCCORE(7)
C                 IS CALL BY COIN OR C8GIVE
      IF(JCCORE(JCDUM2).GE.0)GO TO 3
C                 CALL BY COIN
      JCDUM3=JCCORE(6)+1
      JCDUM4=JCCORE(5)-1
C                 ZEROING POINTERS TO ASL HEADCELLS
      DO 1 I=1,JCDUM3
```

```
      JCDUM4=JCDUM4+1
    1 JCCORE(JCDUM4)=0
      JCDUM3=JCCORE(9)
      JCDUM5=JCCORE(4)
      JCDUM6=JCCORE(4)*JCCORE(3)
C                     PREPARATION OF ASL FOR SIZE JCDUM6 BUDDIES
    2 JCNOCT=0
      CALL J5READY(JCNOCT,JCDUM1,JCDUM6,1,JCDUM3,JCCORE(JCDUM4),1)
      JCDUM1=JCDUM1-JCNOCT*JCDUM5
C                     ARE SMALLER BUDDIES STILL UNLISTED ON ASL
      IF(JCDUM1.LE.0)GO TO 7
C                     PREPARATION FOR SMALLER BUDDY ASL CREATION
      JCDUM3=JCDUM3+JCNOCT*JCDUM6
      JCDUM4=J5IPDT(JCDUM1,1,1)
      JCDUM5=2**JCDUM4
      JCDUM6=JCCORE(3)*JCDUM5
      JCDUM4=JCDUM4+JCCORE(5)
      GO TO 2
C                     CALL BY C8GIVE
    3 JCDUM3=1
      JCDUM4=JCCORE(5)
      JCDUM5=JCDUM2-1
      JCDUM7=JCCORE(21)+1
      DO 9 I=JCDUM4,JCDUM5
      JCDUM6=JCCORE(I)
   10 IF(JCDUM6.EQ.0)GO TO 11
      JCDUM8=(JCDUM6-JCCORE(9))/JCCORE(3)+1
      CALL J5MUPD(JCDUM8,JCDUM3,JCDUM2,1)
      JCDUM6=JCDUM6+JCCORE(JCDUM7)-1
      JCDUM6=J5SAVE(JCDUM7,JCDUM6,1)
      GO TO 10
   11 JCCORE(I)=0
    9 JCDUM3=JCDUM3*2
      JCDUM3=1
C                     LOCATION OF NEXT SEGMENT ON RETURN MASK
    4 JCDUM3=J5NEXT(JCDUM3,JCDUM2,1)
      IF(JCDUM3.EQ.0)GO TO 7
      JCDUM4=1
      JCDUM5=0
C                     A BUDDY CAN BE NO LARGER THAN THE USER SPECIFIED MAX.
    5 IF(JCDUM4.EQ.JCCORE(4))GO TO 6
C                     DETERMINATION OF FEASIBILITY OF NEXT LARGER BUDDY
      JCDUM6=JCDUM1-JCDUM3-JCDUM4
      IF(JCDUM6.LT.0)GO TO 6
      JCDUM6=(JCDUM3-1)/JCDUM4
      JCDUM6=(JCDUM6/2)*2-JCDUM6
      IF(JCDUM6.LT.0)GO TO 6
C                     TEST FOR PRESENCE ON RETURN MASK OF ALL SEGMENTS IN
C                     UPPER HALF OF NEXT LARGER BUDDY
      JCDUM6=JCDUM3+JCDUM4
      JCDUM6=J5TST7(JCDUM6,JCDUM4,JCDUM2,1)
      IF(JCDUM6.EQ.2)GO TO 6
      JCDUM4=2*JCDUM4
      JCDUM5=JCDUM5+1
      GO TO 5
C                     ADD THE LAST FEASIBLE AND COMPLETE BUDDY TO THE
C                     APPROPRIATE AVAILABLE SPACE LIST
    6 JCDUM6=(JCDUM3-1)*JCCORE(3)+JCCORE(9)
      JCDUM5=JCDUM5+JCCORE(5)
      CALL J5SAME(JCDUM7,JCCORE(JCDUM5),JCDUM6,1)
      JCCORE(JCDUM5)=JCDUM6
      JCDUM3=JCDUM3+JCDUM4
```

```
      GO TO 4
C                    ZEROING ALL WORDS OF THE RETURN MASK
    7 JCDUM1=JCCORE(8)
      DO 8 I=1,JCDUM1
      JCCORE(JCDUM2)=0
    8 JCDUM2=JCDUM2+1
      CALL SYSEXT
      RETURN
      END
```

```
C
C*****************************************************************
C                              **********
C                              * C8GIVE *
C                              **********
C
C       PURPOSE
C           C8GIVE IS RESPONSIBLE FOR SUPPLYING THE SUBPROGRAM LIST
C           WITH THE NECESSARY BUDDIES OF AVAILABLE CORE TO SATISFY
C           USER DEMANDS FOR CORE.  C8GIVE ALSO WILL AUTOMATICALLY CALL
C           C8COMB IN THE EVENT A CORE REQUEST FROM LIST CANNOT BE
C           HONORED.
C
C       USAGE
C           CALL C8GIVE(JCHDCL,JCNSIB,JCNOSD,JCMNOS,JCCLNO)
C
C       DATA FORMAT
C           N/A
C
C       DESCRIPTION OF PARAMETERS
C           JCHDCL = THE POINTER TO THE FIRST WORD OF A BUDDY
C           JCNSIB = THE NUMBER OF SEGMENTS IN THE BUDDY
C           JCNOSD = THE NUMBER OF SEGMENTS DESIRED BY LIST
C           JCMNOS = THE MINIMUM NUMBER OF SEGMENTS WHICH A BUDDY MUST
C                    CONTAIN TO BE ACCEPTABLE.
C           JCCLNO = A USER SPECIFIED CALL STATEMENT ID. NUMBER
C
C       REMARKS
C           THE NUMBER REPRESENTED BY JCTCIA MUST BE SUPPLIED IN THE
C           COMMON STATEMENT.
C           THIS SUBPROGRAM IS NOT DESIGNED FOR DIRECT CALL BY USER
C           SUBPROGRAMS.  ALL REQUESTS FOR CORE SHOULD BE MADE VIA THE
C           LIST SUBPROGRAM.
C
C       SUBROUTINE AND FUNCTION SUBPROGRAMS REQUIRED
C           C8COMB
C           J5IPOT
C           J5SAME
C           J5SAVE
C           SYSENT
C           SYSERR
C           SYSEXT
C
C       ERROR CODES FOR THIS SUBPROGRAM
C           CODE FATAL  ERROR(DATA PROVIDED)
C           1    F      NO AVAILABLE CORE REMAINING (JCMNOS)
C
C       METHOD
C           IF THE SMALLEST BUDDY SATISFYING THE REQUEST FROM LIST IS
C           EITHER AVAILABLE OR CAN BE MANUFACTURED FROM A LARGER
C           BUDDY, THEN THIS SMALLEST BUDDY WILL BE ISSUED TO LIST.
C           IF ONLY BUDDIES SMALLER THAN THE REQUESTED SIZE ARE
C           AVAILABLE, THE LARGEST OF THESE WILL BE PASSED TO LIST
C           PROVIDING IT CONTAINS AT LEAST JCMNOS SEGMENTS.  IF NO
C           ACCEPTABLE BUDDY IS AVAILABLE, C8GIVE CALLS C8COMB AND THE
C           SEARCH FOR AN ACCEPTIBLE BUDDY IS REPEATED.  A SECOND
C           FAILURE TO HONOR THE LIST REQUEST FOR CORE TERMINATES
C           PROGRAM EXECUTION.
C
C                              **********
C
      SUBROUTINE C8GIVE(JCHDCL,JCNSIB,JCNOSD,JCMNOS,JCCLNO)
```

```
      COMMON/ALLOC/JCCORE(1)
      CALL SYSENT(1,1,2,2,JCCLNO)
      JCDUM1=0
      JCDUM2=JCCORE(21)+1
C                  CALCULATION OF SIZE OF BUDDY JUST SATISFYING REQUEST
      JCDUM3=J5IPOT(JCNOSO,2,1)
      IF(JCDUM3.GT.JCCORE(6))JCDUM3=JCCORE(6)
      JCDUM4=2**JCDUM3
    1 JCDUM5=2
      JCDUM6=JCDUM3
      JCNSIB=JCDUM4
      JCDUM7=0
    2 JCDUM8=JCCORE(5)+JCDUM6
C                  CHECKING ASL FOR AVAILABILITY OF BUDDY
      JCHDCL=JCCORE(JCDUM8)
      IF(JCHDCL.NE.0)GO TO 7
      GO TO (6,3,4),JCDUM5
    3 JCDUM5=3
C                  SEARCH OF ASL FOR LARGER BUDDY
    4 IF(JCDUM6.GE.JCCORE(6))GO TO 5
      JCDUM6=JCDUM6+1
      JCNSIB=2*JCNSIB
      JCDUM7=JCDUM7+1
      GO TO 2
    5 JCDUM5=1
      JCDUM6=JCDUM3
      JCNSIB=JCDUM4
C                  SEARCH OF ASL FOR SMALLER BUDDY
    6 JCDUM6=JCDUM6-1
      JCNSIB=JCNSIB/2
C                  SMALLER BUDDY MUST CONTAIN JCMNOS SEGMENTS
      IF(JCNSIB.GE.JCMNOS)GO TO 2
C                  CALL FOR C8COMB TO ADD TO THE ASL THE CORE INDICATED
C                  BY THE RETURN MASK
      IF(JCDUM1.EQ.1)CALL SYSERR(-1,JCMNOS)
      JCDUM1=1
      CALL C8COMB(1)
      GO TO 1
C                  UPDATING AVAILABLE SPACE LIST
    7 JCDUM3=JCHDCL+JCCORE(JCDUM2)-1
      JCCORE(JCDUM8)=J5SAVE(JCDUM2,JCDUM3,1)
      IF(JCDUM5.LT.3)GO TO 104
      DO 8 I=1,JCDUM7
      JCNSIB=JCNSIB/2
      JCDUM8=JCDUM8-1
      JCDUM3=JCHDCL+JCNSIB*JCCORE(3)
      CALL J5SAME(JCDUM2,JCCORE(JCDUM8),JCDUM3,1)
    8 JCCORE(JCDUM8)=JCDUM3
  104 CALL SYSEXT
      RETURN
      END
```

```
C
C*****************************************************************
C                              **********
C                              * BSTLNK *
C                              **********
C
C       PURPOSE
C           THIS SUBPROGRAM PERMITS THE SETTING OF THE BACKWARD LINK
C           OF A GIVEN LIST ENTRY
C
C       USAGE
C           CALL BSTLNK(IAUASL,JCPOFP,JCBLNK,JCCLNO)
C
C       DATA FORMAT
C           N/A
C
C       DESCRIPTION OF PARAMETERS
C           IAUASL = A VECTOR CONTAINING THE POINTER TO THE USER'S CORE
C                    ASSIGNMENT MASK, THE POINTER TO THE USER'S
C                    AVAILABLE SPACE LIST HEADCELL, THE NUMBER OF LIST
C                    ENTRIES REQUESTED, THE NUMBER OF WORDS PER LIST
C                    ENTRY, AND LASTLY THE INDEX SPECIFYING THE LIST
C                    TYPE.  THE IAUASL ARRAY MUST CONTAIN THESE FIVE
C                    ENTRIES IN THE INDICATED ORDER. IAUASL MAY BE A
C                    COLUMN FROM A MATRIX STORING THE ABOVE INFORMATION
C                    FOR EACH USER MAINTAINED AVAILABLE SPACE LIST.  IN
C                    THIS CASE THE PASSED PARAMETER WOULD BE THE MATRIX
C                    ELEMENT AT THE TOP OF THE COLUMN DESCRIBING THE
C                    PARAMETERS PERTAINING TO THE SELECTED USER ASL.
C           JCPOFP = THE POINTER TO THE FIRST WORD OF THE LIST ENTRY
C           JCBLNK = THE VALUE AT WHICH THE BACKWARD LINK IS TO BE SET
C           JCCLNO = A USER SPECIFIED CALL STATEMENT ID. NUMBER
C
C       REMARKS
C           THE NUMBER REPRESENTED BY JCTCIA MUST BE SUPPLIED IN THE
C           COMMON STATEMENT.
C           SET JCCORE(14) AT -1 IF ALL ERRORS COMMITTED WITHIN BSTLNK
C           ARE TO BE NONFATAL.  THESE ERRORS ARE FATAL OTHERWISE.
C
C       SUBROUTINE AND FUNCTION SUBPROGRAMS REQUIRED
C           J5SAME
C           SYSENT
C           SYSEXT
C
C       ERROR CODES FOR THIS SUBPROGRAM
C           NONE
C
C       METHOD
C           THE LIST MUST BE FORWARD-BACKWARD TO USE THIS SUBPROGRAM.
C
C                              **********
C
      SUBROUTINE BSTLNK(IAUASL,JCPOFP,JCBLNK,JCCLNO)
      COMMON/ALLOC/JCCORE(1)
      DIMENSION IAUASL(5)
      CALL SYSENT(1,1,3,I,JCCLNO)
      JCTOLO = IAUASL(5)
C                TEST FOR LEGAL JCPOFP
      CALL J5TST1(JCPOFP,1)
C                TEST FOR FORWARD-BACKWARD LIST STRUCTURE
      CALL J5TST3(JCTOLO,JCPOFP,1)
      JCDUM1=JCCORE(JCTOLO+20)+5
```

```
C                    CALL J5SAME TO SET THE LINK
      CALL J5SAME(JCOUM1,JCBLNK,JCPOFP,1)
      CALL SYSEXT
      RETURN
      END
```

```
C
C*****************************************************************
C                              ********
C                              * COPY *
C                              ********
C
C       PURPOSE
C           THE PURPOSE OF COPY IS TO TRANSFER ALL OR ONLY A PART OF
C           THE INFORMATION STORED IN ONE LIST ENTRY TO ANOTHER LIST
C           ENTRY.
C
C       USAGE
C           CALL COPY(IAUASL,JCPOFP,JCPOSP,LOCAL,JCCLNO)
C
C       DATA FORMAT
C           N/A
C
C       DESCRIPTION OF PARAMETERS
C           IAUASL = A VECTOR DESCRIBED IN THE BSTLNK DOCUMENTATION
C           JCPOFP = THE POINTER TO THE FIRST WORD OF THE LIST ENTRY
C                    CONTAINING THE INFORMATION TO BE TRANSFERRED
C           JCPOSP = THE POINTER TO THE FIRST WORD OF THE LIST ENTRY
C                    TO RECEIVE THE INFORMATION
C           LOCAL  = A USER SUPPLIED VECTOR INDICATING THE DATA TYPES
C                    FOR TRANSFER.  THE FIRST WORD INDICATES THE NUMBER
C                    OF DIFFERENT DATA TYPES, AND IS FOLLOWED BY THAT
C                    NUMBER OF SETS OF THREE WORDS EACH.  THE FIRST OF
C                    A THREE WORD SET IS THE DATA TYPE NUMBER FOR THE
C                    LIST TYPE, WHILE THE REMAINING TWO WORDS ARE WORD
C                    INDICES STATING THE FIRST AND LAST DATA ITEM OF
C                    THE CURRENT TYPE FOR TRANSFER.  IF THE THIRD WORD
C                    OF THE SET IS ZERO, THE LAST ITEM TRANSFERRED WILL
C                    BE THE LAST ITEM INDICATED IN THE DATA LOCATION
C                    ARRAY FOR THE CURRENT DATA TYPE.  IF LOCAL(1) IS
C                    -1 NO INFORMATION WILL BE COPIED, AND IF LOCAL(1)
C                    IS 0 ALL OF THE DATA IN JCPOFP WILL BE REPRODUCED
C                    IN JCPOSP.
C           JCCLNO = A USER SPECIFIED CALL STATEMENT ID. NUMBER
C
C       REMARKS
C           THE NUMBER REPRESENTED BY JCTCIA MUST BE SUPPLIED IN THE
C           COMMON STATEMENT.
C           TESTING FOR PRIVATE TERMINATION CELLS (PTC) IS, FOR FOWARD
C           ONLY LISTS, ACCOMPLISHED AUTOMATICALLY.  COPYING FROM A PTC
C           IS NOT A FATAL ERROR, THOUGH IT COULD BE MADE SO BY SETTING
C           JCCORE(14) EQUAL TO +1.
C
C       SUBROUTINE AND FUNCTION SUBPROGRAMS REQUIRED
C           J5TST1
C           J5TST2
C           J5TST4
C           LAND
C           LEOR
C           LOR
C           SETLIN
C           SYSENT
C           SYSEXT
C
C       ERROR CODES FOR THIS SUBPROGRAM
C           NONE
C
C       METHOD
```

