Bellcomm

955 L'Enfant Plaza North, S.W.
Washington, D. C. 20024

B71 12025

date: December 30, 1971

to: Distribution

from: J. L. Marshall, Jr.

subject: WAP - A General Purpose Weight and Performance Analysis Program
Case 310

(NASA-CR-126220) WAP: A GENERAL PURPOSE WEIGHT AND PERFORMANCE ANALYSIS PROGRAM (Bellcomm, Inc.) 103 p

N79-72428

Unclas
00/18 12199

ABSTRACT

Solution of the rocket equation is required for a large class of problems in activities such as space vehicle sizing, mission planning, and weight and performance computation. One characteristic of these problems is that although they are generally not complex, their solution would be quite time-consuming without the aid of a digital computer. Thus, numerous relatively simple, special purpose computer programs have been written to solve them. However, it often happens that when a quick, reliable solution to a problem is required, the available programs are not quite in the proper form for the particular problem under consideration. The Fortran V program described herein was written to alleviate this situation. A strong attempt has been made to make the program usable by those who have little or no experience in using the computer.

Data in this program are organized in a matrix format in which the total vehicle weight and its components (stage weights, propellants, etc.) are listed in columns. Two columns are also provided for the delta velocity and specific impulse associated with each burn. The odd rows of the matrix contain the weights at a given event during the mission, and the even rows contain the weight changes between two events. Thus, the matrix represents a time history of the weights throughout a pre-specified mission. The user specifies the elements of this matrix as known constants or as unknown variables to be calculated. Any combination of knowns and unknowns can be specified and the program will either find the unknowns, or indicate that no solution is possible for the given matrix.

Numerous options are provided to make the program flexible and easy to use. These include facilities to (1) set up the general problem to be solved, using a precompiler (i.e., specify names of weight components and events in the matrix), (2) change data in the matrix (e.g., change an element from known to unknown or change numerical values of elements), (3) modify the format and type of data that are printed, (4) calculate sensitivities, and (5) solve the problem repeatedly with a specified parameter incremented each time. The program is fast enough to provide immediate response when run from a remote terminal.

Bellcomm

955 L'Enfant Plaza North, S.W.
Washington, D. C. 20024

B71 12025

date: December 30, 1971

to: Distribution

from: J. L. Marshall, Jr.

subject: WAP - A General Purpose Weight and Performance Analysis Program
Case 310

MEMORANDUM FOR FILE

Introduction

A large number of problems in activities such as space vehicle sizing, mission planning, and weight and performance computation require the solution of the rocket equation,

$$\Delta V = ISP*G*LN(MI/MF) \ .$$

That is, the change in velocity (ΔV) imparted to a space vehicle by a propulsion system with an efficiency that is characterized by a given specific impulse (ISP) can be expressed as a function of the natural logarithm of the ratio of the mass of the vehicle immediately before the burn (MI) to the mass immediately after the burn (MF). One characteristic of these problems is that although they are generally not complex, their solution would be quite time-consuming without the aid of a digital computer. As a result, numerous relatively simple, special purpose computer programs have been written to solve them. From time to time a quick, reliable solution to such a problem is required. Unfortunately, the problem can be stated in so many different ways that the special purpose program must often be rewritten to suit the new problem. The general purpose Fortran V program described herein was developed to minimize the likelihood that it will be necessary to rewrite a program. Particular attention was paid to keeping the program flexible yet simple to use. As a result, it is expected that those with little or no computer experience will be able to use the program. The speed of the program is such that response from a remote terminal is usually immediate.

Subsequent sections of this memorandum discuss the general type of problem that can be solved with the program, the approach that was taken in developing the program, and the features of the program that contribute to its flexibility. The appendices provide detailed information on the operation of the program. A summary of their contents is as follows:

Appendix A	Use of the precompiler to generate a main program, including an example showing the cards required to set up a particular problem.
Appendix B	Use of the main program generated by the precompiler. Contains illustrations of most of the commands and error messages.
Appendix C	A brief summary and listing of each subprogram.
Appendix D	A description of the commands that may be used during execution of the main program.
Appendix E	A list, with no description, of the acceptable forms for data input cards and commands, for both precompilation and execution of the main program. Includes a list of the abbreviated forms that may be used for the commands.

Although this organization leads to some repetition, it offers the advantage of providing both a step-by-step description of the operation of the program and convenient lists of commands for quick reference.

Statement of the Problem

In the type of problem under consideration, some component of the weight of a space vehicle (or a delta velocity or specific impulse) at some event, or point in time, during the mission is to be calculated. This can be expanded into a more general problem of determining a weight history consisting of the total space vehicle weight and the components of the total, each specified at a number of points during the mission. For convenience, such data can be organized in a matrix in which columns are provided for the delta velocity, the specific impulse, the total weight, and the weight components. Odd numbered rows can then represent the status at a particular event, while the even numbered rows can indicate the changes in weight between two events. Such a matrix, based on the Apollo lunar module, is shown in Table 1. This matrix contains a time history of the total LM weight and four of its components (ascent and descent stage weights, and APS and DPS propellant weights).

To solve this general problem, the user should be able to assign desired values to each of the elements in the matrix, with the option to designate any of the elements as unknowns to be calculated by the program. Of course, this should be accomplished without the necessity of changing or rewriting the program.

TABLE 1

LUNAR MODULE WEIGHT HISTORY

	ASCENT STAGE	APS PROPELLANT	DESCENT STAGE	DPS PROPELLANT	TOTAL LM	ΔV	ISP
BEGIN DESCENT	5755.1	5207.6	6118.2	19508.5	36589.4	7013.9	302.3
WEIGHT CHANGE	-97.0	0.0	-30.2	-18759.2	-18886.4	----	----
TOUCHDOWN	5658.1	5207.6	6088.0	749.3	17703.0	----	----
WEIGHT CHANGE	-19.8	0.0	-6088.0	-749.3	-6857.1	----	----
BEGIN ASCENT	5638.3	5207.6	0.0	0.0	10845.9	6079.8	309.4
WEIGHT CHANGE	-99.7	-4957.2	0.0	0.0	-5056.9	----	----
DOCK	5538.6	250.4	0.0	0.0	5789.0	----	----

Approach

The program described herein was designed to solve the general problem outlined above. The first time a particular problem is to be solved, the user must specify to the precompiler the names of the columns and rows of the matrix, as well as a set of initial conditions for the elements of the matrix. Each element is either given a numerical value or designated as an unknown. The precompiler uses these data to generate a main program which can subsequently be used repeatedly to solve problems using this same basic matrix with different values for the elements. The precompiler is needed again only if the number of rows and columns or their names are to be changed.

During execution of the main program, the matrix defined above is examined and an attempt is made to find values for the elements that have been designated as unknown. To understand the procedure used, consider the diagram in Figure 1, which represents what occurs between two events. At Event A, the total weight (or mass) of the vehicle is MI. If a rocket engine, whose efficiency is characterized by a specific impulse (ISP), is fired and burns a propellant mass (MP), a delta velocity (ΔV) is imparted to the vehicle. If other consumable items (MC) such as oxygen, water, ablative material, etc. are expended before Event B, then the final mass (MF), or the mass at Event B, is given by

$$MF = MI - MP - MC \ . \tag{1}$$

The relationship between the ΔV, the ISP, and the masses is given by the rocket equation,

$$\Delta V = ISP*G*LN(MI/(MI-MP))^{\dagger} \tag{2}$$

where G is the acceleration due to gravity, and
LN(X) is the natural logarithm of X.

---

† This form of the equation applies when the consumable items are assumed to be used discretely <u>after</u> the burn is completed. The program also contains a modified form based on the assumption that the consumable items are used linearly during the burn.

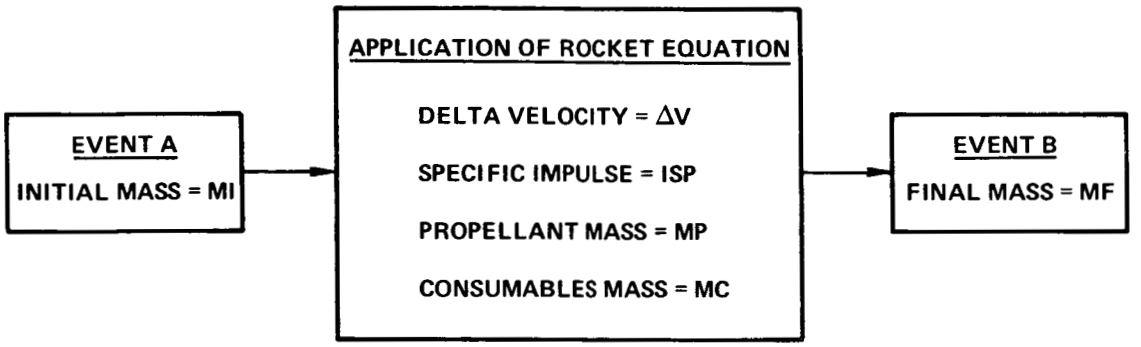


FIGURE 1

Now consider the other weights associated with these two events as outlined below in an abbreviated matrix.

	COL_1	COL_2	COL_3	COL_4	COL_n
EVENT A	M_{A1}	M_{A2}	M_{A3}	M_{A4}	M_{An}
	Δ_1	Δ_2	Δ_3	Δ_4	Δ_n
EVENT B	M_{B1}	M_{B2}	M_{B3}	M_{B4}	M_{Bn}

The n'th column (COL_n) contains the total weight of the vehicle, and columns 1 through n-1 contain the component weights. (The columns for ΔV and ISP are omitted here for clarity.) The following equations must be satisfied.

$$MI = M_{An} = \sum_{i=1}^{n-1} M_{Ai} \tag{3}$$

$$MC + MP = -\Delta_n = -\sum_{j=1}^{n-1} \Delta_j \tag{4}$$

$$MP = -\Delta_p \quad (p \neq n) \tag{5}$$

where p identifies the column (weight component) that is to be used for propellant for this burn.

$$M_{Bk} = M_{Ak} + \Delta_k \text{ for } k = 1,2 \ldots . n \tag{6}$$

$$MF = M_{Bn} . \tag{7}$$

Equations (1) through (7) must be satisfied for each set consisting of an event, a group of weight changes, and a subsequent event. The procedure used in the program is as follows. First, the first pair of events in the matrix is examined to determine if enough data are specified to allow solution of the rocket equation. If so, the appropriate unknowns are calculated. Then the elements of Event A are examined to see if equation (3) can be satisfied. Again, if possible, additional blanks are filled in (i.e., additional unknowns are calculated - unknowns are represented by blanks in the program). Next an attempt is made to satisfy equation (4), and finally, equations (6), each time filling in as many blanks as possible. The same sequence is then followed with the next and all other pairs of events, working from the top to the bottom of the matrix. When the matrix has been examined in this manner, the number of blanks left is determined, and if there are fewer blanks now than when the analysis began, all of these steps are repeated. This entire process is repeated until there are no blanks, or until the number of blanks can no longer be reduced by this procedure. In the former case, a complete solution has been obtained. In the latter case it may be that insufficient data were specified; however, it is also possible that the problem is one of a class that cannot be completely solved by the above approach. Therefore, before giving up, a test is made to determine whether sufficient data were specified. If insufficient data were provided, an error message is printed and the program is ready for new data or commands. If the data provided were sufficient, the program attempts to complete the solution by an iteration procedure. One of the known elements is chosen as a comparison or target variable, and one of the unknowns is assigned various values in an attempt to determine a value that will satisfy the equations for the entire matrix and match the target variable. If necessary, this is tried for many combinations of iterating and target variables, until a solution is found and printed. The program is then ready for new data or commands.

Flexibility

A primary goal when development of this program was begun was to provide a capability for solving a wide variety of problems without the necessity of program changes. On a gross scale, this was accomplished by means of the precompiler

mentioned in the previous section. The precompiler permits the user to specify the overall organization of the program, i.e., the number of weight components and events to be considered, the names of the components and events, and initial values for all variables. On a smaller scale, the user should be able to easily specify any combination of knowns and unknowns in the matrix. The method for accomplishing this by inputting data to the program is discussed in detail in the appendices, but some of the most important characteristics are described here.

All 80 columns of a card (or a line typed on a remote terminal) are read by the program. However, except in a few special cases such as in headings or titles, all blanks are ignored. Thus, characters may be spaced on a card in any convenient manner. Numbers may be specified as integers or floating point constants.

An element in the matrix can be changed by first submitting a card containing only the name of the event in which it appears. This tells the program that subsequent cards contain data pertaining to that event. The next card is of the form

NAME=NUMBER

where NAME is the name of the column in which the element appears, and NUMBER is the new desired value for the element. If it is desired to make the element an unknown, the card contains only the column name, without "=NUMBER." If an element in the row following this particular event (i.e., the row of weight changes) is to be changed, the same procedure is used except the column name is preceded by a "D" (for delta). Other elements of the same event may be changed without submitting a new event card. The same method is used with the precompiler to set up the initial values in the matrix, except in this case all elements are set to zero unless input as a constant or specified as unknown.

Thus, a typical flow of data might be as follows:

```
DSCNT
  ASC=5000
  APS
  DLM
  ISP=301.5
```

This would result in setting the ASC weight at the event DSCNT equal to 5000, making the APS weight and the change to the LM weight for this event unknowns, and setting the ISP for this burn equal to 301.5. The solution to the problem, with this revised data, can then be obtained by submitting a single card with the word LAST.

Several additional features have been included which significantly increase the flexibility of the program. Following is a brief summary of these features. Details on their use are contained in the appendices.

Perhaps the most important of these is a set of print commands that gives the user some control over the printout. These are particularly useful when using a remote terminal, for they allow the user to eliminate some unwanted data and get a quicker answer. Some of the print commands are also helpful for finding errors in the input data. In addition, headings or titles may be specified to clearly identify the conditions for each run.

Sensitivities, or partial derivatives, of certain variables with respect to others are often required. Calculation of sensitivities is simplified by the SENS commands. Using these commands, a particular element of the matrix is designated. Then, a single command results in the calculation and printing of the sensitivity of each element in the matrix with respect to the designated element.

It is often necessary to plot one or more of the variables as a function of another. One might obtain the necessary data for this by running the program several times, each time inputting a new value of the desired variable. A much faster way is to use the STEP commands, which allow the user to designate a particular variable to be "stepped," or incremented, the amount of the increment, and the number of steps. Then, a single command results in the calculation and printing of all of the desired data.

Each input card is examined in detail to determine if the data or command it contains is in the correct format. This is done in order to minimize the possibility of unintentionally destroying previously specified information with an erroneous new input. When an error is detected, a message is printed, usually with enough information to readily identify the error.

When a user becomes proficient at using the program, even a short command can seem long. In order to minimize the amount of time spent typing or punching cards, abbreviated one- or two-letter equivalents have been provided for most commands. For example, the word PRINT may be replaced by the letter P.

The reader is referred to the examples at the end of Appendix B for typical runs illustrating these features. The examples are discussed in some detail in the body of Appendix B.

Acknowledgment

The original program, which was a simplified version of the current one, was written by V. L. Osborne. Portions of this memorandum were extracted from a draft memorandum by V. L. Osborne. Subroutine TEST was written by K. P. Klaasen. S. C. Wynn provided much programming consultation, and various members of Department 2013 offered helpful comments.

James L. Marshall, Jr.

2013-JLM-jab

J. L. Marshall, Jr.

Attachments
Appendices A through E

REFERENCES

1. Caldwell, S. F., "BCMASP Program Revisions," Bellcomm Memorandum for File B69 07020, Case 310, July 8, 1969.

APPENDIX A

USE OF PRECOMPILER

Use of the precompiler to generate a main program is illustrated in this appendix. The example used is the one presented earlier, a typical mission for the lunar module. Note that for brevity and clarity the mission and the number of weight components have been abbreviated. Standard system commands for the Univac 1108 are shown where required, but the reader is referred to the Exec 8 User's Manual for an explanation of their use. All of the precompiler commands and data required to generate a main program, called LM, are shown at the end of this appendix.† A detailed description of these commands follows.

After the usual system commands (RUN, ASG, etc.), the

```
@XQT WAP*WAP.PRE
```

card†† causes execution of the absolute element WAP*WAP.PRE to begin. This absolute element was previously formed from the Fortran program PRECPL. Unless modifications to the precompiler are desired, it is unnecessary to generate a new absolute element.†††

---

† The examples given in this appendix and Appendix B were run from a remote terminal. Those not familiar with the operation of the remote terminals should note that the lines typed by the user are in lower case letters (red on the original) while the computer response is in upper case (black) letters. In the body of these appendices, card images are shown in upper case letters as they would appear on a key-punched card.

†† Throughout this memorandum, the term "card" refers to a card image. It may be either an actual card or a line typed at a remote terminal.

††† If modifications are to be made to the precompiler, a new absolute element must be generated by replacing the @XQT WAP*WAP.PRE statement with cards of the following type:

```
@FØR WAP*WAP.PRECPL,PRECPL
   ----- edit deck -----
@MAP,I X,QUALIFIER*FILE.PRE
   IN PRECPL
@XQT QUALIFIER* FILE.PRE
```

The cards following the XQT card are used to set up the matrix to be used in the main program LM. This includes the specification of the names of the columns and events, and the assignment of initial values to each of the elements of the matrix. All eighty columns of each card are read by the precompiler. Blanks are squeezed out, except in fields delimited by asterisks (*) or dollar signs($). The choice is the user's and is indicated for each individual card by which of the two characters occurs first on the card. Once either does occur, the other loses its significance as a delimiter, so that such a delimited field started by one of these two characters must be closed by that same character. In the discussion that follows asterisks will be used, but dollar signs will serve the same purpose.

The set of cards read by the precompiler consists of two sections, a column specifications section and an event specifications section. Each section is required, and the entire column specifications section must precede the event specifications section.

The column specifications section is simply a series of cards, each of which specifies a component weight column. In its fullest form a column specification card is of the form

CNAME*ØNE/TWØ/THREE/* .

CNAME specifies the name of the column. Rules governing the choice of CNAME are as follows:

1. CNAME must have one to six characters.

2. The characters =, *, and $ are not allowed.

3. No two column names may have the same first five characters.

4. No two column names may be such that one can be formed from the other by prefixing the letter D and, if necessary, dropping the last character to maintain the six character limit.

5. The following words may not be specified as column names: DDDDDD, DELETE, DELTAV, DISCRE, DV, ELTAV, END, EVENT, HDG, I, ISCRE, ISP, LAST, LINEAR, RESTAR, STOP, V, and any six character word beginning with ELTAV, EVENT, or ISCRE.

In most cases, violation of one of these rules leads to an error message indicating that the specified word is illegal and the card has been ignored. However, in a few cases, the card will be interpreted as another type of command and the program will proceed accordingly. For example, if an attempt is made to use the word STØP as a column name, the program will interpret these four characters as a STØP command and terminate the execution (the STØP command will be discussed later).

The field *ØNE/TWØ/THREE* specifies the heading to be printed above this particular column when a printout is ultimately generated by the program being created.† Up to three lines of nine characters each are allowed for this heading, with each line being specified by a subfield of up to nine characters, and with a slash (/) separating one line from the next. Thus ØNE, TWØ, and THREE here represent subfields of up to nine characters. Such subfields will be printed, one per line and left-justified, in a nine-character field centered above the column. The field *ØNE/TWØ/THREE* may have an abbreviated form: any blank subfield may be omitted, and any slash may be omitted that does not have a nonblank subfield between it and the final delimiter; if there is nothing between the delimiters, they, too, may be omitted. Examples of the abbreviated forms are given in Appendix E.

There must be at least three and no more than ten columns specified in the column specifications section. If more than five columns are specified (seven if only the abbreviated headings are printed), the printout at a remote terminal will be jumbled unless one of the PRINT options is used to decrease the number of columns printed. See Appendices B and D. The last column specified always represents the sum of the other columns.

In the LM example, the column specifications section consists of the first five cards after the XQT command. They specify the columns named ASC, APS, DSC, DPS, and LM.

---

† Note that a PRINT option allows the user to specify whether the printed results will contain this heading or an abbreviated form using only the word CNAME. See Appendices B and D.

The event specifications section begins with the first card after the column specifications section, an event specification card. This card specifies an event in the weight history being set up, and in its fullest form is

EVENT ENAME*HEADING*CNAME.

The word EVENT identifies this card as an event specification card. ENAME specifies the name of the event. Rules governing the choice of ENAME are as follows:

1. ENAME must have one to six characters.

2. The characters =, *, and $ are not allowed.

3. No two events may have the same name.

4. An event name may not be identical to a column name, or a name formed by prefixing the letter D to a column name, dropping the last character, if necessary, to maintain the six character limit.

5. The following words may not be specified as event names: DELTAV, DISCRE, DV, END, HDG, I, ISP, LAST, LINEAR, RESTAR, STØP, V.

As is the case with column names, violation of these rules generally leads to an error message indicating that the card is illegal and has been ignored.

The field *HEADING* specifies the identification line to be printed for this particular event in the printouts to be generated by the program now being created.† Up to twenty-eight characters are allowed in this field, and the only illegal character is the character being used as delimiters (* or $). Slashes have no special significance. If the field is to be blank, nothing is required between the delimiters. In the printout to be generated this field will be printed to the extreme left on the same line as the initial data for this event.

CNAME is used to specify that there is to be a burn at this event and that the propellant for the burn will come from the column named CNAME. Obviously, CNAME must be one of the column names already specified, but it cannot be the last one.

---

† As is the case with the column headings, a PRINT option allows the user to specify that the printed results will contain an abbreviated identification line using only the word ENAME. The subheading, discussed later, would also be omitted. See Appendix D.

If there is to be no burn this event, this field must be blank. If the field is blank and the heading field specified by HEADING is also to be blank, then no delimiters are required - i.e., EVENT ENAME is equivalent to EVENT ENAME**.

All the cards between one event specification card and the next event specification card provide information pertaining to the first of the two events. Among these cards there may be up to three cards of the form

SUBHEADING.

Each of these cards specifies for this event a subheading of up to twenty-six characters the same way the *HEADING* field of the event specification card specifies an identification line. The subheadings are set up to be printed out in the order in which they have occurred. If a subheading field is to be blank and no subheading specification cards for the current event are to follow, then that subheading specification card may be omitted. But if subheading specification cards are to follow, then at least the delimiters must appear on that card, for the precompiler completely ignores blank cards. When the subheadings are printed out, the first one is printed two lines below the event identification line, the second one a line below that (on the same line as the weight changes for this event), and the third one a line below the second one. The subheadings are indented two spaces to the right of the first character of the event identification line.

The only other cards which may occur between one event specification card and the next have the purpose of inputting a number for some variable in the problem, specifying some variable as unknown, or specifying that consumables are to be treated linearly.

The cards that input a number are of the form

NAME = NUMBER

where NAME is a word of at least one character† which has to be recognized by the precompiler, and NUMBER is any Fortran floating point or integer constant. For NAME certain words are always recognized by the precompiler and others depend on the component weight column names specified in the column specifications section.

---

† If there are more than six characters in NAME, the precompiler looks at only the first six.

Those always recognized by the precompiler are DELTAV, DV, or V (the delta velocity), and ISP, or I (the specific impulse). (If there is no burn this event, DELTAV, DV, V, ISP, and I are illegal words.) The only other acceptable words for NAME are the component weight column names specified in the column specifications section and words derived from these names by prefixing them with the letter D (and dropping the last character of the resulting word if it has seven characters). The D prefix to the column name indicates a delta, or change, between the current event and the next event.

The number in NUMBER is input, in floating-point format, into the variable indicated by NAME; i.e., the column indicated by NAME at the current event.

The cards that specify variables as unknown are of the form

NAME

where NAME is the same as in the input cards discussed above. In this case, Fieldata code blanks are stored in the appropriate variable to specify that it is not known. All variables are set to zero unless input as a constant or specified as unknown.

The only other cards permitted before the next event specification card are the LINEAR and DISCRE(TE)† cards, which are of the form

LINEAR

and

DISCRE(TE).

These cards specify that all non-propulsive weight changes are to be treated linearly or discretely in the application of the rocket equation for the burn this event. LINEAR means the weight changes are to be used linearly during the burn. DISCRE(TE) means the weight changes are to be treated as if they were used immediately after completion of the burn. If there is no burn this event, these two cards are illegal. If there is a burn and neither of the two cards is specified, DISCRE(TE) is assumed. The user may cancel one of these specifications by submitting

---

† Throughout this memorandum, those characters enclosed in parentheses in a command may be omitted. Thus, the command DISCRE(TE) may be used as either DISCRETE or DISCRE.

the opposite specification afterwards. The last such card the precompiler reads for the particular event is the one that is put into the program that will be generated by the precompiler. When the program generated by the precompiler prints out the information pertaining to the problems it solves, burns that were treated linearly are identified by the word LINEAR or the letter L. No such word is printed for the weight changes that were treated discretely.

The cards between two event specification cards need not be in any special order, except that the subheading specification cards have to occur in the order determined by the printout desired. If there are more than one card referring to the same variable, the last one overrules the others.

There must be at least two events specified, and no more than fifty. The sample problem attached contains four events: DSCNT, TCHDWN, ASCNT, and DØCK.

Two other cards which the precompiler recognizes are the STØP and END cards, which are of the form

STØP

and

END.

These cards can be used any time, and as soon as the precompiler encounters either of them, it will print the message "RUN TERMINATED" and terminate the execution.

The end of the event specifications section is indicated by a LAST card, a card of the form

LAST.

Upon encountering a LAST card, the precompiler decides whether the information it has obtained so far is sufficient and whether the last event is in order (no burn is allowed for the final event, and neither are weight changes). If everything is acceptable, the precompiler immediately writes a Fortran main program onto Fortran logical unit 8 and prints the message "END OF PRECOMPILATION." Otherwise, it will inform the user that the burn and/or weight changes for the final event (but not the event itself) will be deleted if he submits another LAST card and that if he does not, then it will expect more information.

The precompiler is designed to be used from a remote terminal. Hence errors will be commented on as they occur and cards in error will be ignored. Furthermore, there is a special card which the user can submit to cause the precompiler to "forget" the last event or column specified. This is the DELETE card, which has the form

DELETE.

It instructs the precompiler to act as if it had never encountered the last event or column specification card and all information specified since. Upon encountering the DELETE card, the precompiler takes one of three possible courses of action: if any events have been specified, it deletes the last one; if no events have been specified but at least one column has been specified, it deletes the last column; and if no columns have been specified yet, it responds that there is nothing to delete. The DELETE card may be used repeatedly to back up as far as the user desires.