```
C           THE VECTOR LOCAL DETERMINES THE DATA ITEMS TO BE COPIED.
C
C                                  ********
C
      SUBROUTINE COPY(IAUASL,JCPOFP,JCPOSP,LOCAL,JCCLNO)
      COMMON/ALLOC/JCCORE(1)
      DIMENSION LOCAL(1),IAUASL(5)
      CALL SYSENT(1,1,3,2,JCCLNO)
      JCTOLD=IAUASL(5)
      JCSIZE=IAUASL(4)
C                 ARE JCPOFP AND JCPOSP LEGAL
      CALL J5TST1(JCPOFP,1)
      CALL J5TST1(JCPOSP,2)
      JCOUM1=JCCORE(JCTOLD+20)
      IF(JCCORE(JCOUM1).EQ.2)GO TO 1
C                 TESTING JCPOFP AND JCPOSP FOR PTC
      CALL J5TST2(2,JCTOLD,JCPOFP,1)
      CALL J5TST2(1,JCTOLD,JCPOSP,2)
    1 JCOUM1=LOCAL(1)
C                 COPY NONE, ALL OR PART
      JCOUM2=J5TST4(JCOUM1,1)
      GO TO (104,2,5),JCOUM2
C                 COPY ALL, WORD FOR WORD
    2 DO 3 I=1,JCSIZE
      JCOUM2=JCPOSP+I-1
      JCOUM3=JCPOFP+I-1
    3 JCCORE(JCOUM2)=JCCORE(JCOUM3)
      GO TO 104
C                 COPY PART, ONE DATA TYPE AT A TIME
    5 DO 6 I=1,JCOUM1
C                 SETTING THE LIMITS FOR THE COPYING DO LOOP FOR THE
C                 CURRENT DATA TYPE
      CALL SETLIM(IAUASL,JCPOSP,LOCAL,I,JCMNLM,JCMXLM,JCSHFT,JCMASK,1)
      JCOUM2=JCSHFT*JCMASK
      JCOUM3=JCPOFP-JCPOSP
C                 THE COPYING OPERATION, ONE DATA ITEM AT A TIME FOR
C                 EACH DATA TYPE
      DO 6 J=JCMNLM,JCMXLM
      JCOUM5=J+JCOUM3
      JCOUM4=JCCORE(JCOUM5)
      IF(JCOUM2.NE.0)JCOUM4=LAND(JCOUM2,JCOUM4)
      JCOUM5=JCCORE(J)
      IF(JCOUM2.NE.0)JCOUM5=LAND(JCOUM2,JCOUM5)
      JCOUM5=LEOR(JCOUM5,JCCORE(J))
    6 JCCORE(J)=LOR(JCOUM4,JCOUM5)
  104 CALL SYSEXT
      RETURN
      END
```

```
C
C*****************************************************************
C                              **********
C                              * FRMCEL *
C                              **********
C
C       PURPOSE
C           THE PURPOSE OF FRMCEL IS TO EXTRACT ALL OR PART OF THE
C           INFORMATION STORED IN AN INDICATED LIST ENTRY.
C
C       USAGE
C           CALL FRMCEL(IAUASL,JCPOFP,LION,LOCAL,JCCLNO)
C
C       DATA FORMAT
C           N/A
C
C       DESCRIPTION OF PARAMETERS
C           IAUASL = A VECTOR DESCRIBED IN THE BSTLNK DOCUMENTATION
C           JCPOFP = THE POINTER TO THE FIRST WORD OF THE LIST ENTRY
C                    CONTAINING THE DESIRED INFORMATION
C           LION   = A VECTOR INTO WHICH THE DATA EXTRACTED FROM THE
C                    INDICATED LIST ENTRY IS STORED.  THE DATA ITEMS
C                    EXTRACTED ARE PLACED IN LION IN THE ORDER THEY
C                    ARE EXTRACTED.
C           LOCAL  = A VECTOR INDICATING THE DESIRED DATA ITEMS.  SEE
C                    THE DOCUMENTATION FOR COPY FOR A COMPLETE
C                    DESCRIPTION OF LOCAL.
C           JCCLNO = A USER SPECIFIED CALL STATEMENT ID. NUMBER
C
C       REMARKS
C           THE NUMBER REPRESENTED BY JCTCIA MUST BE SUPPLIED IN THE
C           COMMON STATEMENT.
C           REQUESTING INFORMATION FROM A PTC IS NOT FATAL.
C
C       SUBROUTINE AND FUNCTION SUBPROGRAMS REQUIRED
C           J5TST1
C           J5TST2
C           J5TST4
C           LAND
C           SETLIM
C           SYSENT
C           SYSEXT
C
C       ERROR CODES FOR THIS SUBPROGRAM
C           NONE
C
C       METHOD
C           THE VECTOR LOCAL DETERMINES THE DATA ITEMS WHICH ARE TO BE
C           EXTRACTED FROM THE LIST ENTRY AND REPRODUCED IN THE VECTOR
C           LION.
C
C                              **********
C
      SUBROUTINE FRMCEL(IAUASL,JCPOFP,LION,LOCAL,JCCLNO)
      COMMON/ALLOC/JCCORE(1)
      DIMENSION LION(1),LOCAL(1),IAUASL(5)
      CALL SYSENT(1,1,3,3,JCCLNO)
      JCTOLD=IAUASL(5)
      JCSIZE=IAUASL(4)
C                IS JCPOFP LEGAL
      CALL J5TST1(JCPOFP,1)
      JCDUM1=LOCAL(1)
```

```
      JCDUM2=JCCORE(JCTOLD+20)
C                   IS LIST STRUCTURE FORWARD OR FORWARD-BACKWARD
      IF(JCCORE(JCDUM2).EQ.2)GO TO 1
C                   IS JCPOFP A PTC
      CALL J5TST2(2,JCTOLD,JCPOFP,1)
C                   EXTRACT NONE, ALL OR PART
    1 JCDUM3=J5TST4(JCDUM1,1)
      GO TO (104,2,5),JCDUM3
C                   EXTRACT ALL, A WORD AT A TIME
    2 DO 3 I=1,JCSIZE
      JCDUM3=JCPOFP+I-1
    3 LION(I)=JCCORE(JCDUM3)
      GO TO 104
C                   EXTRACT PART, ONE DATA TYPE AT A TIME
    5 JCDUM2=0
      DO 6 I=1,JCDUM1
C                   SETTING THE LIMITS FOR THE DATA EXTRACTION DO LOOP
C                   FOR THE CURRENT DATA TYPE
      CALL SETLIM(IAUASL,JCPOFP,LOCAL,I,JCMNLM,JCMXLM,JCSHFT,JCMASK,1)
      DO 6 J=JCMNLM,JCMXLM
C                   THE DATA EXTRACTION OPERATION, ONE DATA ITEM AT A
C                   TIME FOR EACH DATA TYPE
      JCDUM2=JCDUM2+1
      LION(JCDUM2)=JCCORE(J)
    6 IF(JCMASK.NE.0)LION(JCDUM2)=LAND(JCMASK,LION(JCDUM2)/JCSHFT)
  104 CALL SYSEXT
      RETURN
      END
```

```
C
C*****************************************************************
C                              **********
C                              * FSTLNK *
C                              **********
C
C       PURPOSE
C           THIS SUBPROGRAM PERMITS THE SETTING OF THE FORWARD LINK
C           OF A GIVEN LIST ENTRY.
C
C       USAGE
C           CALL FSTLNK(IAUASL,JCPOFP,JCFLNK,JCCLNO)
C
C       DATA FORMAT
C           N/A
C
C       DESCRIPTION OF PARAMETERS
C           IAUASL = A VECTOR DESCRIBED IN THE BSTLNK DOCUMENTATION
C           JCPOFP = THE POINTER TO THE FIRST WORD OF THE LIST ENTRY
C           JCFLNK = THE VALUE AT WHICH THE FORWARD LINK IS TO BE SET
C           JCCLNO = A USER SPECIFIED CALL STATEMENT ID. NUMBER
C
C       REMARKS
C           THE NUMBER REPRESENTED BY JCTCIA MUST BE SUPPLIED IN THE
C           COMMON STATEMENT.
C
C       SUBROUTINE AND FUNCTION SUBPROGRAMS REQUIRED
C           J5SAME
C           J5TST1
C           SYSENT
C           SYSEXT
C
C       ERROR CODES FOR THIS SUBPROGRAM
C           NONE
C
C       METHOD
C           SELF EXPLANATORY
C
C                              **********
C
      SUBROUTINE FSTLNK(IAUASL,JCPOFP,JCFLNK,JCCLNO)
      COMMON/ALLOC/JCCORE(1)
      DIMENSION IAUASL(5)
      CALL SYSENT(1,1,3,4,JCCLNO)
C                 TEST FOR LEGAL JCPOFP
      CALL J5TST1(JCPOFP,1)
      JCTCLD=IAUASL(5)
      JCDUM1=JCCORE(JCTCLD+20)+1
C                 CALL J5SAME TO SET THE LINK
      CALL J5SAME(JCDUM1,JCFLNK,JCPOFP,1)
      CALL SYSEXT
      RETURN
      END
```

```
C
C*********************************************************************
C                                  **********
C                                  * J5COMP *
C                                  **********
C
C        PURPOSE
C            J5COMP COMPARES GIVEN DATA AGAINST THE CONTENTS OF A LIST
C            ENTRY AS PER A LOCAL VECTOR.
C
C        USAGE
C            CALL J5COMP(JCSPUR,JCPOFP,LION,LOCAL,JCDOFS,JCINOC,JCCLNO)
C
C        DATA FORMAT
C            N/A
C
C        DESCRIPTION OF PARAMETERS
C            JCINOC = 0 FOR MATCH FAILURE OR 1 FOR MATCH SUCCESS
C            FOR THE REMAINING PARAMETERS SEE LOCATE.  NOTE THAT JCSPUR
C            IS THE IAUASL.
C
C        REMARKS
C            THE NUMBER REPRESENTED BY JCTCIA MUST, FOR SOME MACHINES,
C            REPLACE THE 1 IN THE NAMED COMMON STATEMENT.
C
C        SUBROUTINE AND FUNCTION SUBPROGRAMS REQUIRED
C            J5TST4
C            SETLIM
C            SYSENT
C            SYSERR
C            SYSEXT
C
C        ERROR CODES FOR THIS PROGRAM
C            CODE FATAL  ERROR(DATA PROVIDED)
C            1    NF     LOCAL(1) = -1 (JCPOFP)
C
C        METHOD
C            SEE LOCATE.
C
C                                  **********
C
      SUBROUTINE J5COMP(JCSPUR,JCPOFP,LION,LOCAL,JCDOFS,JCINOC,JCCLNO)
      COMMON/ALLOC/JCCORE(1)
      DIMENSION LION(1),LOCAL(1),JCSPUR(1)
      INTEGER SFT
C                 ENTER
      CALL SYSENT(1,1,3,15,JCCLNO)
C                 NONE, ALL OR PART OF DATA
      JCINOC=0
      M1=LOCAL(1)
      M2=J5TST4(M1,1)
      M3=0
      GO TO (1,2,4),M2
C                 NONE
    1 CALL SYSERR(1,JCPOFP)
      GO TO 8
C                 ALL
    2 M1=JCSPUR(4)
      DO 3 I=1,M1
      M2=JCPOFP+I-1
    3 IF(JCCORE(M2).NE.LION(I))M3=1
      GO TO 7
```

```
C                   PART
    4 M4=0
      M5=0
      DO 6 I=1,M1
C                   SETTING LIMITS FOR DO LOOP
      CALL SETLIM(JCSPUR,JCPOFP,LOCAL,I,MIN,MAX,SFT,MSK,1)
C                   COMPARE
      DO 5 J=MIN,MAX
      M4=M4+1
      M6=JCCORE(J)
      IF(MSK.NE.0)M6=LAND(MSK,M6/SFT)
    5 IF(M6.NE.LION(M4))M3=1
      IF(JCDOFS.GT.0)GO TO 6
      IF((M5.EQ.0).AND.(M3.EQ.0))M5=I
      M3=0
    6 CONTINUE
C                   SUCCESS OR FAILURE
      IF((JCDOFS.LT.0).AND.(M5.EQ.0))GO TO 8
    7 IF(M3.NE.0)GO TO 8
C                   SUCCESS
      IF((JCDOFS.LT.0).AND.(M2.EQ.3))JCDOFS=M5
      JCINDC=1
C                   EXIT
    8 CALL SYSEXT
      RETURN
      END
```

```
C
C*******************************************************************
C                              **********
C                              * J5INDX *
C                              **********
C
C        PURPOSE
C            THE FUNCTION J5INDX DETERMINES THE USER AND RETURN MASK
C            WORD INDEX CORRESPONDING TO A SPECIFIED CORE SEGMENT.
C
C        USAGE
C            J5INDX(JCSNUM,JCCLNO)
C
C        DATA FORMAT
C            N/A
C
C        DESCRIPTION OF PARAMETERS
C            JCSNUM = THE SEGMENT NUMBER.  THE SEGMENTS ARE NUMBERED
C                     CONSECUTIVELY STARTING WITH THE FIRST SEGMENT OF
C                     THE ALLOCATABLE CORE.
C            JCCLNO = A USER SPECIFIED CALL STATEMENT ID. NUMBER
C
C        REMARKS
C            THE NUMBER REPRESENTED BY JCTCIA MUST BE SUPPLIED IN THE
C            COMMON STATEMENT.
C            THE ADVANCED USER MAY USE THIS FUNCTION WHEN ESTABLISHING
C            A USER RETURN MASK FOR PARTIAL CORE RETURN.  FOR THE MOST
C            PART THIS FUNCTION IS NOT INTENDED FOR USER UTILIZATION.
C
C        SUBROUTINE AND FUNCTION SUBPROGRAMS REQUIRED
C            SYSENT
C            SYSEXT
C
C        ERROR CODES FOR THIS SUBPROGRAM
C            NONE
C            SEE SUBPROGRAM ERROR FOR A COMPLETE SYSTEM ERROR LISTING.
C
C        METHOD
C            THE WORD INDEX IS BASED ON MASK WORDS CONTAINING ONE LESS
C            BIT THAN THE ACTUAL COMPUTER WORDS POSSESS.  THE FIRST WORD
C            FOR A MASK CARRIES THE LARGEST WORD INDEX, AND ITS BITS
C            CORRESPOND TO THE ALLOCATABLE SEGMENTS HAVING THE LARGEST
C            SEGMENT NUMBERS.  THE FIRST BIT OF MASK WORD INDEX 1
C            REPRESENTS THE FIRST ALLOCATABLE SEGMENT.
C
C                              **********
C
      FUNCTION J5INDX(JCSNUM,JCCLNO)
      COMMON/ALLOC/JCCORE(1)
      CALL SYSENT(1,1,3,16,JCCLNO)
C                  CALCULATION OF THE WORD INDEX
      J5INDX=JCSNUM/(JCCORE(1)-1)
      JCDUM1=J5INDX*(JCCORE(1)-1)
C                  ADJUST J5INDX IF CORRECT BIT IS THE LAST MASK WORD
C                  BIT
      IF(JCDUM1.EQ.JCSNUM)GO TO 104
      J5INDX=J5INDX+1
  104 CALL SYSEXT
      RETURN
      END
```

```
C
C*********************************************************************
C                                  **********
C                                  * J5IPOT *
C                                  **********
C
C        PURPOSE
C            THE FUNCTION J5IPOT CALCULATES THE INTEGER EXPONENT OF TWO
C            CORRESPONDING TO A SPECIFIED NUMBER OF SEGMENTS.
C
C        USAGE
C            J5IPOT(JCNOFS,JCLOST,JCCLNO)
C
C        DATA FORMAT
C            N/A
C
C        DESCRIPTION OF PARAMETERS
C            JCNOFS = NUMBER OF SEGMENTS
C            JCLOST = A PARAMETER USED TO INDICATE IF TWO TO THE J5IPOT
C                     POWER IS TO BE LESS THAN OR EQUAL TO JCNOFS, OR
C                     IS TO BE GREATER THAN OR EQUAL TO JCNOFS.  JCLOST
C                     WILL BE 1 IF THE FORMER IS DESIRED, OR 2 IF THE
C                     LATTER IS DESIRED.
C            JCCLNO = A USER SPECIFIED CALL STATEMENT ID. NUMBER
C
C        REMARKS
C            THE NUMBER REPRESENTED BY JCTCIA MUST BE SUPPLIED IN THE
C            COMMON STATEMENT.
C            THIS FUNCTION MAY BE UTILIZED BY THE USER.
C
C        SUBROUTINE AND FUNCTION SUBPROGRAMS REQUIRED
C            SYSENT
C            SYSEXT
C
C        ERROR CODES FOR THIS SUBPROGRAM
C            NONE
C
C        METHOD
C            SELF EXPLANATORY
C
C                                  **********
C
      FUNCTION J5IPOT(JCNOFS,JCLOST,JCCLNO)
      COMMON/ALLOC/JCCORE(1)
      CALL SYSENT(1,1,3,17,JCCLNO)
C                 CALCULATION OF J5IPOT
      J5IPOT=INT(ALOG(FLOAT(JCNOFS)*1.1)/ALOG(2.0))
      JCDUM1=2**J5IPOT
C                 IS SMALLER OR LARGER DESIRED
      GO TO (1,2),JCLOST
C                 SMALLER
    1 IF(JCDUM1.GT.JCNOFS)J5IPOT=J5IPOT-1
      GO TO 104
C                 LARGER
    2 IF(JCDUM1.LT.JCNOFS)J5IPOT=J5IPOT+1
  104 CALL SYSEXT
      RETURN
      END
```

```
C
C*****************************************************************
C                          **********
C                          * J5MUPD *
C                          **********
C
C       PURPOSE
C           THE SUBROUTINE J5MUPD UPDATES A USER CORE ASSIGNMENT MASK
C           UPON ADDITIONAL ALLOCATION OF CORE TO THE USER.
C
C       USAGE
C           CALL J5MUPD(JCSNUM,JCNOFS,JCFWOM,JCCLNO)
C
C       DATA FORMAT
C           N/A
C
C       DESCRIPTION OF PARAMETERS
C           JCSNUM = THE SEGMENT NUMBER OF THE FIRST SEGMENT OF THE
C                    BUDDY ISSUED TO LIST BY C8GIVE TO SATISFY THE
C                    USER'S REQUEST FOR CORE
C           JCNOFS = THE NUMBER OF SEGMENTS IN THE BUDDY
C           JCFWOM = THE FIRST WORD OF THE USER CORE ASSIGNMENT MASK
C           JCCLNO = A USER SPECIFIED CALL STATEMENT ID. NUMBER
C
C       REMARKS
C           THE NUMBER REPRESENTED BY JCTCIA MUST BE SUPPLIED IN THE
C           COMMON STATEMENT.
C           THIS SUBPROGRAM IS NOT INTENDED FOR USER UTILIZATION.
C
C       SUBRCUTINE AND FUNCTION SUBPROGRAMS REQUIRED
C           J5INDX
C           LOR
C           SYSENT
C           SYSEXT
C
C       ERROR CODES FOR THIS SUBPROGRAM
C           NONE
C
C       METHOD
C           THE USER'S MASK EMPLOYS WORDS CONSISTING OF ONE BIT PER
C           WORD LESS THAN THE ACTUAL COMPUTER WORDS POSSESS.
C
C                          **********
C
      SUBROUTINE J5MUPD(JCSNUM,JCNOFS,JCFWOM,JCCLNO)
      COMMON/ALLOC/JCCORE(1)
      CALL SYSENT(1,1,3,18,JCCLNO)
C                 OBTAIN MASK WORD INDEX
      JCDUM1=J5INDX(JCSNUM,1)
      JCDUM2=(JCCORE(1)-1)*(JCDUM1-1)
      JCDUM3=2**(JCSNUM-JCDUM2-1)
C                 HOW MANY BITS CAN BE SET IN THIS MASK WORD
      JCDUM4=JCCORE(1)-JCSNUM+JCDUM2
      JCDUM5=JCFWOM+JCCORE(8)-JCDUM1
      JCDUM2=0
    1 JCDUM6=JCNOFS-JCDUM2
C                 DOES THIS WORD CONTAIN ALL BITS WHICH MUST BE SET
      IF(JCDUM4.LT.JCDUM6)JCDUM6=JCDUM4
      JCDUM7=JCDUM3*(2**JCDUM6-1)
C                 MASK UPDATE ONE WORD AT A TIME
      JCCORE(JCDUM5)=LOR(JCCORE(JCDUM5),JCDUM7)
      JCDUM5=JCDUM5-1
```