For the Fortran V compiler to be able to access the program generated by the precompiler and written onto Fortran logical unit 8, it is necessary to convert it to an element in binary form on some program file, usually on Fastrand. To accomplish this, a special program is used. This program is ØUTS, an absolute element that is part of the Bellcomm Apollo Simulation Program package (see Reference 1). Once the user has precompiled a set of data cards into card images on Fortran logical unit 8, he has to execute ØUTS by means of a control card of the form

@WAP*WAP.ØUTS,IL DUMMY,FILE.NAME

where DUMMY is a word of one to twelve characters the first of which has to be alphabetic, FILE is the program file on which the new element is to be placed, and NAME is the name of the new element. The I option is essential; the L option indicates that a listing of the element is to be produced (the L should not be used at a remote terminal). The result of this card is the creation of an element FILE.NAME which can be compiled by the Fortran V processor.

Two more steps are necessary. The first is compilation itself, which creates a relocatable element from the symbolic element FILE.NAME. The second is the creation of an absolute element. In the creation of the absolute element, nine relocatable elements must be included in addition to the relocatable element produced from FILE.NAME. These additional elements (subroutines BØDY, ITERAT, PØS, PRINT1, PRINT2, PRINT3, READ, RØCKET, SUMS, and TEST) are on program file WAP*WAP. The function of each of these subroutines is discussed in Appendix C.

The collection of these relocatable elements is accomplished with the MAP control card. In the example included here, the MAP card results in the creation of the absolute element JLM.LM. In Appendix B, a copy of JLM.LM, called WAP*WAP.LM, is used to illustrate how this absolute element can be used to solve specific problems. This LM absolute element has been saved along with the subroutines in a read only Fastrand file, WAP*WAP., for those who wish to experiment.

EXAMPLE A-1

USE OF PRECOMPILER

```
@run jlm,jlm,example

DATE: 002571        TIME: 165412
@asg,up jlm.
READY
@xqt wap*wap.pre
asc* ascent/ stage/ weight/*
 ASC* ASCENT/ STAGE/ WEIGHT/*
aps* ascent/ prop./ weight/*
 APS* ASCENT/ PROP./ WEIGHT/*
dsc*descent/ stage/weight/*
 DSC*DESCENT/ STAGE/WEIGHT/*
dps*descent/ prop./weight/*
 DPS*DESCENT/ PROP./WEIGHT/*
lm* total/    lm/weight/*
 LM* TOTAL/    LM/WEIGHT/*
event dscnt*begin descent*dps
 EVENT DSCNT*BEGIN DESCENT*DPS
*consumables*
 *CONSUMABLES*
*and propellant*
 *AND PROPELLANT*
asc=5755.1
 ASC=5755.1
aps=5207.6
 APS=5207.6
dsc=6118.2
 DSC=6118.2
dps=19508.5
 DPS=19508.5
lm
 LM
dasc=-97
 DASC=-97
ddsc=-30.2
 DDSC=-30.2
ddps
 DDPS
dlm
 DLM
dv=7013.9
 DV=7013.9
isp=302.3
 ISP=302.3
linear
 LINEAR
event tchdwn*touchdown on lunar surface*
 EVENT TCHDWN*TOUCHDOWN ON LUNAR SURFACE*
*jettisoned weight*
 *JETTISONED WEIGHT*
```

EXAMPLE A-1 (Continued)

```
asc
 ASC
aps
 APS
dsc
 DSC
dps
 DPS
lm
 LM
dasc=-19.8
 DASC=-19.8
ddsc
 DDSC
ddps
 DDPS
dlm
 DLM
event ascnt*begin powered ascent*aps
 EVENT ASCNT*BEGIN POWERED ASCENT*APS
*consumables*
 *CONSUMABLES*
*and propellant*
 *AND PROPELLANT*
asc
 ASC
aps
 APS
lm
 LM
dasc=-99.7
 DASC=-99.7
daps
 DAPS
dlm
 DLM
dv=6079.8
 DV=6079.8
isp=309.4
 ISP=309.4
event dock*dock with csm*
 EVENT DOCK*DOCK WITH CSM*
asc
 ASC
aps
 APS
lm
 LM
last
 LAST
 END OF PRECOMPILATION
```

EXAMPLE A-1 (Continued).

```
@wap*wap.outs,i x,lm
@for lm,lm
CYCLE 000 COMPILED BY 1201 BCS7F ON 25 JUN 71 AT 17:06:35.


   STORAGE USED: CODE(1) 000042; DATA(0) 000654; BLANK COMMON(2) 000000
       END OF COMPILATION:          NO  DIAGNOSTICS.
@elt,i lm
ELT PROCESSOR LEVEL      4
in lm
lib wap*wap.
@map lm,jlm.lm
22.01----06/25-17:07 -(0,)

START=037134, PROG SIZE(I/D)=15486/4013
@fin
    RUNID:  JLM       ACCOUNT: JLM              PROJECT: EXAMPLE
    TIME: 00:00:04.527     IN: 57      OUT: 0          PAGES:    2
    INITIATION  TIME:    16:54:12-JUN 25,1971
    TERMINATION TIME:    17:08:16-JUN 25,1971
    CORE-SECONDS: 99
    IO COUNT:      490
    CHARGE:      3.094
```

APPENDIX B

EXECUTION OF MAIN PROGRAM

The procedure described in Appendix A to generate a main program (an absolute element that will subsequently be referred to as the "program") need be repeated only if the names or quantities of the columns and events are to be changed. This appendix discusses the manner in which this program can be used to solve specific problems and gives some sample runs that were made from a remote terminal. Since all of the available commands are not illustrated in this appendix, and all aspects of the illustrated commands are not discussed here, the reader is referred to Appendix D for a more complete description of the commands.

Execution with the initial values of the matrix elements that were built into the program during precompilation is illustrated in Example 1. The XQT card is required by the Univac EXEC 8 system to begin the execution. The LAST card causes the program to evaluate the matrix. The resulting printout is shown in the example. The STØP (or END) card terminates the execution.

Note that the matrix printed in Example 1 differs slightly from the one presented in the body of this memorandum. Each element of the matrix that has a value of zero (or a magnitude less than 0.05) is left blank. It was indicated in the body of the memorandum that the matrix has columns devoted to delta velocities and specific impulses. This is, in fact, the way the matrix is organized within the program for computation. However, to conserve space and allow more columns, the printout has been arranged to show the delta velocity and specific impulse associated with an event directly below the event name. If the non-propulsive weight changes were assumed to be used in a linear fashion, the word "LINEAR" is also printed in this area.

Note also that the matrix printed in this example uses the abbreviated form of labels for the columns and events. The data will automatically be printed out in this format unless the program is told to use the complete labels as illustrated in Example 2. Note that since the character Δ is not available at the remote terminals, the symbol ΔV was replaced by "V. This happens only when the printout is obtained at a remote terminal, and since the complete labels are seldom desired when a remote terminal is used, this erroneous label was considered acceptable.

An essential feature of the program is the capability of defining a particular problem by specifying data inputs and dictating what variables are to be unknown. This is accomplished in much the same way as the same thing was accomplished during precompilation. First an event has to be specified by an event card, which is of the form

ENAME

where ENAME is the name of one of the events specified at precompilation with an event specification card. The event card specifies that the subsequent cards pertain to event ENAME. Event cards may appear anywhere in the deck and there are no restrictions on the order in which they may appear or how many times one event may be specified.

Data input cards may follow the event card to specify the value of a particular variable at that event. Several types of data input cards may be used. To assign a given value to an element, a card of the form

NAME = NUMBER

is used, where NAME is the name of the variable whose value at this event is to be changed, and NUMBER is any Fortran floating point or integer constant. The limitations on NAME are identical to those cited for precompilation. NAME should have from one to six characters; it may have more than six, but all beyond the first six are ignored. Acceptable words are (1) the column names that were specified by the column specification cards used by the precompiler in setting up the initial matrix, (2) words derived from the column names by prefixing them with the letter D (and dropping the last character of the resulting word if it has seven characters), indicating a delta, or change, between the current event and the next event, and (3) the names DELTAV, DV or V (the delta velocity), and ISP or I (the specific impulse of the engine). If there is no burn for this particular event, the names in group (3) above are not allowed. If this is the last event, the names in groups (2) and (3) are not allowed.

A card of the form

NAME = ØLD

causes NAME to take on the value that was calculated for it in the previous evaluation of the matrix.

The form

NAME = NAME op NUMBER op NUMBER . . .

results in NAME being set equal to its current value, modified by the sequence of operators and numbers. "op" may be either "+", "-", "*", or "/". The operations are performed in the order in which they are read; i.e., from left to right. If NAME is currently designated as an unknown, this form is not allowed.

The final type of data input card is of the form

NAME.

In this case, the variable NAME is designated as an unknown.

For all of these types of data cards, the data are stored the same way as during precompilation: for the first three types, the specified number is stored in floating-point format in the appropriate variable, and for the last type, Fieldata code blanks are stored in the variable to indicate that is is unknown and is to be calculated.

Two more cards which may be among those pertaining to a particular event are the LINEAR card and the DISCRE(TE)† card, of the forms

LINEAR

and

DISCRE(TE)†.

These cards specify whether non-propulsive weight changes are to be treated linearly or discretely in the application of the rocket equation. Their use is described in Appendices A and D. If no burn was specified for this event when the main program was precompiled, then these two cards are illegal (as are cards with the variable names DELTAV, DV, V, ISP, or I). If neither of these cards is used, the burn will be performed as specified during precompilation.

The data input cards following an event card need not be in any special order. If there are more than one card referring to the same variable, the last one overrules the others.

---

† Throughout this memorandum, those characters enclosed in parentheses in a command may be omitted. Thus, the command DISCRE(TE) may be used as either DISCRETE or DISCRE.

Example 3 is an illustration of how cards of this type may be used to modify the initial data that were built into the program at precompilation. In the first part of this example, the value of the ASC weight at event DSCNT was changed, and the unknowns in the matrix were recalculated using this new number. In the second part of the example, the problem was changed more significantly. The initial stage and propellant weights were specified in the first part, and the program calculated the amount of propellant required for each burn and the amount of propellant remaining after the burn for the specified ΔV's and ISP's. In the second part, the initial amount of propellant (e.g., tank capacity) and the final amount of propellant (e.g., residual, unusable propellant) were specified, and the maximum initial weight for each stage (for the given ΔV's and ISP's) was calculated. Notice also the PRINT=NØDATA command, which is useful when operating from a remote terminal. Normally, each line typed or appearing on a card is printed by the computer, which is a desirable feature when using cards. However, when a terminal is used, the lines typed by the user provide a record of the input, so time and space are saved by avoiding this extra printing.

The capability to easily change the data in the matrix is the basis for the flexibility of the program. However, this feature must be used cautiously to avoid setting up a problem that has no solution because the data specified is either insufficient or inconsistent. Example 4 is typical of the difficulties that can be encountered. In the first part, specification of the ASC weight at event DSCNT as an unknown results in insufficient data, and an appropriate message is printed. At this point, if the user recognizes his mistake, it can be corrected and rerun immediately. However, it is sometimes difficult to identify the error. To aid in finding such errors, several options have been provided. If the command PRINT=GØØF has been submitted, the entire matrix will be printed even though a complete solution was not obtained. This allows the user to determine just how far the computation progressed before a discrepancy was encountered or it was determined that the data were insufficient. This is illustrated in the first part of Example 4.

The last part of Example 4 shows what can happen if conflicting data are specified. In this case, the insufficient data problem was corrected by setting ASC at event DSCNT equal to 5800. However, the value specified for ASC at event DOCK was inconsistent with the data previously specified. The error message shows where the program ran into difficulty. If this message does not provide enough information to isolate the problem, the PRINT=GØØF command again can provide a means for obtaining more information. Since this command was still in effect (it could have been cancelled by using PRINT=NØGØØF), the entire matrix was printed when the discrepancy was encountered.

Several PRINT commands are provided to enhance the flexibility of the program, especially in its use from a terminal. The user is often interested in only a few elements of the matrix. Example 5 shows how to use the PRINT=CØLUMNS and PRINT=EVENTS commands to print only a selected portion of the matrix. The specified columns and events are deleted only from the printout; all elements are always considered during the calculations.

Note also that abbreviated forms for some of the commands are introduced in Example 5. Thus, "L" is equivalent to "LAST", "P" is equivalent to PRINT, etc. A complete list of valid abbreviations is given in Appendix E.

Occasionally it becomes desirable to know the values that are stored for various elements of the matrix. Example 6 shows how the PRINT=STATUS commands may be used for this purpose. The values printed are the current values, including any changes that have been made with input data cards.

The final command in Example 6 illustrates another feature. The characters for any input or command card do not have to appear in any particular set of columns. Blanks are "squeezed out" on all cards except those specifying headings. Thus, the final command in this example accomplished exactly the same function as the previous one.

Example 7 illustrates the SENS and DELTA commands in several different forms. Note that a special heading is printed to give information about the variable that was perturbed and the amount of the perturbation. Also, the printing format is changed to show more digits to the right of the decimal point.

The STEP and PRINT=TABLE commands are demonstrated in Example 8. The STEP command provides a means for evaluating the elements of the matrix a number of times in succession, with one of the variables incremented each time. Several examples of the use of the STEP command are given. In the first case, the simplest form is illustrated. Only the element being "stepped" is specified; the program assumes five steps and an increment of +1%. The normal printing format was retained (with all but one line deleted in this example to conserve space), resulting in the abbreviated matrix being printed five times. A more convenient printing format can be specified with the PRINT=TABLE commands. As shown, they make it possible to rearrange the data in a tabular form with up to ten columns (seven when using a remote terminal). The element to be printed in each column is specified by the user. Other forms for the STEP command are also illustrated.