```
      JCDUM2=JCDUM2+JCDUM6
      JCDUM3=1
      JCDUM4=JCCORE(1)-1
C                    MUST ADDITIONAL BITS BE SET
      IF(JCDUM2.LT.JCNOFS)GO TO 1
      CALL SYSEXT
      RETURN
      END
```

```
C
C*****************************************************************
C                              **********
C                              * J5NEXT *
C                              **********
C
C       PURPOSE
C           THE FUNCTION J5NEXT PROVIDES C8COMB WITH THE SEGMENT NUMBER
C           OF THE NEXT SEGMENT INDICATED ON THE RETURN MASK.
C
C       USAGE
C           J5NEXT(JCSNUM,JCFWOM,JCCLNO)
C
C       DATA FORMAT
C           N/A
C
C       DESCRIPTION OF PARAMETERS
C           JCSNUM = THE SEGMENT NUMBER OF THE FIRST SEGMENT TO BE
C                    CHECKED FOR OCCURRENCE ON THE RETURN MASK
C           JCFWOM = THE FIRST WORD OF THE RETURN MASK
C           JCCLNO = A USER SPECIFIED CALL STATEMENT ID. NUMBER
C
C       REMARKS
C           THE NUMBER REPRESENTED BY JCTCIA MUST BE SUPPLIED IN THE
C           COMMON STATEMENT.
C           JCFWOM IS A PASSED PARAMETER AS AN ADVANCED USER MAY
C           UTILIZE THIS FUNCTION FOR MASK MANIPULATIONS.
C
C       SUBROUTINE AND FUNCTION SUBPROGRAMS REQUIRED
C           J5INDX
C           LAND
C           SYSENT
C           SYSEXT
C
C       ERROR CODES FOR THIS SUBPROGRAM
C           NONE
C
C       METHOD
C           THE BITS OF THE RETURN MASK ARE INSPECTED INDIVIDUALLY
C           STARTING WITH THE BIT CORRESPONDING TO JCSNUM.  IF THE END
C           OF THE MASK IS REACHED WITHOUT FINDING A SEGMENT PRESENT,
C           J5NEXT IS SET AT ZERO.
C
C                              **********
C
      FUNCTION J5NEXT(JCSNUM,JCFWOM,JCCLNO)
      COMMON/ALLOC/JCCORE(1)
      CALL SYSENT(1,1,3,19,JCCLNO)
C                 DETERMINATION OF RETURN MASK WORD INDEX
      JCDUM1=J5INDX(JCSNUM,1)
      JCDUM2=(JCCORE(1)-1)*(JCDUM1-1)
C                 JCDUM3=BIT MASK
      JCDUM3=2**(JCSNUM-JCDUM2-1)
      JCDUM4=JCCORE(1)-JCSNUM+JCDUM2
      JCDUM5=JCFWOM+JCCORE(8)-JCDUM1
      J5NEXT=JCSNUM
      JCDUM6=(JCCORE(2)-JCCORE(9)+1)/JCCORE(3)
C                 HAS THE END OF THE RETURN MASK BEEN REACHED
    1 IF(J5NEXT.LE.JCDUM6)GO TO 2
C           THE END HAS BEEN REACHED
      J5NEXT=0
  104 CALL SYSEXT
```

```
      RETURN
C                     CHECKING A BIT OF THE RETURN MASK
    2 JCDUM7=LAND(JCCORE(JCDUM5),JCDUM3)
      IF(JCDUM7.NE.0)GO TO 104
C                     PREPARE TO CHECK NEXT BIT
      J5NEXT=J5NEXT+1
      JCDUM4=JCDUM4-1
      IF(JCDUM4.LE.0)GO TO 3
      JCDUM3=2*JCDUM3
      GO TO 1
C                     ADVANCING TO NEXT WORD OF RETURN MASK
    3 JCDUM3=1
      JCDUM4=JCCORE(1)-1
      JCDUM5=JCDUM5-1
      GO TO 1
      END
```

```
C
C*****************************************************************
C                              **********
C                              * J5NUMB *
C                              **********
C
C       PURPOSE
C           THE FUNCTION J5NUMB PROVIDES THE CALLING PROGRAM WITH THE
C           NUMBER OF UNITS CONTAINING AT LEAST A SPECIFIED NUMBER OF
C           ELEMENTS.
C
C       USAGE
C           J5NUMB(JCNUMR,JCONOM,JCCLNO)
C
C       DATA FORMAT
C           N/A
C
C       DESCRIPTION OF PARAMETERS
C           JCNUMR = THE NUMBER OF ELEMENTS
C           JCONOM = THE NUMBER OF ELEMENTS PER UNIT
C           JCCLNO = A USER SPECIFIED CALL STATEMENT ID. NUMBER
C
C       REMARKS
C           THE NUMBER REPRESENTED BY JCTCIA MUST BE SUPPLIED IN THE
C           COMMON STATEMENT.
C
C       SUBROUTINE AND FUNCTION SUBPROGRAMS REQUIRED
C           SYSENT
C           SYSEXT
C
C       ERROR CODES FOR THIS SUBPROGRAM
C           NONE
C
C       METHOD
C           SELF EXPLANATORY
C
C                              **********
C
      FUNCTION J5NUMB(JCNUMR,JCONOM,JCCLNO)
      COMMON/ALLOC/JCCORE(1)
      CALL SYSENT(1,1,3,20,JCCLNO)
      J5NUMB=JCNUMR/JCONOM
      JCDUM1=J5NUMB*JCONOM
      IF(JCNUMR.GT.JCDUM1)J5NUMB=J5NUMB+1
      CALL SYSEXT
      RETURN
      END
```

```
C
C*****************************************************************
C                              **********
C                              * J5REDY *
C                              **********
C
C       PURPOSE
C           THE SUBROUTINE J5REDY IS CALLED BY THE SUBPROGRAM LIST TO
C           PREPARE A FORWARD THREADED LIST FROM THE BUDDY OF CORE
C           PROVIDED BY CBGIVE.
C
C       USAGE
C           CALL J5REDY(JCNOCT,JCNOFS,JCSIZE,JCTOLD,JCHDCL,JCPHCL,
C           JCCLNO)
C
C       DATA FORMAT
C           N/A
C
C       DESCRIPTION OF PARAMETERS
C           JCNOCT = THE NUMBER OF CELLS THREADED BY J5REDY.  ON THE
C                    FIRST CALL, A VALUE FOR JCNOCT OF ZERO MUST BE
C                    PASSED.  A CELL IS A LIST ENTRY.
C           JCNOFS = THE NUMBER OF SEGMENTS IN THE BUDDY
C           JCSIZE = THE NUMBER OF WORDS PER LIST ENTRY
C           JCTOLD = THE TYPE OF LIST EMPLOYED BY THE USER
C           JCHDCL = THE POINTER TO THE FIRST WORD OF THE BUDDY
C           JCPHCL = THE POINTER TO THE FIRST WORD OF THE PREVIOUSLY
C                    THREADED BUDDY, OR TO THE HEADCELL OF THE USER'S
C                    AVAILABLE SPACE LIST.
C           JCCLNO = A USER SPECIFIED CALL STATEMENT ID. NUMBER
C
C       REMARKS
C           THE NUMBER REPRESENTED BY JCTCIA MUST BE SUPPLIED IN THE
C           COMMON STATEMENT.
C           THIS SUBPROGRAM IS NOT RECOMMENDED FOR USER UTILIZATION.
C
C       SUBROUTINE AND FUNCTION SUBPROGRAMS REQUIRED
C           J5SAME
C           SYSENT
C           SYSEXT
C
C       ERROR CODES FOR THIS SUBPROGRAM
C           NONE
C
C       METHOD
C           THE LIST ENTRIES ARE PREPARED ACCORDING TO USER
C           SPECIFICATIONS AND FORWARD THREADED.
C
C                              **********
C
      SUBROUTINE J5REDY(JCNOCT,JCNOFS,JCSIZE,JCTOLD,JCHDCL,JCPHCL,
     1 JCCLNO)
      COMMON/ALLOC/JCCORE(1)
      CALL SYSENT(1,1,3,21,JCCLNO)
C                 HOW MANY LIST ENTRIES ARE CONTAINED IN THE BUDDY
      JCDUM1=(JCNOFS*JCCORE(3))/JCSIZE
      IF(JCDUM1.EQ.0)GO TO 104
      JCNOCT=JCNOCT+JCDUM1
      JCDUM2=JCCORE(JCTOLD+20)+1
      JCDUM3=JCHDCL
      IF(JCDUM1.EQ.1)GO TO 2
      JCDUM1=JCDUM1-1
```

```
C                   LINKING THE LIST ENTRIES CONTAINED IN THE BUDDY
      DO 1 I=1,JCDUM1
      JCDUM4=JCDUM3
      JCDUM3=JCDUM3+JCSIZE
    1 CALL J5SAME(JCDUM2,JCDUM3,JCDUM4,1)
C                   LINKING CURRENT LIST ENTRIES TO THOSE ALREADY
C                   EXISTING
    2 CALL J5SAME(JCDUM2,JCPHCL,JCDUM3,2)
      JCPHCL=JCHDCL
  104 CALL SYSEXT
      RETURN
      END
```

```
C
C*******************************************************************
C                              **********
C                              * J5SAME *
C                              **********
C
C       PURPOSE
C           THE SUBROUTINE J5SAME SETS AN INDICATED LINK AT A SPECIFIED
C           VALUE.
C
C       USAGE
C           CALL J5SAME(JCFWDL,JCLINK,JCPOFP,JCCLNO)
C
C       DATA FORMAT
C           N/A
C
C       DESCRIPTION OF PARAMETERS
C           JCFWDL = THE POINTER TO THE FIRST OF FOUR WORDS DESCRIBING
C                    THE LINK LOCATION WITHIN A LIST ENTRY
C           JCLINK = THE VALUE TO WHICH THE LINK IS TO BE SET
C           JCPOFP = THE POINTER TO THE FIRST WORD OF THE LIST ENTRY
C           JCCLNO = A USER SPECIFIED CALL STATEMENT ID. NUMBER
C
C       REMARKS
C           THE NUMBER REPRESENTED BY JCTCIA MUST BE SUPPLIED IN THE
C           COMMON STATEMENT.
C           THIS SUBPROGRAM IS NOT INTENDED FOR USER UTILIZATION.
C
C       SUBROUTINE AND FUNCTION SUBPROGRAMS REQUIRED
C           J5SAVE
C           LEOR
C           LOR
C           SYSENT
C           SYSEXT
C
C       ERROR CODES FOR THIS SUBPROGRAM
C           NONE
C
C       METHOD
C           THE LINK LOCATION ARRAY PROVIDES THE WORD INDEX, SHIFT AND
C           BIT MASK NECESSARY TO LOCATE THE BITS OF THE LIST ENTRY
C           SERVING AS THE LINK.
C
C                              **********
C
      SUBROUTINE J5SAME(JCFWDL,JCLINK,JCPOFP,JCCLNO)
      COMMON/ALLOC/JCCORE(1)
      CALL SYSENT(1,1,3,22,JCCLNO)
C                 SHIFTING LINK TO CORRECT POSITION
      JCDUM1=JCLINK*JCCORE(JCFWDL+2)
      JCDUM2=JCPOFP+JCCORE(JCFWDL)-1
C                 SECURING CURRENT LINK
      JCDUM3=J5SAVE(JCFWDL,JCDUM2,1)*JCCORE(JCFWDL+2)
C                 REMOVING CURRENT LINK
      JCDUM3=LEOR(JCDUM3,JCCORE(JCDUM2))
C                 INSTALLING NEW LINK
      JCCORE(JCDUM2)=LOR(JCDUM1,JCDUM3)
      CALL SYSEXT
      RETURN
      END
```

```
C
C*****************************************************************
C                              **********
C                              * J5SAVE *
C                              **********
C
C        PURPOSE
C            THE PURPOSE OF THE FUNCTION J5SAVE IS TO RETRIEVE A LINK
C            FROM A LIST ENTRY.
C
C        USAGE
C            J5SAVE(JCFWOL,JCWORD,JCCLNO)
C
C        DATA FORMAT
C            N/A
C
C        DESCRIPTION OF PARAMETERS
C            JCFWOL = THE POINTER TO THE FIRST OF FOUR WORDS DESCRIBING
C                     THE LINK LOCATION WITHIN A LIST ENTRY
C            JCWORD = THE POINTER TO THE LIST ENTRY WORD CONTAINING THE
C                     LINK
C            JCCLNO = A USER SPECIFIED CALL STATEMENT ID. NUMBER
C
C        REMARKS
C            THE NUMBER REPRESENTED BY JCTCIA MUST BE SUPPLIED IN THE
C            COMMON STATEMENT.
C            THIS FUNCTION IS NOT INTENDED FOR USER UTILIZATION.
C
C        SUBROUTINE AND FUNCTION SUBPROGRAMS REQUIRED
C            LAND
C            SYSENT
C            SYSEXT
C
C        ERROR CODES FOR THIS SUBPROGRAM
C            NONE
C
C        METHOD
C            THE LINK LOCATION ARRAY PROVIDES THE SHIFT AND BIT MASK
C            NECESSARY TO LOCATE THE BITS OF THE LIST ENTRY WORD
C            SERVING AS THE LINK.
C
C                              **********
C
      FUNCTION J5SAVE(JCFWOL,JCWORD,JCCLNO)
      COMMON/ALLOC/JCCORE(1)
      CALL SYSENT(1,1,3,23,JCCLNO)
      JCOUM1=JCCORE(JCFWOL+3)
C                RETRIEVING THE LINK
      J5SAVE=JCCORE(JCWORD)
      IF(JCOUM1.NE.0)J5SAVE=LAND(JCOUM1,J5SAVE/JCCORE(JCFWOL+2))
      CALL SYSEXT
      RETURN
      END
```

```
C
C*****************************************************************
C                              **********
C                              * J5TST1 *
C                              **********
C
C       PURPOSE
C           THE PURPOSE OF THIS SUBPROGRAM IS TO TEST THE POINTER TO
C           THE FIRST WORD OF A LIST ENTRY, BUDDY OR SEGMENT FOR BEING
C           GREATER THAN OR EQUAL TO THE FIRST ALLOCATABLE WORD OF CORE
C           AND FOR BEING GREATER THAN THE LAST WORD OF COMMON CORE
C
C       USAGE
C           CALL J5TST1(JCPOFP,JCCLNO)
C
C       DATA FORMAT
C           N/A
C
C       DESCRIPTION OF PARAMETERS
C           JCPOFP = THE POINTER TO THE FIRST WORD OF A LIST ENTRY,
C                    BUDDY OR SEGMENT
C           JCCLNO = A USER SPECIFIED CALL STATEMENT ID. NUMBER
C
C       REMARKS
C           THE NUMBER REPRESENTED BY JCTCIA MUST BE SUPPLIED IN THE
C           COMMON STATEMENT.
C           THIS SUBPROGRAM MAY BE UTILIZED BY THE USER.
C
C       SUBROUTINE AND FUNCTION SUBPROGRAMS REQUIRED
C           SYSENT
C           SYSERR
C           SYSEXT
C
C       ERROR CODES FOR THIS SUBPROGRAM
C           CODE FATAL  ERROR(DATA PROVIDED)
C           1    F      ILLEGAL JCPOFP (JCPOFP)
C
C       METHOD
C           SELF EXPLANATORY
C
C                              **********
C
      SUBROUTINE J5TST1(JCPOFP,JCCLNO)
      COMMON/ALLOC/JCCORE(1)
      CALL SYSENT(1,1,3,24,JCCLNO)
      IF((JCPOFP.LT.JCCORE(9)).OR.(JCPOFP.GT.JCCORE(2)))CALL SYSERR(
     1 -1,JCPOFP)
      CALL SYSEXT
      RETURN
      END
```

```
C
C*****************************************************************
C                              **********
C                              * J5TST2 *
C                              **********
C
C       PURPOSE
C           THE PURPOSE OF SUBROUTINE J5TST2 IS TO TEST A LIST ENTRY
C           FOR A FORWARD LINK OF +1 INDICATING IT TO BE A PTC.
C
C       USAGE
C           CALL J5TST2(JCWRIT,JCTOLD,JCPOFP,JCCLNO)
C
C       DATA FORMAT
C           N/A
C
C       DESCRIPTION OF PARAMETERS
C           JCWRIT = A PARAMETER EITHER 1 OR 2 AS DATA IS DIRECTED TO
C                    OR FROM A LIST ENTRY RESPECTIVELY
C           JCTOLD = A PARAMETER INDICATING THE TYPE OF LIST EMPLOYED
C           JCPOFP = THE POINTER TO THE FIRST WORD OF THE LIST ENTRY
C           JCCLNO = A USER SPECIFIED CALL STATEMENT ID. NUMBER
C
C       REMARKS
C           THE NUMBER REPRESENTED BY JCTCIA MUST BE SUPPLIED IN THE
C           COMMON STATEMENT.
C           THIS SUBPROGRAM MAY BE UTILIZED BY THE USER WITH DUE CARE
C           TO PROVIDE THE APPROPRIATE JCWRIT.
C
C       SUBROUTINE AND FUNCTION SUBPROGRAMS REQUIRED
C           ERROR
C           J5SAME
C           J5SAVE
C           SYSENT
C           SYSERR
C           SYSEXT
C
C       ERROR CODES FOR THIS SUBPROGRAM
C           CODE FATAL  ERROR(DATA PROVIDED)
C           1    NF     JCPOFP IS A PTC (JCPOFP)
C
C       METHOD
C           A FORWARD LINK OF +1 INDICATES A PTC.
C
C                              **********
C
      SUBROUTINE J5TST2(JCWRIT,JCTOLD,JCPOFP,JCCLNO)
      COMMON/ALLOC/JCCORE(1)
      CALL SYSENT(1,1,3,25,JCCLNO)
      JCDUM1=JCCORE(JCTOLD+20)+1
      JCDUM2=JCPOFP+JCCORE(JCDUM1)-1
C                 SECURING THE FORWARD LINK
      JCDUM3=J5SAVE(JCDUM1,JCDUM2,1)
C                 IS THE LIST ENTRY A PTC
      IF(JCDUM3.NE.1)GO TO 104
C                 CALL ERROR IF JCWRIT = 2
      IF(JCWRIT.EQ.2)CALL SYSERR(1,JCPOFP)
C                 SET FORWARD LINK TO 0 IF JCWRIT = 1
      IF(JCWRIT.EQ.1)CALL J5SAME(JCDUM1,0,JCPOFP,1)
  104 CALL SYSEXT
      RETURN
      END
```

```
C
C*****************************************************************
C                              **********
C                              * J5TST3 *
C                              **********
C
C       PURPOSE
C           THE PURPOSE OF J5TST3 IS TO TEST A LIST TYPE FOR A FORWARD-
C           BACKWARD LINK STRUCTURE.
C
C       USAGE
C           CALL J5TST3(JCTOLO,JCPOFP,JCCLNO)
C
C       DATA FORMAT
C           N/A
C
C       DESCRIPTION OF PARAMETERS
C           JCTOLO = A PARAMETER INDICATING THE TYPE OF LIST EMPLOYED
C           JCPOFP = THE POINTER TO THE FIRST WORD OF AN APPROPRIATE
C                    LIST ENTRY
C           JCCLNO = A USER SPECIFIED CALL STATEMENT ID. NUMBER
C
C       REMARKS
C           THE NUMBER REPRESENTED BY JCTCIA MUST BE SUPPLIED IN THE
C           COMMON STATEMENT.
C           THIS SUBPROGRAM MAY BE UTILIZED BY THE USER.
C
C       SUBROUTINE AND FUNCTION SUBPROGRAMS REQUIRED
C           J5TST6
C           SYSENT
C           SYSERR
C           SYSEXT
C
C       ERROR CODES FOR THIS SUBPROGRAM
C           CODE FATAL  ERROR(DATA PROVIDED)
C           1    F      FORWARD-ONLY LIST (JCTOLD)
C
C       METHOD
C           THE FIRST WORD OF THE LINK LOCATION ARRAY FOR EACH LIST
C           TYPE IS 1 OR 2 AS THE LIST IS FORWARD ONLY OR FORWARD-
C           BACKWARD RESPECTIVELY.
C
C                              **********
C
      SUBROUTINE J5TST3(JCTOLD,JCPOFP,JCCLNO)
      COMMON/ALLOC/JCCORE(1)
      CALL SYSENT(1,1,3,26,JCCLNO)
      JCDUM1=J5TST6(JCTOLD,1)
      IF(JCDUM1.EQ.1)CALL SYSERR(-1,JCTOLD)
      CALL SYSEXT
      RETURN
      END
```

```
C
C*****************************************************************
C                              **********
C                              * J5TST4 *
C                              **********
C
C       PURPOSE
C           THE PURPOSE OF THE FUNCTION J5TST4 IS TO DIRECT A DATA
C           TRANSFER OPERATION TO TRANSFER NONE, ALL OR PART OF THE
C           INFORMATION AS THE VALUE OF J5TST4 IS 1, 2 OR 3
C           RESPECTIVELY.
C
C       USAGE
C           J5TST4(JCCANP,JCCLNO)
C
C       DATA FORMAT
C           N/A
C
C       DESCRIPTION OF PARAMETERS
C           JCCANP = THE FIRST WORD OF THE LOCAL ARRAY
C           JCCLNO = A USER SPECIFIED CALL STATEMENT ID. NUMBER
C
C       REMARKS
C           THE NUMBER REPRESENTED BY JCTCIA MUST BE SUPPLIED IN THE
C           COMMON STATEMENT.
C           J5TST4 IS NOT RECOMMENDED FOR USER UTILIZATION.
C
C       SUBROUTINE AND FUNCTION SUBPROGRAMS REQUIRED
C           SYSENT
C           SYSEXT
C
C       ERROR CODES FOR THIS SUBPROGRAM
C           NONE
C
C       METHOD
C           LOCAL(1) MAY BE -1, 0 OR ANY POSITIVE NUMBER, WHICH J5TST4
C           TRANSLATES TO 1, 2 OR 3 RESPECTIVELY.
C
C                              **********
C
      FUNCTION J5TST4(JCCANP,JCCLNO)
      COMMON/ALLOC/JCCORE(1)
      CALL SYSENT(1,1,3,27,JCCLNO)
      J5TST4=3
      IF(JCCANP.LT.0)J5TST4=1
      IF(JCCANP.EQ.0)J5TST4=2
      CALL SYSEXT
      RETURN
      END
```

```
C
C*****************************************************************
C                              **********
C                              * J5TST5 *
C                              **********
C
C       PURPOSE
C           THE SUBPROGRAM J5TST5 TRACES THROUGH A LIST USING THE
C           INDICATED LINK, CHECKING THE CONTINUITY OF THE LIST BETWEEN
C           TWO INDICATED LIST ENTRIES.
C
C       USAGE
C           CALL J5TST5(JCPOFP,JCPOSP,JCTOLD,JCWLNK,JCCLNO)
C
C       DATA FORMAT
C           N/A
C
C       DESCRIPTION OF PARAMETERS
C           JCPOFP = THE POINTER TO THE FIRST WORD OF THE LEADING LIST
C                    ENTRY
C           JCPOSP = THE POINTER TO THE FIRST WORD OF THE TRAILING LIST
C                    ENTRY
C           JCTOLD = A PARAMETER INDICATING THE TYPE OF LIST EMPLOYED
C           JCWLNK = A PARAMETER EITHER 1 OR 2 AS RESPECTIVELY THE
C                    FORWARD OR BACKWARD LINK IS TO BE USED FOR THE
C                    TRACE.
C           JCCLNO = A USER SPECIFIED CALL STATEMENT ID. NUMBER
C
C       REMARKS
C           THE NUMBER REPRESENTED BY JCTCIA MUST BE SUPPLIED IN THE
C           COMMON STATEMENT.
C           THIS SUBPROGRAM MAY BE UTILIZED BY THE USER.
C           THE TERMS LEADING AND TRAILING ARE DEFINED WITH RESPECT TO
C           THE TRACE DIRECTION SPECIFIED BY JCWLNK.
C
C       SUBROUTINE AND FUNCTION SUBPROGRAMS REQUIRED
C           J5TST1
C           J5SAVE
C           SYSENT
C           SYSERR
C           SYSEXT
C
C       ERROR CODES FOR THIS SUBPROGRAM
C           CODE FATAL  ERROR(DATA PROVIDED)
C           1    F      ATTEMPTED BACKWARD TRACE ON FORWARD-ONLY LIST
C                       (JCTOLD)
C           2    F      LACK OF CONTINUITY (ADDRESS OF LAST LIST ENTRY)
C
C       METHOD
C           THE FIRST TEST PERFORMED WOULD EXPOSE A REQUEST FOR A
C           BACKWARD TRACE THROUGH A FORWARD ONLY LIST AND CARRIES THE
C           ERROR CODE JCERRC.  A TEST WITH THE ERROR CODE INCREMENTED
C           (OR DECREMENTED ACCORDING TO THE SIGN OF JCERRC) BY 1
C           CHECKS EACH LIST ENTRY FOR BEING THE LIST END.  A THIRD
C           TEST, THIS TIME WITH JCERRC INCREMENTED (OR DECREMENTED)
C           BY 2, CHECKS THE LEGALITY OF THE FIRST WORD OF EACH LIST
C           ENTRY ENCOUNTERED.
C
C                              **********
C
      SUBROUTINE J5TST5(JCPOFP,JCPOSP,JCTOLD,JCWLNK,JCCLNO)
      COMMON/ALLOC/JCCORE(1)
```

```
      CALL SYSENT(1,1,3,28,JCCLNO)
      JCDUM1=JCPOFP
      JCDUM2=JCCORE(JCTOLD+20)
C                   TESTING FOR LEGAL TRACE DIRECTION
      JCDUM3=JCCORE(JCDUM2)
      IF(JCDUM3.LT.JCWLNK)CALL SYSERR(-1,JCTOLD)
      JCDUM2=JCDUM2+1+4*(JCWLNK-1)
      CALL J5TST1(JCPOFP,1)
    1 JCDUM5=JCDUM1+JCCORE(JCDUM2)-1
C                       SECURING THE LINK
      JCDUM5=J5SAVE(JCDUM2,JCDUM5,1)
      IF(JCPOSP.GT.1)GO TO 2
      IF(JCDUM5.GT.1)GO TO 3
      JCPOSP=JCDUM1
      GO TO 104
C                   TESTING FOR PTC OR LIST END
    2 IF(JCDUM5.LE.1)CALL SYSERR(-2,JCDUM1)
      IF(JCDUM5.EQ.JCPOSP)GO TO 4
    3 JCDUM1=JCDUM5
C                   TESTING FOR LEGAL LIST ENTRY FIRST WORD
      CALL J5TST1(JCDUM1,2)
      GO TO 1
    4 CALL J5TST1(JCPOSP,3)
  104 CALL SYSEXT
      RETURN
      END
```

```
C
C*****************************************************************
C                              **********
C                              * J5TST6 *
C                              **********
C
C       PURPOSE
C           THE PURPOSE OF THE J5TST6 FUNCTION IS TO PROVIDE THE
C           CALLING PROGRAM WITH THE VALUE OF THE PARAMETER INDICATING
C           IF THE LIST IS FORWARD ONLY OR IS FORWARD-BACKWARD THREADED
C
C       USAGE
C           J5TST6(JCTOLD,JCCLNO)
C
C       DATA FORMAT
C           N/A
C
C       DESCRIPTION OF PARAMETERS
C           JCTOLD = THE PARAMETER INDICATING THE TYPE OF LIST EMPLOYED
C           JCCLNO = A USER SPECIFIED CALL STATEMENT ID. NUMBER
C
C       REMARKS
C           THE NUMBER REPRESENTED BY JCTCIA MUST BE SUPPLIED IN THE
C           COMMON STATEMENT.
C
C       SUBROUTINE AND FUNCTION SUBPROGRAMS REQUIRED
C           SYSENT
C           SYSEXT
C
C       ERROR CODES FOR THIS SUBPROGRAM
C           NONE
C
C       METHOD
C           THE FIRST WORD OF THE LINK LOCATION ARRAY FOR EACH LIST
C           TYPE IS 1 OR 2 AS THE LIST IS FORWARD ONLY OR FORWARD-
C           BACKWARD RESPECTIVELY.
C
C                              **********
C
      FUNCTION J5TST6(JCTOLD,JCCLNO)
      COMMON/ALLOC/JCCORE(1)
      CALL SYSENT(1,1,3,29,JCCLNO)
      J5TST6=JCCORE(JCTOLD+20)
      J5TST6=JCCORE(J5TST6)
      CALL SYSEXT
      RETURN
      END
```

```
C
C*****************************************************************
C                              **********
C                              * J5TST7 *
C                              **********
C
C       PURPOSE
C           THE FUNCTION J5TST7 TESTS THE RETURN MASK FOR THE PRESENCE
C           OF ALL BITS CORRESPONDING TO A PARTICULAR BUDDY.
C
C       USAGE
C           J5TST7(JCSNUM,JCNOFS,JCFWOM,JCCLNO)
C
C       DATA FORMAT
C           N/A
C
C       DESCRIPTION OF PARAMETERS
C           JCSNUM = THE SEGMENT NUMBER OF THE FIRST SEGMENT OF THE
C                    BUDDY
C           JCNOFS = THE NUMBER OF THE SEGMENTS IN THE BUDDY
C           JCFWOM = THE POINTER TO THE FIRST WORD OF THE RETURN MASK
C           JCCLNO = A USER SPECIFIED CALL STATEMENT IO. NUMBER
C
C       REMARKS
C           THE NUMBER REPRESENTED BY JCTCIA MUST BE SUPPLIED IN THE
C           COMMON STATEMENT.
C           THIS FUNCTION MAY BE UTILIZED BY THE USER AS JCSNUM NEED
C           NOT BE THE FIRST SEGMENT OF A BUDDY, NOR DOES JCFWOM NEED
C           BE THE POINTER TO THE FIRST WORD OF THE RETURN MASK.
C
C       SUBROUTINE AND FUNCTION SUBPROGRAMS REQUIRED
C           J5INDX
C           LAND
C           SYSENT
C           SYSEXT
C
C       ERROR CODES FOR THIS SUBPROGRAM
C           NONE
C
C       METHOD
C           J5TST7 IS VALUED AT 1 IF ALL SEGMENTS OF THE BUDDY ARE
C           PRESENT ON THE RETURN MASK, OTHERWISE J5TST7 IS 2.
C
C                              **********
C
      FUNCTION J5TST7(JCSNUM,JCNOFS,JCFWOM,JCCLNO)
      COMMON/ALLOC/JCCORE(1)
      CALL SYSENT(1,1,3,30,JCCLNO)
      J5TST7=1
C                   CALCULATING THE MASK WORD INDEX
      JCDUM1=J5INDX(JCSNUM,1)
      JCDUM2=(JCCORE(1)-1)*(JCDUM1-1)
C                   CALCULATING THE SHIFT
      JCDUM3=2**(JCSNUM-JCDUM2-1)
      JCDUM4=JCCORE(1)-JCSNUM+JCDUM2
      JCDUM5=JCFWOM+JCCORE(8)-JCDUM1
      JCDUM2=0
    1 JCDUM6=JCNOFS-JCDUM2
      IF(JCDUM4.LT.JCDUM6)JCDUM6=JCDUM4
C                   CALCULATING THE SHIFTED MASK
      JCDUM7=JCDUM3*(2**JCDUM6-1)
C                   THE TEST
```

```
      JCDUM8=LAND(JCDUM7,JCCORE(JCDUM5))
      IF(JCDUM8.EQ.JCCUM7)GO TO 2
C                   FAILURE
      J5TST7=2
      GO TO 104
    2 JCDUM2=JCDUM2+JCDUM6
      JCDUM3=1
      JCDUM4=JCCORE(1)-1
      JCDUM5=JCDUM5-1
C                   MUST ADDITIONAL MASK WORDS BE CHECKED
      IF(JCDUM2.LT.JCNOFS)GO TO 1
C                   SUCCESS
  104 CALL SYSEXT
      RETURN
      END
```

```
C
C*********************************************************************
C                              **********
C                              * LNKBWD *
C                              **********
C
C       PURPOSE
C           THE FUNCTION LNKBWD EXTRACTS FROM A SPECIFIED LIST ENTRY
C           THE BACKWARD LINK.
C
C       USAGE
C           LNKBWD(IAUASL,JCPOFP,JCCLNO)
C
C       DATA FORMAT
C           N/A
C
C       DESCRIPTION OF PARAMETERS
C           IAUASL = A VECTOR DESCRIBED IN THE BSTLNK DOCUMENTATION
C           JCPOFP = THE POINTER TO THE FIRST WORD OF THE LIST ENTRY
C           JCCLNO = A USER SPECIFIED CALL STATEMENT ID. NUMBER
C
C       REMARKS
C           THE NUMBER REPRESENTED BY JCTCIA MUST BE SUPPLIED IN THE
C           COMMON STATEMENT.
C
C       SUBROUTINE AND FUNCTION SUBPROGRAMS REQUIRED
C           J5SAVE
C           J5TST1
C           J5TST3
C           SYSENT
C           SYSEXT
C
C       ERROR CODES FOR THIS SUBPROGRAM
C           NONE
C
C       METHOD
C           THE LIST STRUCTURE MUST BE FORWARD-BACKWARD TO USE THIS
C           FUNCTION.
C
C                              **********
C
      FUNCTION LNKBWD(IAUASL,JCPOFP,JCCLNO)
      COMMON/ALLOC/JCCORE(1)
      DIMENSION IAUASL(5)
      CALL SYSENT(1,1,3,5,JCCLNO)
      JCTOLD=IAUASL(5)
C                 TEST FOR LEGAL JCPOFP
      CALL J5TST1(JCPOFP,1)
C                 TEST FOR FORWARD-BACKWARD LIST STRUCTURE
      CALL J5TST3(JCTOLD,JCPOFP,1)
      JCDUM1=JCCORE(JCTOLD+20)+5
      JCDUM2=JCPOFP+JCCORE(JCDUM1)-1
C                 CALL J5SAVE TO GET THE LINK
      LNKBWD=J5SAVE(JCDUM1,JCDUM2,1)
      CALL SYSEXT
      RETURN
      END
```