Example 9 demonstrates the HDG commands. Up to five lines of heading may be specified, and the printing of the heading may be suppressed if desired. The specified lines of heading are destroyed only when new lines are supplied. Normally the heading will be centered; however, as shown, the HDG=NØCENTER command causes the heading to be left-justified.

Typical error messages are illustrated in Example 10. In some cases, the error is corrected in a subsequent card. Note that the PRINT=NØDATA command is overruled whenever an error occurs, but it remains in effect for any subsequent correct card. Although a card may contain more than one error, the printed comment refers only to the first one encountered by the program.

EXAMPLE B-1

SIMPLE EXECUTION

```
@run jlm,jlm,example

DATE: 062571       TIME: ¬175514
@xqt wap*wap.lm
last
 LAST


              ASC        APS        DSC        DPS        LM

 DSCNT       5755.1     5207.6     6118.2    19508.5    36589.4
  LINEAR      -97.0                 -30.2   -18759.2   -18886.4
  V=7013.90
  I= 302.30

 TCHDWN      5658.1     5207.6     6088.0      749.3    17703.0
              -19.8               -6088.0     -749.3    -6857.1

 ASCNT       5638.3     5207.6                           10845.9
              -99.7    -4957.2                           -5056.9
  V=6079.80
  I= 309.40

 DOCK        5538.6      250.4                            5789.0


stop
 STOP
 STOP OR END COMMAND---RUN TERMINATED
@fin
    RUNID:  JLM       ACCOUNT: JLM              PROJECT: EXAMPLE
    TIME: 00:00:00.185     IN: 5       OUT: 0          PAGES:    1
    INITIATION  TIME:    17:55:14-JUN 25,1971
    TERMINATION TIME:    17:56:37-JUN 25,1971
    CORE-SECONDS: 2
    IO COUNT:      26
    CHARGE:      0.094
```

EXAMPLE B-2

EXPANDED PRINTOUT

```
@run jlm,jlm,example

DATE: 062571        TIME: 180455
@xqt wap*wap.lm
print=labels
 PRINT=LABELS
last
 LAST
```

	ASCENT STAGE WEIGHT	ASCENT PROP. WEIGHT	DESCENT STAGE WEIGHT	DESCENT PROP. WEIGHT	TOTAL LM WEIGHT
BEGIN DESCENT "V=7013.90 ISP= 302.30 L	5755.1	5207.6	6118.2	19508.5	36589.4
CONSUMABLES AND PROPELLANT	-97.0		-30.2	-18759.2	-18886.4
TOUCHDOWN ON LUNAR SURFACE	5658.1	5207.6	6088.0	749.3	17703.0
JETTISONED WEIGHT	-19.8		-6088.0	-749.3	-6857.1
BEGIN POWERED ASCENT "V=6079.80 ISP= 309.40	5638.3	5207.6			10845.9
CONSUMABLES AND PROPELLANT	-99.7	-4957.2			-5056.9
DOCK WITH CSM	5538.6	250.4			5789.0

```
stop
 STOP
 STOP OR END COMMAND---RUN TERMINATED
```

EXAMPLE B-3

CHANGING DATA IN MATRIX

```
@xqt wap*wap.lm
dscnt
 DSCNT
asc=5800
 ASC=5800
last
 LAST
```

	ASC	APS	DSC	DPS	LM
DSCNT	5800.0	5207.6	6118.2	19508.5	36634.3
LINEAR	-97.0		-30.2	-18782.2	-18909.4
V=7013.90					
I= 302.30					
TCHDWN	5703.0	5207.6	6088.0	726.3	17724.9
	-19.8		-6088.0	-726.3	-6834.1
ASCNT	5683.2	5207.6			10890.8
	-99.7	-4977.7			-5077.4
V=6079.80					
I= 309.40					
DOCK	5583.5	229.9			5813.4

```
print=nodata
 PRINT=NODATA
dscnt
asc
dsc
aps=aps+50
tchdwn
dps=600
dock
aps=old
last
```

	ASC	APS	DSC	DPS	LM
DSCNT	5859.4	5257.6	6254.5	19508.5	36880.0
LINEAR	-97.0		-30.2	-18908.5	-19035.7
V=7013.90					
I= 302.30					
TCHDWN	5762.4	5257.6	6224.3	600.0	17844.3
	-19.8		-6224.3	-600.0	-6844.1
ASCNT	5742.6	5257.6			11000.2
	-99.7	-5027.7			-5127.4
V=6079.80					
I= 309.40					
DOCK	5642.9	229.9			5872.8

```
stop
 STOP OR END COMMAND---RUN TERMINATED
```

EXAMPLE B-4

ERROR MESSAGES AND PRINT=GOOF COMMAND

```
@xqt wap*wap.lm
print=nodata
 PRINT=NODATA
dscnt
asc
last
 NO SOLUTION--DATA INSUFFICIENT
print=goof
last
 NO SOLUTION--DATA INSUFFICIENT


             ASC        APS        DSC        DPS        LM

 DSCNT       UNKNWN     5207.6     6118.2     19508.5    UNKNWN
  LINEAR      -97.0                 -30.2     UNKNWN     UNKNWN
  V=7013.90
  I= 302.30

 TCHDWN      UNKNWN     5207.6     6088.0     UNKNWN     UNKNWN
              -19.8                -6088.0     UNKNWN     UNKNWN

 ASCNT       UNKNWN     5207.6                            UNKNWN
              -99.7     UNKNWN                            UNKNWN
  V=6079.80
  I= 309.40

 DOCK        UNKNWN     UNKNWN                            UNKNWN

dscnt
asc=5800
dock
asc=5000
last
 ***DISCREPANCY ENCOUNTERED IN SUMS, EVENT ASCNT
 SUM=    5000.00 SUMTST=    5583.50 A=    5683.20     -99.70


             ASC        APS        DSC        DPS        LM

 DSCNT       5800.0     5207.6     6118.2     19508.5    36634.3
  LINEAR      -97.0                 -30.2     UNKNWN     UNKNWN
  V=7013.90
  I= 302.30

 TCHDWN      5703.0     5207.6     6088.0     UNKNWN     UNKNWN
              -19.8                -6088.0     UNKNWN     UNKNWN

 ASCNT       5683.2     5207.6                            10890.8
              -99.7     UNKNWN                            UNKNWN
  V=6079.80
  I= 309.40

 DOCK        5000.0     UNKNWN                            UNKNWN

stop
 STOP OR END COMMAND---RUN TERMINATED
```

EXAMPLE B-5

PRINTING AN ABBREVIATED MATRIX

```
@xqt wap*wap.lm
print=nodata
 PRINT=NODATA
last

             ASC        APS        DSC        DPS        LM

 DSCNT       5755.1     5207.6     6118.2    19508.5    36589.4
  LINEAR      -97.0                -30.2   -18759.2   -18886.4
  V=7013.90
  I= 302.30

 TCHDWN      5658.1     5207.6     6088.0      749.3    17703.0
              -19.8               -6088.0     -749.3    -6857.1

 ASCNT       5638.3     5207.6                           10845.9
              -99.7    -4957.2                           -5056.9
  V=6079.80
  I= 309.40

 DOCK        5538.6      250.4                            5789.0

print=columns-aps-dps
print=events-tchdwn-ascnt
last

             ASC        DSC        LM

 DSCNT       5755.1     6118.2    36589.4
  LINEAR      -97.0      -30.2   -18886.4
  V=7013.90
  I= 302.30

 DOCK        5538.6                5789.0

p=e-dscnt+tchdwn
p=c+all-dps
l

             ASC        APS        DSC        LM

 TCHDWN      5658.1     5207.6     6088.0    17703.0
              -19.8               -6088.0    -6857.1

 DOCK        5538.6      250.4                5789.0

end
 STOP OR END COMMAND---RUN TERMINATED
```

EXAMPLE B-6

STATUS COMMANDS

```
@xqt wap*wap.lm
p=nd
 P=ND
print=status,all

                ASC        APS        DSC        DPS        LM

 DSCNT         5755.1     5207.6     6118.2    19508.5     UNKNWN
  LINEAR        -97.0                 -30.2     UNKNWN     UNKNWN
  V=7013.90
  I= 302.30

 TCHDWN        UNKNWN     UNKNWN     UNKNWN     UNKNWN     UNKNWN
                -19.8                UNKNWN     UNKNWN     UNKNWN

 ASCNT         UNKNWN     UNKNWN                           UNKNWN
                -99.7     UNKNWN                           UNKNWN
  V=6079.80
  I= 309.40

 DOCK          UNKNWN     UNKNWN                           UNKNWN

p=c-all+asc+lm
p=e-all+dscnt+dock
print=status,part

                ASC        LM

 DSCNT         5755.1     UNKNWN
  LINEAR        -97.0     UNKNWN
  V=7013.90
  I= 302.30

 DOCK          UNKNWN     UNKNWN

print=status,lm/dock,dasc/ascnt
 LM     (DOCK  )=UNKNWN
 DASC   (ASCNT )=     -99.7
ascnt
dasc=-300
p=s,dasc/ascnt,dps/dock,lm/dock
 DASC   (ASCNT )=    -300.0
 DPS    (DOCK  )=
 LM     (DOCK  )=UNKNWN
   p     =  s,     d a s c/a s c n t,     dp s/do ck,lm /doc k
 DASC   (ASCNT )=    -300.0
 DPS    (DOCK  )=
 LM     (DOCK  )=UNKNWN
stop
 STOP OR END COMMAND---RUN TERMINATED
```

EXAMPLE B-7

SENSITIVITY COMMANDS

```
@xqt wap*wap.lm
p=nd
 P=ND
sens=isp/dscnt
sens=yes
last
```

SENSITIVITIES TO ISP (DSCNT), DELTA= 3.023

	ASC	APS	DSC	DPS	LM
DSCNT					
LINEAR				42.0526	42.0526
V=					
I= 1.0000					
TCHDWN				42.0526	42.0525
				-42.0526	-42.0525
ASCNT					
V=					
I=					
DOCK					

```
p=e-all+dscnt+tchdwn
sens=i/dscnt-1.5
l
```

SENSITIVITIES TO ISP (DSCNT), DELTA= -1.500

	ASC	APS	DSC	DPS	LM
DSCNT					
LINEAR				42.4558	42.4558
V=					
I= 1.0000					
TCHDWN				42.4558	42.4558

EXAMPLE B-7 (Continued)

```
s=isp/dscnt+10*%
1

      SENSITIVITIES TO ISP   (DSCNT ), DELTA=   30.230

              ASC        APS        DSC        DPS        LM

 DSCNT
  LINEAR                                      39.7653    39.7653
  V=
  I=   1.0000

 TCHDWN                                       39.7653    39.7653

sens=no
delta=yes
1

  CHANGES DUE TO CHANGE IN ISP   (DSCNT ), DELTA=   30.230

              ASC        APS        DSC        DPS        LM

 DSCNT
  LINEAR                                     1202.104   1202.104
  V=
  I=   30.230

 TCHDWN                                      1202.104   1202.104

stop
 STOP OR END COMMAND---RUN TERMINATED
```

EXAMPLE B-8

STEP AND TABLE COMMANDS

```
@xqt wap*wap.lm
p=nd
 P=ND
p=e-all+dock
step=asc/dscnt
step=yes
l
```

	ASC	APS	DSC	DPS	LM
DOCK	5538.6	250.4			5789.0

	ASC	APS	DSC	DPS	LM
DOCK	5596.2	224.1			5820.3

	ASC	APS	DSC	DPS	LM
DOCK	5653.7	197.8			5851.5

	ASC	APS	DSC	DPS	LM
DOCK	5711.3	171.5			5882.8

	ASC	APS	DSC	DPS	LM
DOCK	5768.8	145.2			5914.0

EXAMPLE B-8 (Continued)

```
print=table,1/asc/dscnt,2/daps/ascnt,3/lm/dscnt
print=table
last
```

ASC (DSCNT)	DAPS (ASCNT)	LM (DSCNT)
5755.1	-4957.2	36589.4
5812.7	-4983.5	36647.0
5870.2	-5009.8	36704.5
5927.8	-5036.1	36762.1
5985.3	-5062.4	36819.6

```
step=asc/dscnt/3+100
1
```

ASC (DSCNT)	DAPS (ASCNT)	LM (DSCNT)
5755.1	-4957.2	36589.4
5855.1	-5002.9	36689.4
5955.1	-5048.6	36789.4

```
p=t,2/delete,5/aps/dock
st=asc/dscnt/3-2*percen
1
```

ASC (DSCNT)	LM (DSCNT)	APS (DOCK)
5755.1	36589.4	250.4
5640.0	36474.3	303.0
5524.9	36359.2	355.6

```
step=isp/dscnt
p=t,6/isp/dscnt,1/ddps/tchdwn
1
```

DDPS (TCHDWN)	LM (DSCNT)	APS (DOCK)	ISP (DSCNT)
-749.3	36589.4	250.4	302.3
-876.4	36589.4	250.4	305.3
-1002.0	36589.4	250.4	308.3
-1125.9	36589.4	250.4	311.4
-1248.3	36589.4	250.4	314.4

```
stop
 STOP OR END COMMAND---RUN TERMINATED
```