```
C
C*****************************************************************
C                              **********
C                              * LNKFWD *
C                              **********
C
C       PURPOSE
C           THE FUNCTION LNKFWD EXTRACTS FROM A SPECIFIED LIST ENTRY
C           THE FORWARD LINK.
C
C       USAGE
C           LNKFWD(IAUASL,JCPOFP,JCCLNO)
C
C       DATA FORMAT
C           N/A
C
C       DESCRIPTION OF PARAMETERS
C           IAUASL = A VECTOR DESCRIBED IN THE BSTLNK DOCUMENTATION
C           JCPOFP = THE POINTER TO THE FIRST WORD OF THE LIST ENTRY
C           JCCLNO = A USER ASSIGNED CALL STATEMENT ID. PARAMETER
C
C       REMARKS
C           THE NUMBER REPRESENTED BY JCTCIA MUST BE SUPPLIED IN THE
C           COMMON STATEMENT.
C
C       SUBROUTINE AND FUNCTION SUBPROGRAMS REQUIRED
C           J5SAVE
C           J5TST1
C           SYSENT
C           SYSEXT
C
C       ERROR CODES FOR THIS SUBPROGRAM
C           NONE
C
C       METHOD
C           FOR FORWARD ONLY LISTS, LNKFWD CHECKS TO SEE IF THE LIST
C           ENTRY FOLLOWING JCPOFP IS A PTC.  A PTC WOULD BE RETURNED
C           TO THE USER'S AVAILABLE SPACE LIST AND THE FORWARD LINK OF
C           JCPOFP WOULD BECOME ZERO.  THE VALUE OF LNKFWD IN THIS CASE
C           WOULD BE ZERO.
C
C                              **********
C
      FUNCTION LNKFWD(IAUASL,JCPOFP,JCCLNO)
      COMMON/ALLOC/JCCORE(1)
      DIMENSION IAUASL(5)
      CALL SYSENT(1,1,3,6,JCCLNO)
      JCTOLD=IAUASL(5)
C                 TEST FOR LEGAL JCPOFP
      CALL J5TST1(JCPOFP,1)
      JCDUM1=JCCORE(JCTOLD+20)+1
      JCDUM2=JCPOFP+JCCORE(JCDUM1)-1
C                 CALL J5SAVE TO GET THE LINK
      LNKFWD=J5SAVE(JCDUM1,JCDUM2,1)
      JCDUM2=JCCORE(JCDUM1-1)
C                 RETURN IF LIST IS FORWARD-BACKWARD OR IF JCPOFP IS
C                 THE LIST END OR A PTC
      IF((JCDUM2.EQ.2).OR.(LNKFWD.LE.1))GO TO 104
C                 TEST FOR LEGAL LNKFWD
      CALL J5TST1(LNKFWD,2)
      JCDUM2=LNKFWD + JCCORE(JCDUM1)-1
      JCDUM3=J5SAVE(JCDUM1,JCDUM2,2)
```

```
C                  RETURN IF LIST ENTRY FOLLOWING JCPOFP IS NOT A PTC
      IF(JCDUM3.NE.1)GO TO 104
C                  RETURNING THE PTC TO THE USER ASL
      CALL J5SAME(JCDUM1,IAUASL(2),LNKFWD,1)
      IAUASL(2)=LNKFWD
      CALL J5SAME(JCDUM1,0,JCPOFP,2)
      LNKFWD=0
  104 CALL SYSEXT
      RETURN
      END
```

```
C
C*****************************************************************
C                          **********
C                          * LOCATE *
C                          **********
C
C       PURPOSE
C           THE SUBROUTINE LOCATE SEARCHES A LIST IN A SPECIFIED
C           DIRECTION FOR THE LIST ENTRY CONTAINING CERTAIN DATA ITEMS
C           AT SPECIFIED VALUES.
C
C       USAGE
C           CALL LOCATE(IAUASL,JCPOFP,JCPOSP,LION,LOCAL,JCDOFS,JCCLNO)
C
C       DATA FORMAT
C           N/A
C
C       DESCRIPTION OF PARAMETERS
C           IAUASL = A VECTOR DESCRIBED IN THE BSTLNK DOCUMENTATION
C           JCPOFP = THE POINTER TO THE FIRST WORD OF THE LIST ENTRY AT
C                    WHICH THE SEARCH IS TO COMMENCE
C           JCPOSP = THE POINTER TO THE FIRST WORD OF THE LIST ENTRY
C                    CONTAINING THE SOUGHT AFTER DATA ITEMS, OR IF THE
C                    SEARCH FAILS ZERO
C           LION   = A VECTOR CONTAINING THE VALUES OF THE DATA ITEMS
C                    OF INTEREST
C           LOCAL  = A VECTOR INDICATING THE DESIRED DATA ITEMS.  SEE
C                    THE DOCUMENTATION FOR COPY FOR A COMPLETE
C                    DESCRIPTION OF LOCAL.
C           JCDOFS = A PARAMETER EQUAL IN MAGNITUDE TO 1, 2 OR 3 AS
C                    RESPECTIVELY IT IS DESIRED THE SEARCH PROCEED IN
C                    THE FORWARD DIRECTION ONLY, BACKWARD DIRECTION
C                    ONLY OR THE SEARCH IS TO ENCOMPASS ALL LIST
C                    ENTRIES ACCESSIBLE FROM JCPOFP.  IF JCDOFS IS
C                    NEGATIVE, THEN THE SEARCH IS SUCCESSFUL IF ANY
C                    DATA TYPE DESCRIBED IN THE LOCAL VECTOR MATCHES
C                    THE CORRESPONDING VALUES INDICATED IN LION.  HERE
C                    A DATA TYPE MEANS A THREE WORD DATA TYPE
C                    DESCRIPTION FROM LOCAL AND MAY INVOLVE SEVERAL
C                    WORDS OF DATA.  WHEN JCDOFS IS NEGATIVE, THE
C                    PARTICULAR DATA TYPE FOUND TO MATCH IS RETURNED TO
C                    THE USER VIA THE JCDOFS PARAMETER.  FOR A POSITIVE
C                    JCDOFS ALL DATA TYPES IN THE LOCAL ARRAY MUST
C                    MATCH AND THE PARAMETER JCDOFS REMAINS UNCHANGED.
C           JCCLNO = A USER SPECIFIED CALL STATEMENT ID. NUMBER
C
C       REMARKS
C           THE NUMBER REPRESENTED BY JCTCIA MUST BE SUPPLIED IN THE
C           COMMON STATEMENT.
C
C       SUBROUTINE AND FUNCTION SUBPROGRAMS REQUIRED
C           J5COMP
C           J5TST6
C           LAND
C           LNKBWD
C           LNKFWD
C           SETLIM
C           SYSENT
C           SYSERR
C           SYSEXT
C
C       ERROR CODES FOR THIS SUBPROGRAM
```

```
C            CODE FATAL  ERROR(DATA PROVIDED)
C            1    NF     JCPOFP IS A PTC (JCPOFP)
C
C        METHOD
C            THE SEARCH PROCEDURE IS INDICATED BY JCDOFS.  FOR FORWARD
C            ONLY LISTS A TEST FOR A PTC IS CONDUCTED ON EACH LIST ENTRY
C            ENCOUNTERED.  NO BACKWARD SEARCH ON A FORWARD ONLY LIST IS
C            POSSIBLE.  IF JCDOFS IS 3 IN MAGNITUDE THE SEARCH INITIALLY
C            PROCEEDS FORWARD AND IF UNSUCCESSFUL FOR A FORWARD-BACKWARD
C            LIST WILL SEARCH BACKWARD FROM THE SAME INITIAL STARTING
C            POINT.
C
C                                  **********
C
      SUBROUTINE LOCATE(IAUASL,JCPOFP,JCPOSP,LION,LOCAL,JCDOFS,JCCLNO)
      COMMON/ALLOC/JCCORE(1)
      DIMENSION LION(1),LOCAL(1),IAUASL(5)
      CALL SYSENT(1,1,3,7,JCCLNO)
      JCPOSP=0
      JCDU12=IABS(JCDOFS)
      JCDUM1=2
C                   IS LIST FORWARD ONLY OR FORWARD-BACKWARD
      JCDUM2=J5TST6(IAUASL(5),1)
      JCDUM3=JCPOFP
C                   SEARCH FORWARD OR BACKWARD
      IF(JCDU12.EQ.2)GO TO 1
C                   FORWARD
      JCDUM4=LNKFWD(IAUASL,JCDUM3,1)
C                   IS JCPOFP A PTC
      IF(JCDUM4.EQ.1)CALL SYSERR(1,JCPOFP)
      JCDUM1=1
      GO TO 2
C                   BACKWARD
    1 JCDUM4=LNKBWD(IAUASL,JCDUM3,1)
    2 CALL J5COMP(IAUASL,JCDUM3,LION,LOCAL,JCDOFS,JCDUM5,1)
      IF(JCDUM5.EQ.0)GO TO 10
C                   SUCCESS
      JCPOSP=JCDUM3
  104 CALL SYSEXT
      RETURN
C                   FAILURE
   10 IF(JCDUM4.LE.1)GO TO 13
C                   PREPARING TO INSPECT NEXT LIST ENTRY
      JCDUM3=JCDUM4
      IF(JCDUM1.EQ.2)GO TO 1
   11 JCDUM4=LNKFWD(IAUASL,JCDUM3,2)
      GO TO 2
C                   END OF LIST
   13 IF((JCDUM1.EQ.2).OR.(JCDU12.EQ.1).OR.(JCDUM2.EQ.1))GO TO 104
C                   REVERSE DIRECTION OF SEARCH
      JCDUM1=2
      JCDUM4=LNKBWD(IAUASL,JCPOFP,3)
      GO TO 10
      END
```

```
C
C*****************************************************************
C                              **********
C                              * NEWCEL *
C                              **********
C
C       PURPOSE
C           THE SUBPROGRAM NEWCEL PROVIDES THE CALLING PROGRAM WITH A
C           LIST ENTRY FOR ITS USE, WHILE PLACING THE FORWARD LINK OF
C           THIS LIST ENTRY AS THE HEADCELL OF THE USER'S AVAILABLE
C           SPACE LIST.
C
C       USAGE
C           CALL NEWCEL(IAUASL,JCNWCL,JCCLNO)
C
C       DATA FORMAT
C           N/A
C
C       DESCRIPTION OF PARAMETERS
C           IAUASL = A VECTOR DESCRIBED IN THE BSTLNK DOCUMENTATION
C           JCNWCL = THE POINTER TO THE FIRST WORD OF THE NEW LIST
C                    ENTRY
C           JCCLNO = A USER SPECIFIED CALL STATEMENT ID. NUMBER
C
C       REMARKS
C           THE NUMBER REPRESENTED BY JCTCIA MUST BE SUPPLIED IN THE
C           COMMON STATEMENT.
C
C       SUBROUTINE AND FUNCTION SUBPROGRAMS REQUIRED
C           LIST
C           LNKFWD
C           SYSENT
C           SYSEXT
C
C       ERROR CODES FOR THIS SUBPROGRAM
C           NONE
C
C       METHOD
C           NEWCEL SETS JCNWCL EQUAL TO THE AVAILABLE SPACE LIST
C           HEADCELL AND THEN REPLACES THE HEADCELL BY THE FORWARD LINK
C           OF THE FORMER HEADCELL.  IF THE ASL HEADCELL IS ZERO, THE
C           LIST SUBPROGRAM IS SUMMONED TO REPLENISH THE USER'S ASL.
C           THE FORWARD LINK, AND IF APPLICABLE THE BACKWARD LINK, OF
C           JCNWCL ARE SET AT ZERO.
C
C                              **********
C
      SUBROUTINE NEWCEL(IAUASL,JCNWCL,JCCLNO)
      COMMON/ALLOC/JCCORE(1)
      DIMENSION IAUASL(5)
      CALL SYSENT(1,1,3,8,JCCLNO)
    1 JCNWCL=IAUASL(2)
C                     IS THE ASL EMPTY
      IF(JCNWCL.NE.0)GO TO 2
C                     REPLENISHING THE ASL
      CALL LIST(IAUASL,1)
      GO TO 1
C                     SETTING THE ASL HEADCELL
    2 IAUASL(2)=LNKFWD(IAUASL,JCNWCL,1)
C                     ZERO EQCH WORD
      JCOUM1=JCNWCL+IAUASL(4)-1
      DO 3 I=JCNWCL,JCOUM1
```

```
3 JCCORE(I)=0
  CALL SYSEXT
  RETURN
  END
```

```
C
C*****************************************************************
C                              *********
C                              * POPUP *
C                              *********
C
C       PURPOSE
C           POPUP STORES DATA ITEMS OF A LIST ENTRY IN LION, RETURNS
C           ALL LIST ENTRIES FROM THIS TO A SECOND NAMED LIST ENTRY TO
C           THE USER'S AVAILABLE SPACE LIST, THEN PATCHES THE LINKS TO
C           PRESERVE THE CONTINUITY OF THE LIST.
C
C       USAGE
C           CALL POPUP(IAUASL,JCPOFP,JCPOSP,LION,LOCAL,JCWLNK,JCCLNO)
C
C       DATA FORMAT
C           N/A
C
C       DESCRIPTION OF PARAMETERS
C           IAUASL = A VECTOR DESCRIBED IN THE BSTLNK DOCUMENTATION
C           JCPOFP = THE POINTER TO THE FIRST WORD OF THE FIRST LIST
C                    ENTRY TO BE REMOVED FROM THE LIST
C           JCPOSP = THE POINTER TO THE FIRST WORD OF THE LAST LIST
C                    ENTRY TO BE REMOVED, OR 0 IF ALL LIST ENTRIES FROM
C                    JCPOFP TO THE LIST END ARE TO BE REMOVED
C           LION   = THE VECTOR INTO WHICH THE CONTENTS OF THE FIRST
C                    REMOVED LIST ENTRY ARE STORED
C           LOCAL  = A VECTOR INDICATING THE DESIRED DATA ITEMS.  SEE
C                    THE DOCUMENTATION FOR COPY FOR A COMPLETE
C                    DESCRIPTION OF LOCAL.
C           JCWLNK = A PARAMETER EITHER 1 OR 2 AS RESPECTIVELY JCPOFP
C                    PRECEEDS OR FOLLOWS JCPOSP ON THE LIST.  THUS
C                    JCWLNK INDICATES WHETHER THE FORWARD OR THE
C                    BACKWARD LINKS ARE TO BE USED FOR THE POPUP
C                    OPERATION.
C           JCCLNO = A USER SPECIFIED CALL STATEMENT ID. NUMBER
C
C       REMARKS
C           THE NUMBER REPRESENTED BY JCTCIA MUST BE SUPPLIED IN THE
C           COMMON STATEMENT.
C           FOR A FORWARD ONLY LIST, JCWLNK IS RESTRICTED TO BE 1 ONLY.
C
C       SUBROUTINE AND FUNCTION SUBPROGRAMS REQUIRED
C           BSTLNK
C           COPY
C           FRMCEL
C           FSTLNK
C           J5TST6
C           LNKBWD
C           LNKFWD
C           RETURN
C           SETLNK
C           SYSENT
C           SYSEXT
C
C       ERROR CODES FOR THIS SUBPROGRAM
C           NONE
C
C       METHOD
C           THE CONTENTS OF JCPOFP ARE REPRODUCED IN LION ACCORDING TO
C           LOCAL.  IF JCPOSP IS ZERO OR THE LAST LIST ENTRY, AND THE
C           LIST IS FORWARD ONLY LINKED, JCPOFP IS MADE A PTC AND ALL
```

```
C           LIST ENTRIES FOLLOWING JCPOFP ARE RETURNED TO THE USER ASL.
C           FOR THIS SAME JCPOSP WITH A FORWARD-BACKWARD LIST,THE ABOVE
C           STATEMENT APPLIES BUT WITH THE FOLLOWING CHANGE.  IF A LIST
C           ENTRY IS ADJACENT TO JCPOFP IN THE DIRECTION OPPOSITE THAT
C           INDICATED BY JCWLNK, THEN THIS ADJACENT LIST ENTRY IS
C           RETURNED, WITH ITS CONTENTS COPIED INTO JCPOFP.  IF THERE
C           IS NO ADJACENT LIST ENTRY, JCPOFP IS RETURNED.  FOR A
C           JCPOSP WHICH IS NOT A LIST END, THE CONTENTS OF THE LIST
C           ENTRY ADJACENT TO JCPOFP IN THE DIRECTION INDICATED BY
C           JCWLNK ARE COPIED INTO JCPOFP, AND ALL LIST ENTRIES FROM
C           THIS ADJACENT LIST ENTRY TO THE LIST ENTRY ADJACENT TO
C           JCPOSP IN THE DIRECTION INDICATED BY JCWLNK ARE RETURNED TO
C           THE USER ASL.
C
C                                 *********
C
      SUBROUTINE POPUP(IAUASL,JCPOFP,JCPOSP,LION,LOCAL,JCWLNK,JCCLNO)
      COMMON/ALLOC/JCCORE(1)
      DIMENSION LION(1),LOCAL(1),IAUASL(5)
      CALL SYSENT(1,1,3,9,JCCLNO)
C                 IS LIST FORWARD ONLY OR FORWARD-BACKWARD
      JCDUM1=JSTST6(IAUASL(5),1)
C                 LOAD DATA FROM JCPOFP INTO LION
      CALL FRMCEL(IAUASL,JCPOFP,LION,LOCAL,1)
      JCDUM6=LOCAL(1)
      LOCAL(1)=0
      GO TO (2,1),JCDUM1
    1 GO TO (5,8),JCWLNK
C                 LIST IS FORWARD ONLY
    2 JCDUM2=LNKFWD(IAUASL,JCPOFP,1)
      IF(JCDUM2.LE.1)GO TO 4
      IF(JCPOSP.EQ.0)GO TO 3
      JCDUM3=LNKFWD(IAUASL,JCPOSP,2)
      IF(JCDUM3.LE.1)GO TO 3
C                 TRANSFER ALL DATA FROM THE LIST ENTRY FOLLOWING
C                 JCPOSP INTO JCPOFP
      CALL COPY(IAUASL,JCDUM3,JCPOFP,LOCAL,1)
C                 RETURN ALL LIST ENTRIES FROM THE ONE FOLLOWING JCPOFP
C                 TO THE LIST ENTRY FOLLOWING JCPOSP
      CALL RETURN(IAUASL,JCDUM2,JCDUM3,1)
      GO TO 11
C                 RETURN ALL LIST ENTRIES FROM JCPOFP TO THE LIST END
    3 JCDUM3=0
      CALL RETURN(IAUASL,JCDUM2,JCDUM3,2)
C                 JCPOFP IS MADE A PTC
    4 CALL FSTLNK(IAUASL,JCPOFP,1,1)
      JCPOFP=0
      JCPOSP=0
      GO TO 11
C                 JCPOFP PRECEEDS JCPOSP
    5 JCDUM2=LNKFWD(IAUASL,JCPOFP,3)
      JCDUM3=LNKBWD(IAUASL,JCPOFP,1)
      IF((JCDUM2.EQ.0).OR.(JCPOSP.EQ.0))GO TO 6
      JCDUM4=LNKFWD(IAUASL,JCPOSP,4)
      IF(JCDUM4.EQ.0)GO TO 6
C                 RETURN ALL LIST ENTRIES FROM THE ONE FOLLOWING JCPOFP
C                 TO THE ENTRY FOLLOWING JCPOSP
      JCDUM5=LNKFWD(IAUASL,JCDUM4,5)
C                 TRANSFER ALL DATA FROM THE LIST ENTRY FOLLOWING
C                 JCPOSP INTO JCPOFP
      CALL COPY(IAUASL,JCDUM4,JCPOFP,LOCAL,2)
      CALL BSTLNK(IAUASL,JCPOFP,JCDUM3,1)
```

```
      IF(JCDUM5.NE.0)CALL BSTLNK(IAUASL,JCDUM5,JCPOFP,2)
      CALL RETURN(IAUASL,JCDUM2,JCDUM4,3)
      GO TO 11
    6 IF(JCDUM3.EQ.0)GO TO 7
C                     RETURN THE LIST ENTRY PRECEEDING JCPOFP, AND ALL LIST
C                     ENTRIES FROM THE ONE FOLLOWING JCPOFP TO THE LIST END
      JCDUM5=LNKBWD(IAUASL,JCDUM3,2)
      IF(JCDUM5.NE.0)CALL FSTLNK(IAUASL,JCDUM5,JCPOFP,2)
      CALL FSTLNK(IAUASL,JCDUM3,0,3)
C                     TRANSFER ALL DATA FROM THE LIST ENTRY PRECEEDING
C                     JCPOFP INTO JCPOFP
      CALL COPY(IAUASL,JCDUM3,JCPOFP,LOCAL,3)
      IF(JCDUM2.NE.0)CALL FSTLNK(IAUASL,JCDUM3,JCDUM2,4)
      JCDUM2=0
      CALL RETURN(IAUASL,JCDUM3,JCDUM2,4)
      GO TO 11
C                     RETURN THE ENTIRE LIST
    7 JCDUM2=0
      CALL RETURN(IAUASL,JCPOFP,JCDUM2,5)
      JCPOFP=0
      JCPCSP=0
      GO TO 11
C                     JCPOSP PRECEEDS JCPOFP
    8 JCDUM2=LNKBWD(IAUASL,JCPOFP,3)
      JCDUM3=LNKFWD(IAUASL,JCPOFP,6)
      IF((JCDUM2.EQ.0).OR.(JCPOSP.EQ.0))GO TO 9
      JCDUM4=LNKBWD(IAUASL,JCPOSP,4)
      IF(JCDUM4.EQ.0)GO TO 9
C                     RETURN ALL LIST ENTRIES FROM THE ONE PRECEEDING
C                     JCPOSP TO THE ENTRY PRECEEDING JCPOFP
      JCDUM5=LNKBWD(IAUASL,JCDUM4,5)
C                     TRANSFER ALL DATA FROM THE LIST ENTRY PRECEEDING
C                     JCPOSP INTO JCPOFP
      CALL COPY(IAUASL,JCDUM4,JCPOFP,LOCAL,4)
      CALL FSTLNK(IAUASL,JCPOFP,JCDUM3,5)
      IF(JCDUM5.NE.0)CALL FSTLNK(IAUASL,JCDUM5,JCPOFP,6)
      CALL RETURN(IAUASL,JCDUM4,JCDUM2,6)
      GO TO 11
    9 IF(JCDUM3.EQ.0)GO TO 10
C                     RETURN ALL LIST ENTRIES FROM THE LIST BEGINNING TO
C                     THE ENTRY PRECEEDING JCPOFP, AND RETURN THE LIST
C                     ENTRY FOLLOWING JCPOFP
      JCDUM5=LNKFWD(IAUASL,JCDUM3,7)
      IF(JCDUM5.NE.0)CALL BSTLNK(IAUASL,JCDUM5,JCPOFP,3)
      CALL BSTLNK(IAUASL,JCDUM3,0,4)
      CALL COPY(IAUASL,JCDUM3,JCPOFP,LOCAL,5)
      IF(JCDUM2.NE.0)CALL SETLNK(IAUASL,JCDUM2,JCDUM3,1)
      JCDUM2=0
      CALL RETURN(IAUASL,JCDUM2,JCDUM3,7)
      GO TO 11
C                     RETURN THE ENTIRE LIST
   10 JCDUM2=0
      CALL RETURN(IAUASL,JCDUM2,JCPOFP,8)
      JCPOFP=0
   11 LOCAL(1)=JCDUM6
      CALL SYSEXT
      RETURN
      END
```

```
C
C*****************************************************************
C                              ********
C                              * PUSH *
C                              ********
C
C       PURPOSE
C           THE PUSH SUBPROGRAM INSERTS A NEW CELL INTO A LIST AND
C           SUPPLIES IT WITH DATA FROM A DATA STORAGE VECTOR.
C
C       USAGE
C           CALL PUSH(IAUASL,JCPOFP,LION,LOCAL,JCPBOA,JCCLNO)
C
C       DATA FORMAT
C           N/A
C
C       DESCRIPTION OF PARAMETERS
C           IAUASL = A VECTOR DESCRIBED IN THE BSTLNK DOCUMENTATION
C           JCPOFP = THE POINTER TO THE FIRST WORD OF THE LIST ENTRY TO
C                    WHICH THE NEW LIST ENTRY IS TO BE ADJCACENT
C           LION   = THE VECTOR CONTAINING THE DATA FOR THE NEW LIST
C                    ENTRY
C           LOCAL  = A VECTOR INDICATING THE DESIRED DATA ITEMS.  SEE
C                    THE DOCUMENTATION FOR COPY FOR A COMPLETE
C                    DESCRIPTION OF LOCAL.
C           JCPBOA = A PARAMETER VALUED AT -2,-1,1, OR 2 AS IT IS
C                    DESIRED THAT THE OPERATION PERFORMED BE INSERT
C                    BEFORE, PUSH UP, PUSH DOWN OR INSERT AFTER
C                    RESPECTIVELY.
C           JCCLNO = A USER SPECIFIED CALL STATEMENT ID. NUMBER
C
C       REMARKS
C           THE NUMBER REPRESENTED BY JCTCIA MUST BE SUPPLIED IN THE
C           COMMON STATEMENT.
C           FOR A FORWARD ONLY LINKED LIST, JCPBOA IS RESTRICTED TO
C           POSITIVE VALUES ONLY.
C
C       SUBROUTINE AND FUNCTION SUBPROGRAMS REQUIRED
C           BSTLNK
C           COPY
C           FSTLNK
C           J5TST6
C           LNKBWD
C           LNKFWD
C           NEWCEL
C           SETLNK
C           SYSENT
C           SYSEXT
C           TOCELL
C
C       ERROR CODES FOR THIS SUBPROGRAM
C           NONE
C
C       METHOD
C           FOR THE INSERT MODE OF OPERATION A JCNWCL IS PROCURED AND
C           PLACED IN THE LIST EITHER BEFORE OR AFTER JCPOFP AS JCPBOA
C           IS +2 OR -2 RESPECTIVELY.  JCNWCL RECEIVES THE INFORMATION
C           STORED IN LION.  WHEN JCPBOA IS UNITY IN MAGNITUDE, THE
C           INFORMATION IN JCPOFP IS COPIED INTO JCNWCL AND THE
C           INFORMATION IN LION IS TRANSFERRED TO JCPOFP.  THE JCNWCL
C           IS THREADED INTO THE LIST AFTER JCPOFP FOR JCPBOA = +1 AND
C           BEFORE JCPOFP FOR JCPBOA = -1.  IF FOR A FORWARD ONLY LIST
```

```
C           JCPOFP IS A PTC, THE FORWARD LINK OF JCPOFP IS SET AT ZERO
C           AND THE INFORMATION IN LION TRANSFERRED TO JCPOFP.
C
C                                 ********
C
      SUBROUTINE PUSH(IAUASL,JCPOFP,LION,LOCAL,JCPBOA,JCCLNO)
      COMMON/ALLOC/JCCORE(1)
      DIMENSION LION(1),LOCAL(1),IAUASL(5)
      CALL SYSENT(1,1,3,10,JCCLNO)
      JCDUM2=2
      JCDUM4=5
      IF(JCPOFP.EQ.0)GO TO 2
C                 IS LIST FORWARD ONLY OR FORWARD-BACKWARD
      JCDUM1=J5TST6(IAUASL(5),1)
      JCDUM2=IABS(JCPBOA)
      JCDUM3=LOCAL(1)
      IF(JCPBOA.GE.0)GO TO 1
C                 BACKWARD LINK
      JCDUM4=6
      JCDUM5=LNKBWD(IAUASL,JCPOFP,1)
      IF(JCDUM2.EQ.2)JCDUM4=7
      GO TO 2
C                 FORWARD LINK
    1 JCDUM4=JCDUM2+2*JCDUM1-2
      JCDUM5=LNKFWD(IAUASL,JCPOFP,1)
      IF(JCDUM5.NE.1)GO TO 2
      JCDUM4=5
      GO TO 4
C                 OBTAIN A NEW CELL
    2 CALL NEWCEL(IAUASL,JCNWCL,1)
      IF(JCPOFP.EQ.0)JCPOFP=JCNWCL
      JCDUM6=JCNWCL
      GO TO (3,5),JCDUM2
C                 PUSH
    3 LOCAL(1)=0
      CALL COPY(IAUASL,JCPOFP,JCNWCL,LOCAL,1)
      LOCAL(1)=JCDUM3
    4 JCDUM6=JCPOFP
C                 INSERT
    5 CALL TOCELL(IAUASL,JCDUM6,LION,LOCAL,1)
      GO TO (7,6,9,11,8,12,14),JCDUM4
C                 INSERT AFTER, FORWARD ONLY LIST
    6 CALL FSTLNK(IAUASL,JCNWCL,JCDUM5,1)
C                 PUSH DOWN, FORWARD ONLY LIST
    7 CALL FSTLNK(IAUASL,JCPOFP,JCNWCL,2)
C                 REACTIVATION OF A PTC
    8 CALL SYSEXT
      RETURN
C                 PUSH DOWN, FORWARD-BACKWARD LIST
    9 IF(JCDUM5.NE.0)CALL BSTLNK(IAUASL,JCDUM5,JCNWCL,1)
   10 CALL SETLNK(IAUASL,JCPOFP,JCNWCL,1)
      GO TO 8
C                 INSERT AFTER, FORWARD-BACKWARD LIST
   11 IF(JCDUM5.NE.0)CALL SETLNK(IAUASL,JCNWCL,JCDUM5,2)
      GO TO 10
C                 PUSH UP
   12 IF(JCDUM5.NE.0)CALL FSTLNK(IAUASL,JCDUM5,JCNWCL,3)
   13 CALL SETLNK(IAUASL,JCNWCL,JCPOFP,3)
      GO TO 8
C                 INSERT BEFORE
   14 IF(JCDUM5.NE.0)CALL SETLNK(IAUASL,JCDUM5,JCNWCL,4)
      GO TO 13
```

```
END
```

```
C
C*****************************************************************
C                              **********
C                              * RETURN *
C                              **********
C
C       PURPOSE
C           THE PURPOSE OF RETURN IS TO RETURN TO THE USER'S AVAILABLE
C           SPACE LIST A PORTION OF A LINKED LIST.
C
C       USAGE
C           CALL RETURN(IAUASL,JCPOFP,JCPOSP,JCCLNO)
C
C       DATA FORMAT
C           N/A
C
C       DESCRIPTION OF PARAMETERS
C           IAUASL = A VECTOR DESCRIBED IN THE BSTLNK DOCUMENTATION
C           JCPOFP = THE POINTER TO THE FIRST WORD OF THE LEADING LIST
C                    ENTRY FOR RETURN, OR ZERO IF ALL ENTRIES FROM
C                    JCPOSP TO LIST BEGINNING ARE TO BE RETURNED
C           JCPOSP = THE POINTER TO THE FIRST WORD OF THE TRAILING LIST
C                    ENTRY FOR RETURN, OR ZERO IF ALL ENTRIES FROM
C                    JCPOFP TO LIST END ARE TO BE RETURNED
C           JCCLNO = A USER SPECIFIED CALL STATEMENT ID. NUMBER
C
C       REMARKS
C           THE NUMBER REPRESENTED BY JCTCIA MUST BE SUPPLIED IN THE
C           COMMON STATEMENT.
C
C       SUBROUTINE AND FUNCTION SUBPROGRAMS REQUIRED
C           FSTLNK
C           J5TST5
C           SYSENT
C           SYSEXT
C
C       ERROR CODES FOR THIS SUBPROGRAM
C           NONE
C
C       METHOD
C           RETURN TRACES THROUGH THE LIST FROM JCPOFP TO JCPOSP TO
C           INSURE CONTINUITY OF THE ASL FOLLOWING THE RETURN
C           OPERATION.
C
C                              **********
C
      SUBROUTINE RETURN(IAUASL,JCPOFP,JCPOSP,JCCLNO)
      COMMON/ALLOC/JCCORE(1)
      DIMENSION IAUASL(5)
      CALL SYSENT(1,1,3,11,JCCLNO)
      IF(JCPOFP.EQ.0)CALL J5TST5(JCPOSP,JCPOFP,IAUASL(5),2,1)
      IF(JCPOFP.NE.JCPOSP)CALL J5TST5(JCPOFP,JCPOSP,IAUASL(5),1,2)
C                    RETURN LIST ENTRIES TO ASL
     1 CALL FSTLNK(IAUASL,JCPOSP,IAUASL(2),1)
      IAUASL(2)=JCPOFP
      CALL SYSEXT
      RETURN
      END
```

```
C
C*******************************************************************
C                              **********
C                              * SETLIM *
C                              **********
C
C       PURPOSE
C           THE PURPOSE OF SETLIM IS TO PROVIDE THE CALLING PROGRAM
C           WITH POINTERS TO THE FIRST AND LAST WORD OF A LIST ENTRY,
C           THE SHIFT AND THE MASK REQUIRED FOR A DATA TRANSFER
C           OPERATION.
C
C       USAGE
C           CALL SETLIM(IAUASL,JCPOFP,LOCAL,JCINDX,JCMNLM,JCMXLM,
C           JCSHFT,JCMASK,JCCLNO)
C
C       DATA FORMAT
C           N/A
C
C       DESCRIPTION OF PARAMETERS
C           IAUASL = A VECTOR DESCRIBED IN THE BSTLNK DOCUMENTATION
C           JCPOFP = THE POINTER TO THE FIRST WORD OF THE INVOLVED
C                    LIST ENTRY
C           LOCAL  = A VECTOR INDICATING THE DATA TYPES FOR TRANSFER.
C                    SEE THE DOCUMENTATION FOR COPY FOR A COMPLETE
C                    DESCRIPTION OF LOCAL.
C           JCINDX = A PARAMETER INDICATING WHICH OF THE DATA TYPES
C                    LISTED IN THE LOCAL ARRAY IS CURRENTLY BEING
C                    PREPARED FOR THE TRANSFER OPERATION
C           JCMNLM = THE POINTER TO THE FIRST WORD INVOLVED IN A DATA
C                    TRANSFER OPERATION
C           JCMXLM = THE POINTER TO THE LAST WORD INVOLVED IN A DATA
C                    TRANSFER OPERATION
C           JCSHFT = THE SHIFT ASSOCIATED WITH THE DATA TYPE FOR
C                    TRANSFER
C           JCMASK = THE BIT MASK ASSOCIATED WITH THE DATA TYPE FOR
C                    TRANSFER
C           JCCLNO = A USER SPECIFIED CALL STATEMENT ID. NUMBER
C
C       REMARKS
C           THE NUMBER REPRESENTED BY JCTCIA MUST BE SUPPLIED IN THE
C           COMMON STATEMENT.
C
C       SUBROUTINE AND FUNCTION SUBPROGRAMS REQUIRED
C           SYSENT
C           SYSERR
C           SYSEXT
C
C       ERROR CODES FOR THIS SUBPROGRAM
C           CODE FATAL  ERROR(DATA PROVIDED)
C           1    F      NON-POSITIVE LOWER LIMIT (JCINDX)
C           2    F      LOWER LIMIT EXCEEDS DATA ITEM LIMITS (JCINDX)
C           3    F      LOWER LIMIT EXCEEDS LIST ENTRY SIZE (JCINDX)
C           4    F      LOWER LIMIT BEYOND BOUNDS OF JCCORE (JCPOFP)
C           5    F      UPPER LIMITS EXCEEDS DATA ITEM LIMITS (JCINDX)
C           6    F      UPPER LIMITS EXCEEDS LIST ENTRY SIZE (JCINDX)
C           7    F      UPPER LIMIT BEYOND BOUNDS OF JCCORE (JCPOFP)
C           8    F      LOWER LIMIT EXCEEDS UPPER LIMIT (JCINDX)
C
C       METHOD
C           ONLY ONE DATA TYPE AT A TIME IS CONSIDERED BY SETLIM.  THE
C           JCINDX PARAMETER INDICATES WHICH DATA TYPE FROM THE LOCAL
```