EXAMPLE B-9

HEADING COMMANDS

```
@xqt wap*wap.lm
p=nd
 P=ND
p=e-all+dock
l

                ASC        APS        DSC        DPS        LM

 DOCK           5538.6      250.4                            5789.0

print=hdg
hdg
example of heading
l

                    EXAMPLE OF HEADING

                ASC        APS        DSC        DPS        LM

 DOCK           5538.6      250.4                            5789.0

hdg=3
h
      example of heading---line 1
line 2
                                                 line 3
last

                EXAMPLE OF HEADING---LINE 1
                          LINE 2
                          LINE 3

                ASC        APS        DSC        DPS        LM

 DOCK           5538.6      250.4                            5789.0
```

EXAMPLE B-9 (Continued)

```
hdg=2
last

                    EXAMPLE OF HEADING---LINE 1
                              LINE 2

              ASC        APS        DSC        DPS        LM

 DOCK         5538.6     250.4                            5789.0

h=3
h=nocenter
last

        EXAMPLE OF HEADING---LINE 1
 LINE 2
                                                LINE 3

              ASC        APS        DSC        DPS        LM

 DOCK         5538.6     250.4                            5789.0

stop
 STOP OR END COMMAND---RUN TERMINATED
@fin
    RUNID:  JLM       ACCOUNT: JLM              PROJECT: EXAMPLE
    TIME: 00:00:02.438     IN: 80      OUT: 0         PAGES:
    INITIATION  TIME:    18:58:37-JUN 25,1971
    TERMINATION TIME:    19:30:35-JUN 25,1971
    CORE-SECONDS: 16
    IO COUNT:      86
    CHARGE:      0.617
```

EXAMPLE B-10

ERROR MESSAGES

```
@xqt wap*wap.lm
nd
 ND
 'ND     ' NOT RECOGNIZED
p=nd
 P=ND
asc=5800
 ASC=5800
 NO EVENT SPECIFIED YET---ABOVE LINE IGNORED
dscnt
ascnt=5800
 ASCNT=5800
 ABOVE LINE IN ERROR---IGNORED
asc=5800
docking
 'DOCKIN' NOT RECOGNIZED
dock
isp=400
 ISP=400
 NO BURN THIS EVENT---ABOVE LINE IGNORED
dlm=-50
 DLM=-50
 CHANGE TO FINAL EVENT---ABOVE LINE IGNORED
xxxxxxxx
 'XXXXXX' NOT RECOGNIZED
old
 'OLD    ' NOT RECOGNIZED
aps=old
asc=asc+100
 ASC=ASC+100
 VARIABLE IS AN UNKNOWN---ABOVE LINE IGNORED
dsc=dsc+asc
 DSC=DSC+ASC
 ABOVE LINE IN ERROR---IGNORED
print=events--dscnt
 ABOVE LINE IGNORED (TWO OPERATORS IN SUCCESSION)
asc==50
 ABOVE LINE IGNORED (TWO EQUAL SIGNS)
=28
 ABOVE LINE IGNORED (FIRST CHARACTER IS AN EQUAL)
asc=50.2.3
 ABOVE LINE IGNORED (TWO DECIMALS IN ONE NUMBER)
5=2.3
 ABOVE LINE IGNORED (FIRST WORD IS A NUMBER)
```

EXAMPLE B-10 (Continued)

```
print=events-dscnt+jjjj
 PRINT=EVENTS-DSCNT+JJJJ
 PRINT COMMAND IN ERROR---INVALID PART IGNORED
print1
 'PRINT1' NOT RECOGNIZED
print=
 PRINT=
 PRINT COMMAND IN ERROR---IGNORED
p=
 P=
 PRINT COMMAND IN ERROR---IGNORED
print=yes
 PRINT=YES
 PRINT COMMAND IN ERROR---IGNORED
sens=lm/dscnt
sens=yes
last
 LAST
 PERTURBING VARIABLE IS AN UNKNOWN---'LAST' COMMAND IGNORED
sens=asc/aps
 SENS=ASC/APS
 SENS OR DELTA COMMAND IN ERROR---IGNORED
print=table,,2/lm/dscnt
 ABOVE LINE IGNORED (TWO OPERATORS IN SUCCESSION)
print=table,lm/dscnt/2
 PRINT=TABLE,LM/DSCNT/2
 PRINT COMMAND IN ERROR---INVALID PART IGNORED
print=table,2/lm/dscnt
print=table,3/lm
 PRINT=TABLE,3/LM
 PRINT COMMAND IN ERROR---INVALID PART IGNORED
p=tabl,2/lm/dscnt
 P=TABL,2/LM/DSCNT
 PRINT COMMAND IN ERROR---IGNORED
step=asc/asc
 STEP=ASC/ASC
 STEP COMMAND IN ERROR---IGNORED
print=status,asc
 PRINT=STATUS,ASC
 PRINT COMMAND IN ERROR---IGNORED
stop
 STOP OR END COMMAND---RUN TERMINATED
efin
    RUNID:  JLM       ACCOUNT: JLM              PROJECT: EXAMPLE
    TIME: 00:00:01.251     IN: 54      OUT: 0          PAGES:   3
    INITIATION  TIME:    20:03:30-JUN 25,1971
    TERMINATION TIME:    20:15:58-JUN 25,1971
    CORE-SECONDS: 6
    IO COUNT:      38
    CHARGE:      0.271
```

APPENDIX C

PROGRAMS AND SUBPROGRAMS SUMMARY AND LISTINGS†

PRECPL

PRECPL is the precompiler, which is used for the initial set-up of the problem. The user supplies data specifying the names of the columns and events and the initial value of each element of the matrix (including the delta-velocity and specific impulse for each burn). Each element may be specified as an unknown or a constant. PRECPL uses these data to generate a main Fortran program that is used with the subprograms to solve the specified problem. The only purpose of this generated main program is to supply the initial conditions and the names of the columns and events to the subprograms. Control of the execution is via subprogram BØDY, which is called by the main program. The main program generated for the examples shown in this memorandum, called LM, is listed at the end of this appendix after the listing of PRECPL.

PRECPL is self-contained and does not require any of the other subprograms.

BØDY

BØDY is the primary part of the program. It is called by the generated main program, and it governs the overall flow of the execution. It controls the reading of all input data and determines what to do with the data. First, each input command or data card is examined for errors in format. Then BØDY proceeds to carry out the command or update the matrix with the new data. When a "LAST" card is received, BØDY begins the process of solving for the unknowns in the matrix. The actual solution is accomplished by a series of calls to other subprograms. If a solution is obtained it is printed in accordance with previously given instructions. If there is no solution or an error has been made, this information is printed. Re-initialization then occurs, and the program is ready for new commands or data.

ITERAT

ITERAT is used whenever it is necessary to perform the iteration process described in the body of this memorandum. In this case ITERAT performs the following functions. First,

---

† Listings are at the end of this appendix.

an unknown variable in the matrix is selected and assigned a value. Then an already known element is selected as a target or comparison variable. This element is made an unknown, but its actual known value is saved. The modified matrix is returned to BØDY where an attempt is made to solve it with the assumed value for the unknown. If a solution thus obtained results in a match between the newly calculated and the desired values of the target variable, the assumed value is the correct one. If a match does not occur, an appropriate modification is made to the assumed value and the process is repeated. This iteration is continued until the proper solution is found, or until it is determined that it cannot be found with this combination of elements for the iterating and target variables. Other combinations are then tried until the proper solution is found.

PØS

Subroutine PØS is used to center the heading. Blanks at the beginning and end of the heading are ignored, but the spacing within the heading is retained.

PRINT1

PRINT1 is used to print the output matrix when the abbreviated names (not the more lengthy labels) are desired. The proper formats are set up, depending on whether the results to be printed are sensitivities or the normal values of the elements of the matrix. If an abbreviated form of the matrix (with some columns or events deleted) has been specified, the required conditions are set up. Then the headings and values of the elements of the matrix are printed in accordance with previously specified instructions.

PRINT2

PRINT2 functions exactly like PRINT1, except the more detailed labels are used, resulting in a more complete description at the expense of space.

PRINT3

PRINT3 is similar to PRINT1, but it is used when the tabular form of output is desired.

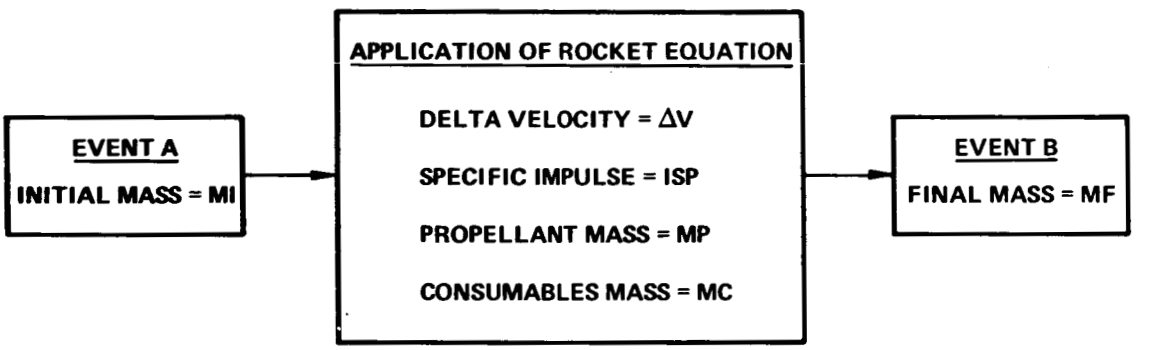
READ

Subroutine READ is used to read in data and commands. All 80 columns of a card are read and then processed. Blanks are ignored. The information is checked for errors in format.

Information returned to the calling program include the number of "words" on the card, the operator associated with each word (+, -, *, /, or comma), and the type of word (alphanumeric characters or all numbers). Both integer and floating point numbers are accepted by READ, but intergers are converted to floating point before the return to the calling program. For alphanumeric variables only the first six characters are considered.

RØCKET

Subroutine RØCKET is used for solution of the rocket equation. RØCKET solves the type of problem diagrammed below.



This represents a rocket stage with an initial mass, MI, and an engine with an efficiency characterized by the specific impulse, ISP. If the engine burns a propellant mass, MP, and the amount of consumables (oxygen, water, ablative material, etc.) used is MC, then the final mass is MF = MI-MP-MC and the vehicle experiences a change in velocity of ΔV. These six variables, in any combination of knowns and unknowns, are provided to RØCKET. If they appear in a combination that cannot be solved because there are too many unknowns, RØCKET returns to the calling program with a logical variable, DØNE, set to FALSE. If all six of the input variables have known values, ROCKET checks them to be sure they are internally consistent. If they are, DØNE is set to TRUE before returning. If they are not consistent, an error message (see last paragraph of this section) is printed with a summary of the data and the event they were associated with, and control is returned to the calling program. If all six input variables are not known, but there is sufficient data to determine one or more of the unknowns, RØCKET chooses the proper form of the rocket equation, and returns to the calling program with the result and with DØNE set to TRUE.

RØCKET treats the consumables mass MC in one of two ways, as specified in the calling argument LINEAR. If LINEAR is FALSE, the consumables mass is assumed to be removed discretely after the burn is completed (i.e., enough propellant must be provided to carry MC throughout the burn). If LINEAR is TRUE, the consumables mass is assumed to be removed linearly during the burn. The equations solved are as follows:

A. For LINEAR = FALSE

$$MI = MF+MP+MC$$

$$\Delta V = ISP*G*LN\left(\frac{MI}{MI-MP}\right)$$

B. For LINEAR = TRUE

$$MI = MF+MP+MC$$

$$\Delta V = \frac{ISP*G*MP}{MI-MF}*LN\left(\frac{MI}{MF}\right)$$

For some combinations of the six input variables a transcendental equation is involved, and iteration is necessary to find a solution. If this is attempted but convergence is not achieved, the error message

```
***NEWTON-RAPHSON FAILED, EVENT ENAME
```

is printed, where ENAME is the name of the event involved.

If the data provided to RØCKET are inconsistent, an error message of the form

```
***DISCREPANCY ENCOUNTERED IN ROCKET, EVENT ENAME
DV=    7013.90 DVTEST=    6961.27 ISP=     302.30
MI=   37000.00 MP=   18872.80 MC=     127.20 MF=   18000.00
```

is printed where the numbers shown are typical. Again, ENAME is the name of the event involved. DV is the delta velocity that was provided to RØCKET. DVTEST is the delta velocity that was calculated from the other quantities that were provided to RØCKET. ISP, MI, MP, MC, and MF are the variables that were defined earlier in this section.

SUMS

Subroutine SUMS is given an array, A, of N numbers plus a variable SUM, which represents the sum of the numbers in A. These N+1 variables may be given in any combination of knowns and unknowns. If SUM and all of the A's have values, SUMS checks the validity of SUM. If a discrepancy is found, an error message

is printed with a summary of the data and the event they were associated with, and control is returned to the calling program. If SUM and the A's are consistent, SUMS returns with a logical variable DØNE set to TRUE. If only one of the N+1 variables is unknown, SUMS calculates the unknown variable and returns to the calling program with DØNE set to TRUE. If there is more than one unknown, SUMS returns to the calling program with DØNE set to FALSE.

The error message that is printed if the data are inconsistent is of the form

```
***DISCREPANCY ENCOUNTERED IN SUMS, EVENT ENAME
SUM= 5000.0 SUMTST= 5583.5 A= 5683.20  -99.70
```

ENAME is the name of the event involved. SUM is the sum that was provided to SUMS. SUMTST is the sum that was calculated from the other quantities that were provided to SUMS. A is the array of N numbers.

TEST

TEST is called by BØDY whenever an iterative procedure is required to solve the problem. TEST determines whether or not sufficient information is available to obtain a complete solution to the problem. The number of independent, known elements in the matrix must equal a critical value if the matrix is to be completely determined. This critical number is equal to the total number of elements in the matrix minus the total number of independent equations relating these elements. TEST begins by calculating this critical number. Each element whose value is known is then examined to determine whether or not it is independent of all other known elements, and the total number of independent, known elements is found. If this number is equal to the critical number, a complete solution can be found. In this case, control is returned to BØDY which procedes to solve the problem using iteration. If the total number of independent known elements is less than the critical number, the data input to the program is insufficient to completely solve the matrix. In this case, control is returned to BØDY where the message

```
NØ SOLUTION--DATA INSUFFICIENT
```

is printed out, and no further attempt is made toward solution.

PRECPL

```
      REAL                    MF(50),                 MPF(50)
C
      LOGICAL                 EVENTS,                 FLAG
     .,FLAG1,                 FLAG2,                  FLAG3
     .,LINFLG(50),            NEGFLG
C
      DIMENSION               V(12),                  VAR(12,101)
C
      INTEGER                 BLANK /'      '/,       CDWRD(10)/10*'D'/
     .,CH(80),                CHEAD(2,3,10)/60*' '/,CHSAV
     .,COMMA /',     '/,      CWORD(10),              DELETE/'DELETE'/
     .,DISCRE/'DISCRE'/,      DOLLAR/'$     '/,       END   /'END   '/
     .,EQUAL /'=     '/,      EVENT /'EVENT '/,       IBURN(50)
     .,IC(5) /3*1,2*2 /,      ISUBHD(50),             IVAR(12,101)
     .,LAST  /'LAST  '/,      LINEAR/'LINEAR'/,       MINUS /'-     '/
     .,PLUS  /'+     '/,      POINT /'.     '/,       RHEAD(5,50)/250*' '/
     .,RSUBHD(5,3,50)/750*' '/,                       RWORD(50)
     .,SLASH /'/     '/,      STAR  /'*     '/,       STOP  /'STOP  '/
     .,TAG(13),               TAG1(5)/'DELTAVDV    V     ISP   I  '/
     .,GO/'GO'/
C
      DATA (TAG(I),I=6,13)/'DISCREHDG   LAST  LINEARRESTARDDDDDDELTAV I
     .SCRE'/
C
      EQUIVALENCE             (TAG,TAG1),             (VAR,IVAR)
C
C               READ AND PRINT A CARD
C
   25 READ(5,26,END=245)CH
   26 FORMAT(80A1)
      PRINT 27,CH
   27 FORMAT(' ',80A1)
C
C               INITIALIZE
C
      J=0
      K1=0
      K2=0
      K3=0
      K4=0
      L=-2
      FLAG=.FALSE.
      NEGFLG=.FALSE.
C
C               CHECK CHARACTERS ON CARD FOR VALIDITY
C
      DO 44 I=1,80
      IF(CH(I).NE.STAR.AND.CH(I).NE.DOLLAR)GO TO 35
      IF(L.NE.-2.AND.CH(I).NE.CHSAV)GO TO 43
      L=L+1
      IF(L),30,260
      CHSAV=CH(I)
      K1=J+1
      GO TO 44
   30 K2=J
      GO TO 44
```

```
   35 IF(L.EQ.-1)GO TO 43
      IF(CH(I).EQ.BLANK)GO TO 44
      IF(CH(I).NE.EQUAL)GO TO 40
      IF(K3.NE.0.OR.J.EQ.0.OR.L.EQ.0)GO TO 260
      K3=J+1
      GO TO 44
   40 IF(K3.EQ.0)GO TO 43
      IF(CH(I).EQ.PLUS)GO TO 41
      IF(CH(I).NE.MINUS)GO TO 42
      NEGFLG=.TRUE.
   41 IF(FLAG.OR.K3.LE.J.OR.K4.NE.0)GO TO 260
      FLAG=.TRUE.
      GO TO 44
   42 IF(CH(I).NE.POINT)GO TO 43
      IF(K4.NE.0)GO TO 260
      K4=J
      GO TO 44
   43 J=J+1
      CH(J)=CH(I)
   44 CONTINUE
      IF(J.EQ.0.AND.K1.EQ.0)GO TO 25
      IF(L.EQ.-1)GO TO 260
       KK=0
       CALL PACK(1,3,ITEMP)
       IF(ITEMP.EQ.END)KK=3
       CALL PACK(1,4,ITEMP)
       IF(ITEMP.EQ.STOP)KK=4
       IF(KK.EQ.0)GO TO 46
C
C                STOP OR END COMMAND
C
       IF(K1.EQ.KK+1.OR.K3.EQ.KK+1)GO TO 260
       IF(J.NE.KK.OR.K1.NE.0)GO TO 46
      PRINT 45
   45 FORMAT(' RUN TERMINATED')
      STOP
   46 FLD(24,6,ITEMP)=FLD(0,6,CH(5))
      IF(ITEMP.EQ.EVENT.AND.(K1.EQ.0.OR.K1.GT.5))GO TO 110
      FLD(30,6,ITEMP)=FLD(0,6,CH(6))
      IF(ITEMP.NE.DELETE.OR.(K1.NE.0.AND.K1.LT.7).OR.J.EQ.5)GO TO 57
C
C                DELETE COMMAND
C
      IF(K1.NE.0.OR.K3.NE.0)GO TO 260
      IF(EVENTS)GO TO 51
      IF(ICOL.NE.0)GO TO 48
      PRINT 47
   47 FORMAT(' NOTHING TO DELETE - ABOVE LINE IGNORED')
      GO TO 25
   48 PRINT 49
   49 FORMAT(' COLUMN DELETED')
      DO 50 I=1,2
      DO 50 K=1,3
   50 CHEAD(I,K,ICOL)=BLANK
      ICOL=ICOL-1
      GO TO 25
   51 PRINT 52
```

```
   52 FORMAT(' EVENT DELETED')
      DO 53 I=1,5
      RHEAD(I,IROW)=BLANK
      DO 53 K=1,3
   53 RSUBHD(I,K,IROW)=BLANK
      I2=2*IROW+1
      M=IBURN(IROW)
      IF(FLAG1)VAR(M,I2)=0.
      VAR(ICOL,I2)=0.
      ICP2=ICOL+2
      DO 54 I=1,2
      I2=I2-1
      DO 54 I1=1,ICP2
   54 VAR(I1,I2)=0.
      IBURN(IROW)=0
      LINFLG(IROW)=.FALSE.
      IROW=IROW-1
      EVENTS=IROW.NE.0
      IF(.NOT.EVENTS)GO TO 25
      M=IBURN(IROW)
      FLAG1=M.NE.0
      IF(FLAG1)VAR(M,I2)=MPF(IROW)
      VAR(ICOL,I2)=MF(IROW)
      DO 55 I=0,1
      DO 55 I1=1,ICP2
   55 IF(IVAR(I1,I2-I).NE.0)FLAG2=.TRUE.
      FLAG3=.FALSE.
      GO TO 25
   57 IF(EVENTS)GO TO 155
      IF(K1.EQ.1.OR.K1.GT.7.OR.K2.NE.0.AND.K2.NE.J.OR.K2.EQ.0.AND.J.GT.6
     . .OR.K3.NE.0)GO TO 260
C
C       COLUMN SPECIFICATION CARD
C
        CALL PACK(1,4,ITEMP)
        IF(ITEMP.NE.LAST)GO TO 58
        IF(K1.EQ.5)GO TO 260
        IF(J.EQ.4)GO TO 236
   58 IF(ICOL.NE.10)GO TO 60
      PRINT 59
   59 FORMAT(' TOO MANY COLUMNS CALLED FOR - ONLY 10 ALLOWED - ABOVE LIN
     .E IGNORED')
      GO TO 25
   60 K=J
      IF(K1.NE.0)K=K1-1
      CALL PACK(1,K,ITEMP)
       DO 61 I=1,13
   61 IF(ITEMP.EQ.TAG(I))GO TO 63
      IF(ICOL.EQ.0)GO TO 65
      ITMP1=FLD(0,30,ITEMP)
       IF(ITMP1.EQ.FLD(0,30,TAG(12)).OR.ITMP1.EQ.FLD(0,30,TAG(13)))GO TO
       63
      DO 62 I=1,ICOL
       IF(ITMP1.EQ.FLD(6,30,CWORD(I)).AND.FLD(0,6,CWORD(I)).EQ.
     .FLD(0,6,DELETE))GO TO 63
   62 IF(ITMP1.EQ.FLD(0,30,CWORD(I)).OR.ITEMP.EQ.CDWRD(I))GO TO 63
      GO TO 65
```

```
171          63  PRINT 64,ITEMP
172          64  FORMAT(' COLUMN NAME ''',A6,''' IS ILLEGAL - ABOVE LINE IGNORED')
173              GO TO 25
174          65  IF(K1.GE.K2)GO TO 100
175              IF(CH(K2).EQ.SLASH)GO TO 66
176              K2=K2+1
177              CH(K2)=SLASH
178          66  NSLASH=0
179              KFIRST=K1
180              DO 70 K=K1,K2
181              IF(CH(K).NE.SLASH)GO TO 70
182              IF(NSLASH.EQ.3.OR.K-KFIRST.GT.9)GO TO 260
183              NSLASH=NSLASH+1
184              KFIRST=K+1
185          70  CONTINUE
186              KFIRST=K1
187              I2=1
188              ICOL=ICOL+1
189          75  DO 80 K=KFIRST,K2
190              IF(CH(K).EQ.SLASH)GO TO 85
191          80  CONTINUE
192          85  K=K-1
193              IF(KFIRST.GT.K)GO TO 95
194              IF(K-KFIRST.GT.5)GO TO 90
195              CALL PACK(KFIRST,K,CHEAD(1,I2,ICOL))
196              GO TO 95
197          90  CALL PACK(KFIRST,KFIRST+5,CHEAD(1,I2,ICOL))
198              CALL PACK(KFIRST+6,K,CHEAD(2,I2,ICOL))
199          95  KFIRST=K+2
200              IF(KFIRST.GT.K2)GO TO 105
201              I2=I2+1
202              GO TO 75
203         100  ICOL=ICOL+1
204         105  CWORD(ICOL)=ITEMP
205              FLD(6,30,CDWRD(ICOL))=FLD(0,30,CWORD(ICOL))
206              GO TO 25
207        C
208        C                EVENT SPECIFICATION OR DATA CARD
209        C
210         110  IF(ICOL.GT.2)GO TO 115
211              IF(ICOL.EQ.0)PRINT 111
212         111  FORMAT(' CANNOT BEGIN WITH EVENTS - ABOVE LINE IGNORED')
213              IF(ICOL.NE.0)PRINT 114
214         114  FORMAT(' AT LEAST THREE COLUMNS MUST BE CALLED FOR FIRST - ABOVE L
215             .INE IGNORED')
216              GO TO 25
217         115  IF(K1.EQ.6.OR.K1.GT.12.OR.K1.EQ.0.AND.J.GT.11.OR.J.EQ.5.OR.
218             . K2-K1.GT.27.OR.K2.NE.0.AND.J-K2.GT.6.OR.K3.NE.0)GO TO 260
219              IF(IROW.NE.50)GO TO 121
220              PRINT 120
221         120  FORMAT(' TOO MANY EVENTS CALLED FOR - ONLY 50 ALLOWED - ABOVE LINE
222             . IGNORED')
223              GO TO 25
224         121  K=J
225              IF(K1.NE.0)K=K1-1
226              CALL PACK(6,K,ITMP1)
227               IF(ITMP1.EQ.STOP.OR.ITMP1.EQ.END.OR.ITMP1.EQ.LAST)GO TO 125
```

```
      DO 122 I=1,10
122   IF(ITMP1.EQ.TAG(I))GO TO 125
      DO 123 I=1,ICOL
123   IF(ITMP1.EQ.CWORD(I).OR.ITMP1.EQ.CDWRD(I))GO TO 125
      IF(IROW.EQ.0)GO TO 127
      DO 124 I=1,IROW
124   IF(ITMP1.EQ.RWORD(I))GO TO 125
      GO TO 127
125   PRINT 126,ITMP1
126   FORMAT(' EVENT NAME ''',A6,''' IS ILLEGAL - ABOVE LINE IGNORED')
      GO TO 25
127   IF(K2.EQ.0.OR.K2.EQ.J)GO TO 145
      CALL PACK(K2+1,J,ITEMP)
      DO 130 I=1,ICOL
130   IF(ITEMP.EQ.CWORD(I))GO TO 137
134   PRINT 135,ITEMP