```
C            ARRAY IS BEING CONSIDERED.  A JCINDX OF 1 THUS MEANS THE
C            JCMNLM, JCMXLM, JCSHFT AND JCMASK PARAMETERS ARE TO BE
C            CALCULATED FOR THE FIRST DATA TYPE OF LOCAL.  IF THE DATA
C            IN THE ENTIRE LIST ENTRY IS TO BE TRANSFERRED AND LOCAL(1)
C            IS 0, SETLIM SHOULD NOT BE CALLED.
C
C                                  **********
C
      SUBROUTINE SETLIM(IAUASL,JCPOFP,LOCAL,JCINDX,JCMNLM,JCMXLM,
     1 JCSHFT,JCMASK,JCCLNO)
      COMMON/ALLOC/JCCORE(1)
      DIMENSION LOCAL(1),IAUASL(5)
      CALL SYSENT(1,1,3,12,JCCLNO)
C                  SECURING FROM LOCAL THE PARAMETERS DESCRIBING THE
C                  CURRENT DATA TYPE
      JCDUM3=3*JCINDX-1
      M1=IAUASL(5)
      JCDUM1=JCCORE(M1+20)+4*(LOCAL(JCDUM3)-1)+1
      JCDUM2=1
      M1=-2
C                  SETTING THE MINIMUM LIMIT
      JCDUM4=LOCAL(JCDUM3+1)
      IF(JCDUM4.LE.0)CALL SYSERR(-1,JCINDX)
    1 JCDUM4=JCCORE(JCDUM1)+JCDUM4-1
      IF((JCDUM4.GT.JCCORE(JCDUM1+1)).AND.(JCCORE(JCDUM1+1).GT.0))
     1 CALL SYSERR(M1,JCINDX)
      M1=M1-1
    2 IF(JCDUM4.GT.IAUASL(4))CALL SYSERR(M1,JCINDX)
      M1=M1-1
    3 JCDUM4=JCPOFP+JCDUM4-1
      IF(JCDUM4.GT.JCCORE(2))CALL SYSERR(M1,JCPOFP)
      GO TO (4,5),JCDUM2
    4 JCDUM2=2
      JCMNLM=JCDUM4
C                  SETTING THE MAXIMUM LIMIT
      M1=-5
      JCDUM4=LOCAL(JCDUM3+2)
      IF(JCDUM4.NE.0)GO TO 1
      JCDUM4=JCCORE(JCDUM1+1)
      M1=-6
      IF(JCDUM4.NE.0)GO TO 2
      JCDUM4=IAUASL(4)
      M1=-7
      GO TO 3
    5 JCMXLM=JCDUM4
      IF(JCMNLM.GT.JCMXLM)CALL SYSERR(-8,JCINDX)
C                  SETTING THE SHIFT AND BIT MASK
      JCSHFT=JCCORE(JCDUM1+2)
      JCMASK=JCCORE(JCDUM1+3)
      CALL SYSEXT
      RETURN
      END
```

```
C
C*****************************************************************
C                              **********
C                              * SETLNK *
C                              **********
C
C       PURPOSE
C           THE PURPOSE OF SETLNK IS TO LINK TWO INDICATED LIST ENTRIES
C           IN FORWARD-BACKWARD THREADED FASHION.
C
C       USAGE
C           CALL SETLNK(IAUASL,JCPOFP,JCPOSP,JCCLNO)
C
C       DATA FORMAT
C           N/A
C
C       DESCRIPTION OF PARAMETERS
C           IAUASL = A VECTOR DESCRIBED IN THE BSTLNK DOCUMENTATION
C           JCPOFP = THE POINTER TO THE FIRST WORD OF THE LEADING LIST
C                    ENTRY
C           JCPOSP = THE POINTER TO THE FIRST WORD OF THE TRAILING LIST
C                    ENTRY
C           JCCLNO = A USER ASSIGNED CALL STATEMENT ID. NUMBER
C
C       REMARKS
C           THE NUMBER REPRESENTED BY JCTCIA MUST BE SUPPLIED IN THE
C           COMMON STATEMENT.
C
C       SUBROUTINE AND FUNCTION SUBPROGRAMS REQUIRED
C           BSTLNK
C           FSTLNK
C           SYSENT
C           SYSEXT
C
C       ERROR CODES FOR THIS SUBPROGRAM
C           NONE
C
C       METHOD
C           SELF EXPLANATORY
C
C                              **********
C
      SUBROUTINE SETLNK(IAUASL,JCPOFP,JCPOSP,JCCLNO)
      COMMON/ALLOC/JCCORE(1)
      DIMENSION IAUASL(5)
      CALL SYSENT(1,1,3,13,JCCLNO)
      IF(JCPOFP.NE.0)CALL FSTLNK(IAUASL,JCPOFP,JCPOSP,1)
      IF(JCPOSP.NE.0)CALL BSTLNK(IAUASL,JCPOSP,JCPOFP,1)
      CALL SYSEXT
      RETURN
      END
```

```
C
C*******************************************************************
C                              **********
C                              * SYSOMP *
C                              **********
C
C       PURPOSE
C           SYSOMP OUTPUTS THE CONTENTS OF THE NAMED COMMON ALLOC.
C
C       USAGE
C           CALL SYSOMP
C
C       DATA FORMAT
C           N/A
C
C       DESCRIPTION OF PARAMETERS
C           NONE
C
C       REMARKS
C           THE NUMBER REPRESENTED BY JCTCIA MUST, FOR SOME MACHINES,
C           REPLACE THE 1 IN THE NAMED COMMON STATEMENT.
C
C       SUBROUTINE AND FUNCTION SUBPROGRAMS REQUIRED
C           NONE
C
C       ERROR CODES FOR THIS PROGRAM
C           NONE
C
C       METHOD
C           SELF-EXPLANATORY
C
C                              **********
C
      SUBROUTINE SYSOMP
      COMMON/ALLOC/JCCORE(1)
      JCCORE(13)=6
      M1=JCCORE(9)-1
      M2=JCCORE(11)
      WRITE(M2,101)
  101 FORMAT(11H1ALLOC DUMP)
      WRITE(M2,102)
  102 FORMAT(16H0PRE-ALLOCATABLE)
      M3=1
      M4=5
    1 IF(M4.GT.M1)M4=M1
      WRITE(M2,104)M3,(JCCORE(I),I=M3,M4)
  104 FORMAT(1H I9,1H)5I22)
      M3=M3+5
      M4=M4+5
      IF(M3.LE.M1)GO TO 1
      M3=M1+1
      M1=JCCORE(2)
      IF(M3.GT.M1)GO TO 2
      M4=M3+4
      WRITE(M2,105)
  105 FORMAT(12H0ALLOCATABLE)
      GO TO 1
    2 IF(JCCORE(19).GE.0)GO TO 3
      M1=IABS(JCCORE(19))
      M2=JCCORE(2)
      WRITE(M1,106)(JCCORE(I),I=1,M2)
  106 FORMAT(4I20)
```

```
3 RETURN
  END
```

```
C
C*****************************************************************
C                              **********
C                              * SYSENT *
C                              **********
C
C
C       PURPOSE
C           SYSENT RECORDS THE ENTRANCE INTO A SUBPROGRAM.
C
C       USAGE
C           CALL SYSENT(JCSYST,JCLEVL,JCPAKG,JCPROG,JCCLNO)
C
C       DATA FORMAT
C           N/A
C
C       DESCRIPTION OF PARAMETERS
C           JCSYST = SYSTEM ID NUMBER
C           JCLEVL = LEVEL ID NUMBER
C           JCPAKG = PACKAGE ID NUMBER
C           JCPROG = PROGRAM ID NUMBER
C           JCCLNO = CALL NUMBER
C
C       REMARKS
C           THE NUMBER REPRESENTED BY JCTCIA AND THE DIMENSION OF THE
C           JCTRAS VECTOR MUST, FOR SOME MACHINES, REPLACE THE 1'S IN
C           THEIR RESPECTIVE COMMON STATEMENTS.
C
C       SUBROUTINE AND FUNCTION SUBPROGRAMS REQUIRED
C           NONE
C
C       ERROR CODES FOR THIS PROGRAM
C           NONE
C
C       METHOD
C           JCCORE(12) HOLDS THE CURRENT LEVEL.  JCCORE(13) HOLDS THE
C           CURRENT LINE POSITION.  JCCORE(20) HOLDS THE TRACE CUT-OFF
C           LEVEL.  IF JCCORE(12) EXCEEDS JCCORE(20), NO TRACE MESSAGE
C           IS PROVIDED.
C
C                              **********
C
      SUBROUTINE SYSENT(JCSYST,JCLEVL,JCPAKG,JCPROG,JCCLNO)
      COMMON/ALLOC/JCCORE(1)
      COMMON JCTRAS(1)
C                   RECORD SUBPROGRAM IN JCTRAS
      M1=1+5*JCCORE(12)
      M2=JCCORE(13)
      DO 1 I=1,5
      JCTRAS(M1)=JCSYST*(1/I)+JCLEVL*(2/I)*(I/2)+JCPAKG*(3/I)*(I/3)
     1 +JCPROG*(4/I)*(I/4)+JCCLNO*(I/5)
    1 M1=M1+1
C                   INCREMENT LEVEL
      JCCORE(12)=JCCORE(12)+1
C                   TRACE MESSAGE
      IF(JCCORE(20).LT.JCCORE(12))GO TO 8
      MO=JCCORE(11)
      GO TO (3,4,5,6,7,2),M2
    2 JCCORE(13)=1
      WRITE(MO,101)JCCORE(12),JCSYST,JCLEVL,JCPAKG,JCPROG,JCCLNO
  101 FORMAT(2H NI2,1H(I2,1H,I2,1H,I2,1H,I2,1H,I2,2H) )
      GO TO 8
    3 JCCORE(13)=2
```

```
      WRITE(MO,102)JCCORE(12),JCSYST,JCLEVL,JCPAKG,JCPROG,JCCLNO
  102 FORMAT(1H+20X,1HN12,1H(12,1H,12,1H,12,1H,12,1H,12,2H) )
      GO TO 8
    4 JCCORE(13)=3
      WRITE(MO,103)JCCORE(12),JCSYST,JCLEVL,JCPAKG,JCPROG,JCCLNO
  103 FORMAT(1H+40X,1HN12,1H(12,1H,12,1H,12,1H,12,1H,12,2H) )
      GO TO 8
    5 JCCORE(13)=4
      WRITE(MO,104)JCCORE(12),JCSYST,JCLEVL,JCPAKG,JCPROG,JCCLNO
  104 FORMAT(1H+60X,1HN12,1H(12,1H,12,1H,12,1H,12,1H,12,2H) )
      GO TO 8
    6 JCCORE(13)=5
      WRITE(MO,105)JCCORE(12),JCSYST,JCLEVL,JCPAKG,JCPROG,JCCLNO
  105 FORMAT(1H+80X,1HN12,1H(12,1H,12,1H,12,1H,12,1H,12,1H,12,2H) )
      GO TO 8
    7 JCCORE(13)=6
      WRITE(MO,106)JCCORE(12),JCSYST,JCLEVL,JCPAKG,JCPROG,JCCLNO
  106 FORMAT(1H+100X,1HN12,1H(12,1H,12,1H,12,1H,12,1H,12,1H,12,2H) )
C                    EXIT
    8 RETURN
      END
```

```
C
C*****************************************************************
C                              **********
C                              * SYSERR *
C                              **********
C
C       PURPOSE
C           SYSERR PROVIDES AN ERROR MESSAGE.
C
C       USAGE
C           CALL SYSERR(JCCODE,JCDATA)
C
C       DATA FORMAT
C           N/A
C
C       DESCRIPTION OF PARAMETERS
C           JCCODE = THE ERROR CODE, TWO DIGIT MAXIMUM
C           JCDATA = ANY INTEGER DATA ITEM TO ASSIST IN DETERMINING THE
C                    NATURE OR CAUSE OF THE ERROR
C
C       REMARKS
C           SEE REMARKS FOR SYSENT.
C
C       SUBROUTINE AND FUNCTION SUBPROGRAMS REQUIRED
C           NONE
C
C       ERROR CODES FOR THIS PROGRAM
C           NONE
C
C       METHOD
C           SELF-EXPLANATORY
C
C                              **********
C
      SUBROUTINE SYSERR (JCCODE,JCDATA)
      COMMON/ALLOC/JCCORE(1)
      COMMON JCTRAS(1)
C                  SET DO LOOP LIMITS AND OTHER PARAMETERS
      M1=JCCORE(12)
      MO=JCCORE(11)
      M2=IABS(JCCODE)
C                  ERROR MESSAGE
      WRITE(MO,100)
  100 FORMAT(7H ERROR )
      MA=5
      DO 6 I=1,M1
      M3=1+5*(I-1)
      M4=M3+4
      GO TO (2,3,4,5,1),MA
    1 MA=1
      WRITE(MO,101)I,(JCTRAS(J),J=M3,M4)
  101 FORMAT(1H+6X,1HNI2,1H(I2,1H,I2,1H,I2,1H,I2,1H,I2,2H) )
      GO TO 6
    2 MA=2
      WRITE(MO,102)I,(JCTRAS(J),J=M3,M4)
  102 FORMAT(1H+26X,1HNI2,1H(I2,1H,I2,1H,I2,1H,I2,1H,I2,2H) )
      GO TO 6
    3 MA=3
      WRITE(MO,103)I,(JCTRAS(J),J=M3,M4)
  103 FORMAT(1H+46X,1HNI2,1H(I2,1H,I2,1H,I2,1H,I2,1H,I2,2H) )
      GO TO 6
    4 MA=4
```

```
      JCDUM2=J5TST6(JCTOLD,1)
C                     IS LIST STRUCTURE FORWARD ONLY OR FORWARD-BACKWARD
      IF(JCDUM2.EQ.2)GO TO 1
C                     IS JCPOFP A PTC
      CALL J5TST2(1,JCTOLD,JCPOFP,1)
    1 JCDUM2=LOCAL(1)
C                     STORE DATA IN NONE, ALL OR PART
      JCDUM3=J5TST4(JCDUM2,1)
      GO TO (104,2,5),JCDUM3
C                     STORE IN ALL, A WORD AT A TIME
    2 DO 3 I=1,JCSIZE
      JCDUM3=JCPOFP+I-1
    3 JCCORE(JCDUM3)=LION(I)
      GO TO 104
C                     STORE IN PART, ONE DATA TYPE AT A TIME
    5 DO 6 I=1,JCDUM2
C                     SETTING THE LIMITS FOR THE DATA STORAGE DO LOOP FOR
C                     THE CURRENT DATA TYPE
      JCMASK=-42
      CALL SETLIM(IAUASL,JCPOFP,LOCAL,I,JCMNLM,JCMXLM,JCSHFT,JCMASK,1)
      JCDUM3=JCSHFT*JCMASK
      DO 6 J=JCMNLM,JCMXLM
C                     THE DATA STORAGE OPERATION, ONE DATA ITEM AT A TIME
C                     FOR EACH DATA TYPE
      JCDUM1=JCDUM1+1
      JCDUM4=LION(JCDUM1)*JCSHFT
      JCDUM5=JCCORE(J)
      IF(JCDUM3.NE.0)JCDUM5=LAND(JCDUM3,JCDUM5)
      JCDUM5=LEDR(JCDUM5,JCCORE(J))
    6 JCCORE(J)=LOR(JCDUM4,JCDUM5)
  104 CALL SYSEXT
      RETURN
      END
```

```
C
C*****************************************************************
C                              **********
C                              * TOCELL *
C                              **********
C
C       PURPOSE
C           TOCELL STORES IN A SPECIFIED LIST ENTRY THE DATA PROVIDED
C           BY A VECTOR LION.
C
C       USAGE
C           CALL TOCELL(IAUASL,JCPOFP,LION,LOCAL,JCCLNO)
C
C       DATA FORMAT
C           N/A
C
C       DESCRIPTION OF PARAMETERS
C           LION   = A VECTOR PROVIDING THE DATA TO BE STORED
C           JCPOFP = THE POINTER TO THE FIRST WORD OF THE LIST ENTRY TO
C                    RECEIVE THE DATA
C           JCTOLD = A PARAMETER INDICATING THE TYPE OF LIST EMPLOYED
C           LOCAL  = A VECTOR INDICATING THE LOCATIONS WITHIN THE LIST
C                    ENTRY TO RECEIVE DATA.  SEE THE DOCUMENTATION FOR
C                    COPY FOR A COMPLETE DESCRIPTION OF LOCAL.
C           JCSIZE = THE NUMBER OF WORDS PER LIST ENTRY
C           JCCLNO = A USER SPECIFIED CALL STATEMENT ID. NUMBER
C
C       REMARKS
C           THE NUMBER REPRESENTED BY JCTCIA MUST BE SUPPLIED IN THE
C           COMMON STATEMENT.
C
C       SUBROUTINE AND FUNCTION SUBPROGRAMS REQUIRED
C           J5TST1
C           J5TST2
C           J5TST4
C           J5TST6
C           LAND
C           LEOR
C           LOR
C           SETLIM
C           SYSENT
C           SYSEXT
C
C       ERROR CODES FOR THIS SUBPROGRAM
C           NONE
C
C       METHOD
C           THE VECTOR LOCAL DETERMINES THE LOCATIONS WITHIN JCPOFP FOR
C           THE DATA ITEMS TRANSFERRED FROM LION.  IF JCPOFP IS A PTC,
C           IT BECOMES THE LAST LIST ENTRY OF THE LIST.
C
C                              **********
C
      SUBROUTINE TOCELL(IAUASL,JCPOFP,LION,LOCAL,JCCLNO)
      COMMON/ALLOC/JCCORE(1)
      DIMENSION LION(1),LOCAL(1),IAUASL(5)
      CALL SYSENT(1,1,3,14,JCCLNO)
      JCTOLD=IAUASL(5)
      JCSIZE=IAUASL(4)
      JCDUM1=0
C                 IS JCPOFP LEGAL
      CALL J5TST1(JCPOFP,1)
```