135   FORMAT(' ''',A6,''' NOT RECOGNIZED - ABOVE LINE IGNORED')
      GO TO 25
137   IF(I.NE.ICOL)GO TO 140
      PRINT 138,ITEMP
138   FORMAT(' ILLEGAL BURN SPECIFIED BY ''',A6,''' - ABOVE LINE IGNORED
     .')
      GO TO 25
140   IROW=IROW+1
      IBURN(IROW)=I
      FLAG1=.TRUE.
      GO TO 146
145   IROW=IROW+1
      FLAG1=.FALSE.
146   FLAG2=.FALSE.
      FLAG3=.FALSE.
      EVENTS=.TRUE.
      ISUBHD(IROW)=0
      RWORD(IROW)=ITMP1
      IF(K1.GE.K2)GO TO 151
      K=0
      DO 150 I=K1,K2,6
      K=K+1
150   CALL PACK(I,MIND(I+5,K2),RHEAD(K,IROW))
151   IF(IROW.EQ.1)GO TO 25
      IRM1=IROW-1
      M=IBURN(IRM1)
      K=2*IROW-1
      IF(M.NE.0)MPF(IRM1)=VAR(M,K)
      MF(IRM1)=VAR(ICOL,K)
      GO TO 25
C
C                DATA CARD
C
155   IF(K1.EQ.0)GO TO 180
      IF(K1.NE.1.OR.K2.GT.26.OR.K2.NE.J)GO TO 260
      IF(ISUBHD(IROW).LT.3)GO TO 165
      PRINT 160
160   FORMAT(' TOO MANY SUBHEADINGS - ABOVE LINE IGNORED')
      GO TO 25
165   ISUBHD(IROW)=ISUBHD(IROW)+1
      FLAG3=.FALSE.
```

```
      IF(K2.EQ.0)GO TO 25
      ISUB=ISUBHD(IROW)
      K=0
      DO 175 I=K1,K2,6
      K=K+1
175   CALL PACK(I,MIN0(I+5,K2),RSUBHD(K,ISUB,IROW))
      GO TO 25
180   IF(K3.EQ.1.OR.J.LT.K3)GO TO 260
      K=J
      IF(K3.NE.0)K=K3-1
      CALL PACK(1,MIN0(6,K),ITEMP)
      DO 190 I1=1,ICOL
      IF(ITEMP.NE.CWORD(I1))GO TO 185
      I2=-1
      GO TO 210
185   IF(ITEMP.NE.CDWRD(I1))GO TO 190
      I2=0
      GO TO 210
190   CONTINUE
       DO 195 I=1,5
      IF(TAG1(I).NE.ITEMP)GO TO 195
       IF(.NOT.FLAG1)GO TO 205
      I1=ICOL+IC(I)
       I2=-1
      GO TO 210
195   CONTINUE
       GO TO 230
205   PRINT 206
206   FORMAT(' NO BURN THIS EVENT - ABOVE LINE IGNORED')
      GO TO 25
210   I2=2*IROW+I2
      IF(K3.NE.0)GO TO 215
      IVAR(I1,I2)=BLANK
      GO TO 225
215   IF(K4.EQ.0)K4=J
      TEMP=0.
      DO 220 K=K3,J
      NMBR=FLD(0,6,CH(K))-48
      IF(NMBR.LT.0.OR.NMBR.GT.9)GO TO 260
220   TEMP=TEMP+NMBR*10.**(K4-K)
      VAR(I1,I2)=TEMP
      IF(NEGFLG)VAR(I1,I2)=-VAR(I1,I2)
225   IF(2*IROW.LE.I2)FLAG2=.TRUE.
      FLAG3=.FALSE.
      GO TO 25
230    IF(ITEMP.NE.LINEAR.AND.ITEMP.NE.DISCRE)GO TO 235
      IF(K3.NE.0)GO TO 260
      IF(.NOT.FLAG1)GO TO 205
      LINFLG(IROW)=ITEMP.EQ.LINEAR
      FLAG3=.FALSE.
      GO TO 25
235    IF(ITEMP.NE.LAST)GO TO 134
      IF(K3.NE.0)GO TO 260
      IF(IROW.GE.2)GO TO 238
236   PRINT 237
237   FORMAT(' AT LEAST THREE COLUMNS AND TWO EVENTS REQUIRED - ABOVE LI
     .NE IGNORED')
```

```
      GO TO 25
  238 IF(.NOT.FLAG1.AND..NOT.FLAG2)GO TO 270
      IF(FLAG3)GO TO 250
      FLAG3=.TRUE.
      PRINT 240
  240 FORMAT(' BURN AND/OR WEIGHT CHANGES SPECIFIED FOR FINAL EVENT - NO
     .NE ALLOWED'/' ''LAST'' ABOVE IGNORED - ''LAST'' AGAIN IF BURN AND/
     .OR CHANGES TO BE DELETED')
      GO TO 25
  245 IF(ICOL.GE.3.AND.IROW.GE.2)GO TO 247
      PRINT 246
  246 FORMAT(' AT LEAST THREE COLUMNS AND TWO EVENTS REQUIRED - RUN TERM
     .INATED')
      STOP
  247 IF(FLAG1)GO TO 248
      IF(.NOT.FLAG2)GO TO 270
      GO TO 250
  248 IBURN(IROW)=0
      LINFLG(IROW)=.FALSE.
      I2=2*IROW-1
      VAR(ICOL+1,I2)=0.
      VAR(ICOL+2,I2)=0.
  250 PRINT 255
  255 FORMAT(' FINAL BURN AND/OR WEIGHT CHANGES IGNORED')
      GO TO 270
  260 PRINT 265
  265 FORMAT(' ABOVE LINE IN ERROR - IGNORED')
      GO TO 25
C
C                 WRITE PROGRAM
C
  270   ICM1=ICOL-1
        ICP1=ICOL+1
        ICP2=ICOL+2
        IRM1=IROW-1
        IRX2=IROW*2-1
        WRITE(8,300)
  300   FORMAT(7X,'COMMON /DIM/ ICM1,ICOL,ICP1,ICP2,IRM1,IROW,IRX2'/
       . 7X,'COMMON /PRT/ C,DELTA,DENOM,HDG,HEAD,ICTR,IHDG,ISENS,PDELTA,'/
       . 5X,'. PHDG,SENS,SNSCOL,SNSROW')
        WRITE(8,305)
  305   FORMAT('C')
        WRITE(8,310)ICM1,ICOL,ICP1,ICP2,IRM1,IROW,IRX2
  310   FORMAT(7X,'DATA',9X,'ICM1/',I2,'/,    ICOL/',I2,'/,    ICP1/',
       . I2,'/,    ICP2/',I2,'/',/5X,'.,IRM1/',I2,'/,    IROW/',I2,
       . '/,    IRX2/',I2,'/')
        WRITE(8,305)
        WRITE(8,315)IROW,IBURN(1),(COMMA,BLANK,IBURN(I),I=2,IROW),SLASH
  315   FORMAT(7X,'DIMENSION IBURN(',I2,')/ ',14(I1,2A1)/
       . (5X,'. ',21(I1,2A1)))
        WRITE(8,305)
        WRITE(8,320)ICOL,IROW,ICP2,IRX2,ICP2,IRX2,ICP2,IRX2,ICP2,IRX2,
       . ICP2,IRX2
  320   FORMAT(7X,'DIMENSION',13X,'A(',I2,'),',16X,'IBURNS(',I2,')'/
       . 5X,'.,IVAR(',I2,',',I2,'),',10X,'VAR(',I2,',',I2,'),',11X,
       . 'VARFNL(',I2,',',I2,')'/5X,'.,VARSAV(',I2,',',I2,'),',8X,
       . 'VRINIT(',I2,',',I2,')')
```

```
      WRITE(8,305)
      WRITE(8,325)ICOL,IROW,IROW,IROW,ICOL,IRX2,IROW
325   FORMAT(7X,'INTEGER',15X,'CTAGSV(',I2,'),',11X,'IRKC(',I2,')'/
     . 5X,'.,IROCK(',I2,'),',12X,'JCOL(',I2,',',I2,'),',10X,'NROW(',
     . I2,')'/5X,'.,RTAGSV(',I2,')')
      WRITE(8,305)
      WRITE(8,330)ICOL,ICOL,IROW,IROW,IROW,IROW,IROW,ICOL,IRM1,IRX2
330   FORMAT(7X,'LOGICAL',15X,'CPRT(',I2,')/',I2,'*1/'
     .       /5X,'.,LINFLG(',I2,'),',11X,'LNFLGS(',I2,'),',11X,
     . 'RDONE(',I2,') '/5X,'.,RPRT(',I2,')/',I2,'*1/,',7X,
     . 'SDONE(',I2,',',I2,'),',9X,'TDONE(',I2,')')
      WRITE(8,335)IROW,LINFLG(1),(COMMA,BLANK,LINFLG(I),I=2,IROW),SLASH
335   FORMAT(7X,'LOGICAL LFINIT(',I2,')/ ',15(O1,2A1)/
     . (5X,'. ',21(O1,2A1)))
      WRITE(8,305)
      WRITE(8,340)
340   FORMAT(7X,'EQUIVALENCE (VAR,IVAR)')
      WRITE(8,305)
      WRITE(8,345)IROW
345   FORMAT(7X,'INTEGER RTAG(',I2,')')
      CALL DATA(RWORD,IROW)
      WRITE(8,350) ICOL
350   FORMAT(7X,'INTEGER CTAG(',I2,')')
      CALL DATA(CWORD,ICOL)
      WRITE(8,355)ICOL
355   FORMAT(7X,'INTEGER DTAG(',I2,')')
      CALL DATA(CDWRD,ICOL)
      WRITE(8,305)
      WRITE(8,360)IROW,IROW
360   FORMAT(7X,'DIMENSION RHEAD(5,',I2,'),RSUBHD(5,3,',I2,')')
      DO 375 J=1,IROW
      WRITE(8,365)J
365   FORMAT(7X,'DATA (RHEAD(I,',I2,'),I=1,5)')
      CALL DATA(RHEAD(1,J),5)
      WRITE(8,370)J
370   FORMAT(7X,'DATA ((RSUBHD(I,J,',I2,'),I=1,5),J=1,3)')
375   CALL DATA(RSUBHD(1,1,J),15)
      WRITE(8,305)
      WRITE(8,380)ICOL
380   FORMAT(7X,'DIMENSION CHEAD(2,3,',I2,')')
      CALL DATA(CHEAD,6*ICOL)
      WRITE(8,305)
      DO 395 J=1,IRX2
      WRITE(8,385)J,ICP2
385   FORMAT(7X,'DATA(VRINIT(I,',I2,'),I=1,',I2,')')
      DO 390 I=1,ICP2
390   V(I)=VAR(I,J)
395   CALL DATA(V,ICP2)
      WRITE(8,305)
      WRITE(8,400)
400   FORMAT(7X,'CALL BODY(A,CHEAD,CPRT,CTAG,CTAGSV,DTAG,IBURN,IBURNS,'
     . 'IRKC,IROCK'/5X,'.,IVAR,JCOL,LFINIT,LINFLG,LNFLGS,NROW,RDONE,'
     . 'RHEAD,RPRT,RSUBHD'/5X,'.,RTAG,RTAGSV,SDONE,TDONE,VAR,VARFNL,'
     . 'VARSAV,VRINIT)')
      WRITE(8,305)
      WRITE(8,410)
410   FORMAT(7X,'END')
```

```
      PRINT 415
  415 FORMAT(' END OF PRECOMPILATION')
      STOP
C
      SUBROUTINE PACK(I1,I2,IWORD)
      IWORD=BLANK
      IBIT=0
      DO 15 ICH=I1,I2
      FLD(IBIT,6,IWORD)=FLD(0,6,CH(ICH))
   15 IBIT=IBIT+6
      RETURN
C
      SUBROUTINE DATA(ARRAY,NUMBER)
      DIMENSION ARRAY(2)
      WRITE(6,5)ARRAY(1),(COMMA,BLANK,ARRAY(I),I=2,NUMBER),SLASH
    5 FORMAT(5X,'./ 0',O12,3(2A1,'0',O12),2A1/
     . (5X,'.  ',4('0',O12,2A1)))
      RETURN
C
      END
```

```
      COMMON /DIM/ ICM1,ICOL,ICP1,ICP2,IRM1,IROW,IRX2
      COMMON /PRT/ C,DELTA,DENOM,HDG,HEAD,ICTR,IHDG,ISENS,PDELTA,
     . PHDG,SENS,SNSCOL,SNSROW
C
      DATA          ICM1/ 4/,    ICOL/ 5/,    ICP1/ 6/,    ICP2/ 7/
     .,IRM1/ 3/,    IROW/ 4/,    IRX2/ 7/
C
      DIMENSION IBURN( 4)/ 4, 0, 2, 0/
C
      DIMENSION                A( 5),                    IBURNS( 4)
     .,IVAR( 7, 7),            VAR( 7, 7),               VARFNL( 7, 7)
     .,VARSAV( 7, 7),          VRINIT( 7, 7)
C
      INTEGER                  CTAGSV( 5),               IRKC( 4)
     .,IROCK( 4),              JCOL( 4, 5),              NROW( 7)
     .,RTAGSV( 4)
C
      LOGICAL                  CPRT( 5)/ 5*1/
     .,LINFLG( 4),             LNFLGS( 4),               RDONE( 4)
     .,RPRT( 4)/ 4*1/,         SDONE( 5, 3),             TDONE( 7)
      LOGICAL LFINIT( 4)/ 1, 0, 0, 0/
C
      EQUIVALENCE (VAR,IVAR)
C
      INTEGER RTAG( 4)
     ./ 0113010233105, 0311015113423, 0063010233105, 0112410200505/
      INTEGER CTAG( 5)
     ./ 0063010050505, 0062530050505, 0111230100505, 0112530050505,
     .  0212205050505/
      INTEGER DTAG( 5)
     ./ 0110630100505, 0110625300505, 0111112301005, 0111125300505,
     .  0112122050505/
C
      DIMENSION RHEAD(5, 4),RSUBHD(5,3, 4)
      DATA (RHEAD(I, 1),I=1,5)
     ./ 0071214162305, 0111230101223, 0310505050505, 0050505050505,
     .  0050505050505/
      DATA ((RSUBHD(I,J, 1),I=1,5),J=1,3)
     ./ 0102423303222, 0060721123005, 0050505050505, 0050505050505,
     .  0050505050505, 0062311052527, 0242512212106, 0233105050505,
     .  0050505050505, 0050505050505, 0050505050505, 0050505050505,
     .  0050505050505, 0050505050505, 0050505050505/
      DATA (RHEAD(I, 2),I=1,5)
     ./ 0312432101511, 0243423052423, 0053032271306, 0101205050505,