```
      WRITE(MO,104)I,(JCTRAS(J),J=M3,M4)
  104 FORMAT(1H+66X,1HNI2,1H(I2,1H,I2,1H,I2,1H,I2,1H,I2,2H) )
      GO TO 6
    5 MA=5
      WRITE(MO,105)I,(JCTRAS(J),J=M3,M4)
  105 FORMAT(1H+86X,1HNI2,1H(I2,1H,I2,1H,I2,1H,I2,1H,I2,2H) )
      WRITE(MO,106)
  106 FORMAT(1H )
    6 JCCORE(13)=6
      GO TO (8,9,10,11,7),MA
    7 WRITE(MO,201)M2,JCDATA
  201 FORMAT(1H+6X,1HEI2,2H OI15)
      GO TO 12
    8 WRITE(MO,202)M2,JCDATA
  202 FORMAT(1H+26X,1HEI2,2H OI15)
      GO TO 12
    9 WRITE(MO,203)M2,JCDATA
  203 FORMAT(1H+46X,1HEI2,2H OI15)
      GO TO 12
   10 WRITE(MO,204)M2,JCDATA
  204 FORMAT(1H+66X,1HEI2,2H OI15)
      GO TO 12
   11 WRITE(MO,205)M2,JCDATA
  205 FORMAT(1H+86X,1HEI2,2H OI15)
C                     DUMP
   12 IF(JCCORE(19).NE.0)CALL SYSDMP
C                     IS OVER-RIDE NON-FATAL IN EFFECT
      IF(JCCORE(14).GT.0)GO TO 13
C                     IS OVER-RIDE FATAL IN EFFECT
      IF(JCCORE(14).LT.0)STOP
C                     IS ERROR CODE FATAL
      IF(JCCODE.LT.0)STOP
C                     EXIT
   13 RETURN
      END
```

```
C
C*********************************************************************
C                              **********
C                              * SYSEXT *
C                              **********
C
C       PURPOSE
C           SYSEXT RECORDS THE EXIT FROM A SUBPROGRAM.
C
C       USAGE
C           CALL SYSEXT
C
C       DATA FORMAT
C           N/A
C
C       DESCRIPTION OF PARAMETERS
C           N/A
C
C       REMARKS
C           SEE REMARKS FOR SYSENT.
C
C       SUBROUTINE AND FUNCTION SUBPROGRAMS REQUIRED
C           NONE
C
C       ERROR CODES FOR THIS PROGRAM
C           NONE
C
C       METHOD
C           SYSEXT TURNS THE TRACE ON IF A NEGATIVE JCTRAS WORD IS
C           ENCOUNTERED.  A TRACE MESSAGE IS DISPLAYED IF JCCORE(20)
C           IS GREATER THAN JCCORE(12).
C
C                              **********
C
      SUBROUTINE SYSEXT
      COMMON/ALLOC/JCCORE(1)
      COMMON JCTRAS(1)
C                 ACCESS JCTRAS VECTOR
      M1=1+5*(JCCORE(12)-1)
      M2=M1+4
C                 TRACE MESSAGE
      IF(JCCORE(20).LT.JCCORE(12))GO TO 7
      MO=JCCORE(11)
      MA=JCCORE(13)
      GO TO (2,3,4,5,6,1),MA
    1 JCCORE(13)=1
      WRITE(MO,101)JCCORE(12),(JCTRAS(I),I=M1,M2)
  101 FORMAT(2H XI2,1H(I2,1H,I2,1H,I2,1H,I2,1H,I2,2H) )
      GO TO 7
    2 JCCORE(13)=2
      WRITE(MO,102)JCCORE(12),(JCTRAS(I),I=M1,M2)
  102 FORMAT(1H+20X,1HXI2,1H(I2,1H,I2,1H,I2,1H,I2,1H,I2,2H) )
      GO TO 7
    3 JCCORE(13)=3
      WRITE(MO,103)JCCORE(12),(JCTRAS(I),I=M1,M2)
  103 FORMAT(1H+40X,1HXI2,1H(I2,1H,I2,1H,I2,1H,I2,1H,I2,2H) )
      GO TO 7
    4 JCCORE(13)=4
      WRITE(MO,104)JCCORE(12),(JCTRAS(I),I=M1,M2)
  104 FORMAT(1H+60X,1HXI2,1H(I2,1H,I2,1H,I2,1H,I2,1H,I2,2H) )
      GO TO 7
    5 JCCORE(13)=5
```

```
      WRITE(MO,105)JCCORE(12),(JCTRAS(I),I=M1,M2)
  105 FORMAT(1H+80X,1HX,I2,1H(,I2,1H,,I2,1H,,I2,1H,,I2,1H,,I2,2H) )
      GO TO 7
    6 JCCORE(13)=6
      WRITE(MO,106)JCCORE(12),(JCTRAS(I),I=M1,M2)
  106 FORMAT(1H+100X,1HX,I2,1H(,I2,1H,,I2,1H,,I2,1H,,I2,1H,,I2,2H) )
C                  DECREMENT LEVEL
    7 JCCORE(12)=JCCORE(12)-1
C                  EXIT
      RETURN
      END
```

APPENDIX B

In its current version, there is no incentive for employing indexed variables and equations with GENDER. However, this will no always be the case. Memory and CPU time savings offer a strong incentive for the solution of such problems as pattern recognition. Since research is currently pursuing the index related problems, it was felt that ample justification existed for the inclusion of software within GENDER to provide an indexing capability.

The indices permitted in both SECEDE and the GENDER list are each composed of seven components. The components are data types 9 through 15 for both list types 1 (SECEDE) and 2 (GENDER list). The LEND's must be devised to permit the storage of all seven components in a single word.

The incentive for the complexity of the seven part index is a mechanism we shall call mapping. Mapping will in turn facilitate the implementation of the GENDER equivalent of FORTRAN DO loops. Let us consider an index i and an index j. Index i maps to index j if i is a function j - i.e. i = F(j). Thus, an output variable index may be a mapping of an equation index. If the equation index is permitted a range of values, the variation of the output variable index (as a consequence of mapping) will be automatic.

Index ranges in GENDER are permitted by the inclusion of a minimum index and a maximum index. Index ranges are allowed for equations, ER's, constraints and groups. This feature is expected to be of considerable

importance in reducing the analysis effort and storage requirements for multistage processing units.

For the purpose of considering the combination of the seven components into a single index value, let us identify the data fields 9 through 15 by the letters a through g respectively. Components a and b identify the index to be employed in the mapping calculation. When a is 1, b is the depth of the group supplying the index. Depth specifies the particular group in a cascade of dimensioned groups. When a is 2, b specifies the particular equation, ER or constraint index to be used. Indices are numbered in the order of appearance in the group body. When a is 3, b refers to an index of the same entity for which the index calculation is to be performed. That is, an equation index may be a function of another index of the same equation.

Using a and b, a mapping index j is selected. This j is related to i by the relationship.

```
i = j*(-1)**d*(e**(-1)**c) + (-1)**f*g
```

in which the FORTRAN operator symbols are employed. When use of a mapping index is not desired, the components a through f will all be valued at zero. To illustrate the use of this relationship, let us consider a numberical example. Suppose that we have a dimensioned output variable, one index, i, of which is one larger than the second function index, j. The seven components for the output variable index would be (2, 2, 0, 0, 1, 0, 1), to yield

```
i = j + 1
```

Each group on a GENDER list is permitted one index. Two words are reserved for the group index and are employed to store the minimum and maximum index values in the seven part format. This feature imparts the character of a DO statement to the dimensioned group. Since groups

are permitted on the group bodies of other groups, a cascade of DO loops can be simulated on a GENDER list. The group indices are numbered for identification in the order of encounter in descending from the outermost group to the innermost group. This number is termed the depth of the group. Note that undimensioned groups encountered in descending from group to inner group do not contribute to the depth computation. Also note that index calculations stipulating a mapping onto a group index will employ the current value of the group index. Current values to group indices are stored in the vector provided by CUDGEL when creating the crude GENDER list.

We shall consider a particular group at depth 3, with a current index value k. Let us continue our numberical example by assuming that j is four less than one half the index k for the group at depth 3. In this case, the seven components would be (1, 3, 1, 0, 2, 1, 4), giving the relation

$$j = k/2 - 4$$

Finally, assuming that k does not map onto another group index, the seven components representing k would be (0, 0, 0, 0, 0, 0, k). If k is valued at 8, then the mappings stipulate values of 4 and 5 for j and i respectively.

APPENDIX C

1	0	0	0	1
1	122	0	0	
1	110	0	0	
1	111	0	0	
1	103	0	0	
1	107	0	0	
1	109	0	0	
1	112	0	0	
1	108	0	0	
1	101	0	0	
1	104	0	0	
1	123	0	0	
1	118	0	0	
1	132	0	1	
1	131	0	1	
1	130	0	1	
1	129	0	1	
1	128	0	1	
1	127	0	1	
1	126	0	1	
1	125	0	1	
1	1	0		
1				
1	3	104	0	1
1	124	0	0	
1	3	103	0	1
1	121	0	0	
1	3	101	0	1
1	115	0	0	
1	3	102	0	1
1	119	0	0	
1	3	112	0	1
1	120	0	0	
1	3	111	0	1
1	116	0	0	
1	0	0	0	1
1	113	0	0	
1	105	0	0	
1	3	109	0	1
1	106	0	0	
1	3	108	0	1
1	102	0	0	

1	3	105	0	1
1	114	0	0	
1	1	-1		
1				
1	1	1		
2				
1	1	2		
3				
1	1	3		
4				
1	1	4		
5				
1	1	5		
6				
1	1	6		
7				
1	1	7		
8				
1	1	8		
9				
1	1	9		
1	3	110	0	1
1	117	0	0	
1	3	106	0	1
1	105	0	0	
1	3	107	0	1
1	113	0	0	
1	1	-7		
1				
1	1	1		
2				
1	1	2		
3				

APPENDIX D

1	0	0	0	1
1	1	0	0	
1	132	0	0	
1	301	0	0	
1	302	0	0	
1	303	0	0	
1	304	0	0	
1	2	0	1	
1	3	0	1	
1	4	0	1	
1	5	0	1	
1	6	0	1	
1	7	0	1	
1	8	0	1	
1	9	0	1	
1	10	0	1	
1	11	0	1	
1	12	0	1	
1	13	0	1	
1	1	0		
	1			
1	3	108	0	1
1	103	0	0	
1	3	208	0	1
1	203	0	0	
1	3	308	0	1
1	303	0	0	
1	3	408	0	1
1	403	0	0	
1	0	0	0	2
1	123	0		
1	114	0		
1	113	0		
1	116	0		
1	223	0		
1	19	0		
1	18	0		
1	21	0		
1	323	0		
1	27	0		
1	26	0		
1	29	0		
1	423	0		
1	35	0		
1	34	0		
1	37	0		
1	43	0		
1	42	0		
1	45	0		
1	3	505	0	1
1	511	0	0	

1	1	-1		
	1			
1	1	1		
	2			
1	1	2		
	3			
1	1	3		
	4			
1	1	4		
	5			
1	1	5		
	6			
1	3	103	0	1
1	121	0	0	
1	3	104	0	1
1	122	0	0	
1	3	109	0	1
1	115	0	0	
1	3	105	0	1
1	15	0	0	
1	3	101	0	1
1	114	0	0	
1	3	102	0	1
1	16	0	0	
1	3	106	0	1
1	14	0	0	
1	3	107	0	1
1	113	0	0	
1	3	110	0	1
1	17	0	0	
1	3	111	0	1
1	116	0	0	
1	3	112	0	1
1	123	0	0	
1	3	203	0	1
1	221	0	0	
1	3	204	0	1
1	222	0	0	
1	3	209	0	1
1	20	0	0	
1	3	205	0	1
1	23	0	0	
1	3	201	0	1
1	19	0	0	
1	3	202	0	1

1	24	0	0	
1	3	206	0	1
1	22	0	0	
1	3	207	0	1
1	18	0	0	
1	3	210	0	1
1	25	0	0	
1	3	111	0	1
1	21	0	0	
1	3	112	0	1
1	223	0	0	
1	3	303	0	1
1	321	0	0	
1	3	304	0	1
1	322	0	0	
1	3	309	0	1
1	28	0	0	
1	3	305	0	1
1	302	0	0	
1	3	301	0	1
1	27	0	0	
1	3	302	0	1
1	32	0	0	
1	3	306	0	1
1	301	0	0	
1	3	307	0	1
1	26	0	0	
1	3	310	0	1
1	304	0	0	
1	3	311	0	1
1	29	0	0	
1	3	312	0	1
1	323	0	0	
1	3	403	0	1
1	421	0	0	
1	3	404	0	1
1	422	0	0	
1	3	409	0	1
1	36	0	0	
1	3	405	0	1
1	39	0	0	

1	3	401	0	1
1	35	0	0	
1	3	402	0	1
1	40	0	0	
1	3	406	0	1
1	38	0	0	
1	3	407	0	1
1	34	0	0	
1	3	410	0	1
1	41	0	0	
1	3	411	0	1
1	37	0	0	
1	3	412	0	1
1	423	0	0	
1	3	501	0	1
1	43	0	0	
1	3	502	0	1
1	44	0	0	
1	3	504	0	1
1	508	0	0	
1	3	503	0	1
1	42	0	0	
1	3	506	0	1
1	45	0	0	

1	1	-5
1	1	1
1	1	2
1	1	3
1	1	4
1	1	5
1	1	6
1	1	7
1	1	8
1	1	9
1	1	10
1	1	11
1	1	12

	13	
1	1	13
	14	
1	1	14
	15	
1	1	15
	16	
1	1	16
	17	
1	1	17
	18	
1	1	18
	19	
1	1	19
	20	
1	1	20
	21	
1	1	21
	22	
1	1	22
	23	
1	1	23
	24	
1	1	24
	25	
1	1	25
	26	
1	1	26
	27	
1	1	27
	28	
1	1	28
1	29	
1	1	29
	30	
1	1	30
	31	
1	1	31
1	32	
1	1	32
	33	
1	1	33
	34	
1	1	34
	35	
1	1	35
	36	
1	1	36
	37	
1	1	37
	38	
1	1	38
	39	
1	1	39
	40	
1	1	40
	41	
1	1	41
	42	
1	1	42
	43	
1	1	43
	44	
1	1	44
	45	
1	1	45
	46	
1	1	46
	47	
1	1	47
	48	
1	1	48
	49	

BIBLIOGRAPHY

Barkley, R. W. and R. L. Motard, "Decomposition of Nets," NATO Institute on Decomposition, Cambridge, England (July, 1972).

Christensen, J. H., "The Structuring of Process Optimization," AIChE J., 16(2), pp. 177-184 (1970).

Cooper, D. C. and H. Whitfield, "ALP: An Autocode List-Processing Language," The Computer Journal, 5(28), (1962/3).

DeBrosse, C. J., "Feasible-point and Optimization Algorithms for Structured Design Systems in Chemical Engineering," Ph.D Dissertation, pp. 67-84, University of Florida (1971).

Edie, F. C., "Convergence and Tearing Algorithms for Solving Structured Systems of Design Equations in Chemical Engineering," Ph.D. Dissertation, pp. 20-27, University of Florida (1970).

Edie, F. C. and A. W. Westerberg, "Computer Aided Design, Part 3," Chem. Engng. J., 2, pp. 114-124 (1971).

Gupta, P.K., "Application of Analysis and Optimization Techniques for Structured Systems to the Design of a Double Effect Evaporator System," Master's Thesis, University of Florida (1972).

Kevorkian, A. K. and J. Snoek, "Decomposition in Large Scale Systems, Parts I and II," NATO Institute on Decomposition, Cambridge England (July, 1972).

Knuth, D. E., "The Art of Computer Programming, Vol. 1," Addison-Wesley Pub. Co., Reading, Mass. (1968).

Lee, W. and Y. Ozawa, "Complex Chemical Engineering Systems," Chem. Engng. Educ., 5(2), pp. 144-145 (1971).

Mah, R. S. H. and M. Rafal, "Automatic Program Generation in Chemical Engineering Computation," Trans. Instn. Chem. Engrs., 49, 101-108 (1971).

Sargent, R. W. H. and A. W. Westerberg, "'SPEED-UP' in Chemical Engineering Design," Trans. Instn. Chem. Engrs., 42(5), pp. T190-197 (1964).

Soylemez, S., "Computer Generated Fortran Programs to Prepare Material and Energy Balances for Process Units," Ph.D. Dissertation, Univ. of Pennsylvania (1971).

Soylemez, S. and W. D. Seider, "A New Technique for Precedence-Ordering Chemical Process Equation Sets," 71st National Meeting of the AIChE, Dallas, Texas (1972).

BIOGRAPHICAL SKETCH

James Richard Cunningham was born on January 14, 1942, in Houston, Texas, to Carl D. and Annette E. Cunningham. He was graduated from Destrehan High School at Destrehan, Louisiana, in June, 1959. In June, 1963, he was awarded the Bachelor of Science degree in chemical engineering from Louisiana State University.

Following graduation from Louisiana State University, he accepted a position at Texaco, Inc., in the Domestic Producing Department (Gas). His duties at Texaco involved primarily the operations aspects of the Paradis, Louisiana, natural gas recycling facility. This employment was terminated in January, 1964 for induction into the United States Air Force. He served four years as an egress system specialist, attaining the final grade of SSGT prior to honorable discharge. After discharge, he was employed at the Shell Chemical Company plant in Norco, Louisiana, as a technologist. In addition to providing technical support to plant operations personnel, he developed a computer simulation program modeling a reaction-distillation unit in the steady state.

In September, 1968, he elected to pursue graduate study at the University of Florida. He received the degree of Master of Engineering in chemical engineering in June, 1970.

James Cunningham is married to the former Eleanor Gayle Kincaid of Nashville, Tennessee. They are the parents of one son, Sean Carl, in whom they are well pleased.

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

Arthur W. Westerberg, Chairman
Associate Professor of Chemical Engineering

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

Frank P. May
Professor of Chemical Engineering

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

Ralph G. Selfridge
Professor of Mathematics and Director of the Computing Center

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

Frank D. Vickers
Associate Professor of Computer and Information Sciences

This dissertation was submitted to the Dean of the College of Engineering and to the Graduate Council, and was accepted as partial fulfillment of the requirements for the degree of Doctor of Philosophy.

December, 1972

Robert E Uhrig
Dean, College of Engineering

Dean, Graduate School

UNIVERSITY OF FLORIDA
3 1262 08666 411 6