     .  0050505050505/
      DATA ((RSUBHD(I,J, 2),I=1,5),J=1,3)
     ./ 0171231311630, 0242312110534, 0121614153105, 0050505050505,
     .  0050505050505, 0050505050505, 0050505050505, 0050505050505,
     .  0050505050505, 0050505050505, 0050505050505, 0050505050505,
     .  0050505050505, 0050505050505, 0050505050505/
      DATA (RHEAD(I, 3),I=1,5)
     ./ 0071214162305, 0252434122712, 0110506301012, 0233105050505,
     .  0050505050505/
      DATA ((RSUBHD(I,J, 3),I=1,5),J=1,3)
     ./ 0102423303222, 0060721123005, 0050505050505, 0050505050505,
     .  0050505050505, 0062311052527, 0242512212106, 0233105050505,
```

```
     .  0050505050505, 0050505050505, 0050505050505, 0050505050505,
     .  0050505050505, 0050505050505, 0050505050505/
      DATA (RHEAD(I, 4),I=1,5)
     ./ 0112410200534, 0163115051030, 0220505050505, 0050505050505,
     .  0050505050505/
      DATA ((RSUBHD(I,J, 4),I=1,5),J=1,3)
     ./ 0050505050505, 0050505050505, 0050505050505, 0050505050505,
     .  0050505050505, 0050505050505, 0050505050505, 0050505050505,
     .  0050505050505, 0050505050505, 0050505050505, 0050505050505,
     .  0050505050505, 0050505050505, 0050505050505/

      DIMENSION CHEAD(2,3, 5)
     ./ 0063010122331, 0050505050505, 0303106141205, 0050505050505,
     .  0341216141531, 0050505050505, 0063010122331, 0050505050505,
     .  0252724257505, 0050505050505, 0341216141531, 0050505050505,
     .  0111230101223, 0310505050505, 0303106141205, 0050505050505,
     .  0341216141531, 0050505050505, 0111230101223, 0310505050505,
     .  0252724257505, 0050505050505, 0341216141531, 0050505050505,
     .  0312431062105, 0050505050505, 0212205050505, 0050505050505,
     .  0341216141531, 0050505050505/

      DATA(VRINIT(I, 1),I=1, 7)
     ./ 0215547543146, 0215505363146, 0215576306314, 0217460643777,
     .  0050505050505, 0215666274631, 0211456231463/
      DATA(VRINIT(I, 2),I=1, 7)
     ./ 0570173777777, 0000000000000, 0572034631463, 0050505050505,
     .  0050505050505, 0000000000000, 0000000000000/
      DATA(VRINIT(I, 3),I=1, 7)
     ./ 0050505050505, 0050505050505, 0050505050505, 0050505050505,
     .  0050505050505, 0000000000000, 0000000000000/
      DATA(VRINIT(I, 4),I=1, 7)
     ./ 0572303146314, 0000000000000, 0050505050505, 0050505050505,
     .  0050505050505, 0000000000000, 0000000000000/
      DATA(VRINIT(I, 5),I=1, 7)
     ./ 0050505050505, 0050505050505, 0000000000000, 0000000000000,
     .  0050505050505, 0215573771463, 0211465314631/
      DATA(VRINIT(I, 6),I=1, 7)
     ./ 0570161146314, 0050505050505, 0000000000000, 0000000000000,
     .  0050505050505, 0000000000000, 0000000000000/
      DATA(VRINIT(I, 7),I=1, 7)
     ./ 0050505050505, 0050505050505, 0000000000000, 0000000000000,
     .  0050505050505, 0000000000000, 0000000000000/

      CALL BODY(A,CHEAD,CPRT,CTAG,CTAGSV,DTAG,IBURN,IBURNS,IRKC,IROCK
     .,IVAR,JCOL,LFINIT,LINFLG,LNFLGS,NROW,RDONE,RHEAD,RPRT,RSUBHD
     .,RTAG,RTAGSV,SDONE,TDONE,VAR,VARENL,VARSAV,VRINIT)

      END

END FIN

RUNID:  JLMWAP   ACCOUNT: JLM              PROJECT: WEIGHT
```

```
      SUBROUTINE BODY(A,CHEAD,CPRT,CTAG,CTAGSV,DTAG,IBURN,IBURNS,IRKC
     .,IROCK,IVAR,JCOL,LFINIT,LINFLG,LNFLGS,NROW,RDONE,RHEAD,RPRT,RSUBHD
     .,RTAG,RTAGSV,SDONE,TDONE,VAR,VARFNL,VARSAV,VRINIT)
C
      REAL MC
C
      COMMON /DIM/ ICM1,ICOL,ICP1,ICP2,IRM1,IROW,IRX2
      COMMON /PRT/ C,DELTA,DENOM,HDG,HEAD,ICTR,IHDG,ISENS,PDELTA,PHDG
     . ,SENS,CSENS,RSENS
C
      LOGICAL                  CPRT(ICOL),            ITDUN
     .,LFINIT(IROW),           LINFLG(IROW),          LNFLGS(IROW)
     .,RDONE(IROW),            RPRT(IROW),            SDONE(ICOL,IRM1)
     .,TDONE(IRX2)
C
      DIMENSION                A      (ICOL),         CHEAD (2,3,ICOL)
     .,F      (12),           FORM   (144),          HEAD  (5,80)
     .,RHEAD (5,IROW),        RSUBHD(5,3,IROW),      V     (11)
     .,VAR   (ICP2,IRX2),     VARFNL(ICP2,IRX2),     VARSAV(ICP2,IRX2)
     .,VRINIT(ICP2,IRX2),     WORD   (40)
C
      DATA                     BLANK /'       '/,     HEAD  /400*' '/
     .,UNKNWN/'UNKNWN'/
C
      INTEGER                  ALL    /'ALL    '/,     CENTER/'CENTER'/
     .,COL    /'COLUMN'/,     COLUMN(80),            COMMA /',      '/
     .,CSENS,                 CSTAT (11),            CTAG  (ICOL)
     .,CTAGSV(ICOL),          DATA   /'DATA   '/,    DELETE/'DELETE'/
     .,DELTA /'DELTA '/,      DISCRE/'DISCRE'/,      DTAG  (ICOL)
     .,E,                     END    /'END    '/,    ERRFLG/'NOGOOF'/
     .,ERRTAG,                EVENTS/'EVENTS'/,      GOOF  /'GOOF   '/
     .,HDG    /'HDG    '/,     IBURN (IROW),          IBURNS(IROW)
     .,ICSENS/1        /,     ICSTEP/1        /,     ICTABL(10)
     .,ICTR   /'CENTER'/,     IHDG   /1       /,     IOP   (40)
     .,IRSENS/1        /,     IRSTEP/1        /,     IRTABL(10)
     .,ISENS /'NO     '/,     ISTAT /'NO     '/,     ISTEP /'NO     '/
     .,IT     /'       '/,     ITAG1 /'       '/,     ITEL  /'ITERAT'/
     .,ITFLG /'NO     '/,     ITERATE/'YES    '/,    ITYPE (40)
     .,IVAR  (ICP2,IRX2),     JWORD (40),            LABELS/'LABELS'/
     .,LAST   /'LAST   '/,    LINEAR/'LINEAR'/,      MINUS /'-      '/
     .,NCOL   /1       /,     NO     /'NO     '/,    NODATA/'NODATA'/
     .,NODELT/'NODELT'/,      NOGOOF/'NOGOOF'/,      NOHDG /'NOHDG  '/
     .,NOIT   /'NOITER'/,     NOLABE/'NOLABE'/,      NOTABL/'NOTABL'/
     .,NSTEPS/5        /,     NUMBER/'NUMBER'/,      OLD   /'OLD    '/
     .,PART   /'PART   '/,    PC     /'%      '/,    PDATA /'DATA   '/
     .,PDELTA/'DELTA '/,      PERCEN/'PERCEN'/,      PGOOF /'NOGOOF'/
     .,PHDG   /'NOHDG '/,     PITFLG/'NOITFL'/,      PLABEL/'NOLABE'/
     .,PLUS   /'+     '/,     PRINT /'PRINT  '/,     PTABLE/'NOTABL'/
     .,RESTAR/'RESTAR'/,      RSENS,                 RSTAT (11)
     .,RTAG   (IROW),         RTAGSV(IROW),          SENS  /'SENS   '/
     .,SLASH /'/      '/,     STAR   /'*      '/,    STATUS/'STATUS'/
     .,STEP   /'STEP   '/,    STOP   /'STOP   '/,    TABLE /'TABLE  '/
     .,VARBL /'VARBLE'/,      YES    /'YES    '/
C
      INTEGER                  IRKC(IROW),            IROCK(IROW)
     .,JCOL(IROW,ICOL),        NROW(IRX2)
```

```
C
      INTEGER                 CTABLE(10)/10*' '/,   IC(5)/1,1,1,2,2/
     .,RTABLE(10)/10*' '/,    TAG1(5)/'DELTAVDV    V     ISP   I     '/
C
      INTEGER      AX/'A'/,     C/'C'/,      D/'D'/,      DL/'DL'/
     .,EV/'E'/,    G/'G'/,      H/'H'/,      IX/'I'/,     LST/'L'/
     .,ND/'ND'/,   NDL/'NDL'/,  NG/'NG'/,    NH/'NH'/,    NI/'NI'/
     .,NL/'NL'/,   NT/'NT'/,    P/'P'/,      R/'R'/,      S/'S'/
     .,ST/'ST'/,   T/'T'/,      Y/'Y'/
C
      EQUIVALENCE             (BLANK,IBLANK),        (JWORD(1),NAME)
     .,(VAR,IVAR),            (WORD,JWORD),          (WORD(2),VALUE)
C
C               ERROR MESSAGE FORMATS
C
      DATA FORM/'(" ABOVE LINE IN ERROR---IGNORED")',6*' '
     . ,'(" PRINT COMMAND IN ERROR---IGNORED")',5*' '
     . ,'(" VARIABLE IS AN UNKNOWN---ABOVE LINE IGNORED")',4*' '
     . ,'(" NO EVENT SPECIFIED YET---ABOVE LINE IGNORED")',4*' '
     . ,'(" PERTURBING VARIABLE IS AN UNKNOWN---""LAST"" COMMAND IG
     .NORED")',' '
     . ,'(" CHANGE TO FINAL EVENT---ABOVE LINE IGNORED")',4*' '
     . ,'(" SENS OR DELTA COMMAND IN ERROR---IGNORED")',4*' '
     . ,'(" NO BURN THIS EVENT---ABOVE LINE IGNORED")',4*' '
     . ,'(" PRINT COMMAND WITH TOO FEW WORDS---IGNORED")',4*' '
     . ,'(" PRINT COMMAND IN ERROR---INVALID PART IGNORED")',3*' '
     . ,'(" STEP COMMAND IN ERROR---IGNORED")',6*' '
     . ,'(" DELTA FOR SENSITIVITY TOO SMALL")',6*' '
     . /
C
C               BEGIN EXECUTION
C
  100 DO 110 I=1,ICP2
      DO 110 J=1,IRX2
      VARSAV(I,J)=VRINIT(I,J)
  110 VAR(I,J)=VRINIT(I,J)
      DO 120 I=1,IROW
      RTAGSV(I)=RTAG(I)
      IBURNS(I)=IBURN(I)
      LNFLGS(I)=LFINIT(I)
  120 LINFLG(I)=LFINIT(I)
      DO 125 I=1,ICOL
  125 CTAGSV(I)=CTAG(I)
      GO TO 190
  130 DO 140 I=1,ICP2
      DO 140 J=1,IRX2
  140 VAR(I,J)=VARSAV(I,J)
      IT=IBLANK
      IF(ISTEP.NE.STEP.OR.KSTEP.EQ.NSTEPS)GO TO 150
      VAR(ICSTEP,IRSTEP)=VARSAV(ICSTEP,IRSTEP)*(1.+STFCTR*FLOAT(KSTEP))
     . +FLOAT(KSTEP)*STDELT
  150 IF(ISENS.NE.SENS.AND.ISENS.NE.DELTA)GO TO 160
      IF(ISTEP.EQ.STEP.AND.KSTEP.EQ.NSTEPS)GO TO 160
      IF(KSENS.EQ.0)GO TO 1630
      IF(KSENS.GT.1)GO TO 160
      VAR(ICSENS,IRSENS)=(1.+FCTR)*VAR(ICSENS,IRSENS)+DELT
      GO TO 1630
```

```
  160   DO 170 I=1,IROW
        RTAG(I)=RTAGSV(I)
        IBURN(I)=IBURNS(I)
  170   LINFLG(I)=LNFLGS(I)
        DO 180 I=1,ICOL
  180   CTAG(I)=CTAGSV(I)
  190   K=0
        KSENS=0
        IF(ISTEP.EQ.STEP.AND.KSTEP.LT.NSTEPS)GO TO 1630
C
C               READ A NEW CARD
C
  200   CALL READ(COLUMN,WORD,IOP,ITYPE,PDATA,IWORD,$1150)
        IF(NAME.EQ.D)NAME=DELTA
        IF(NAME.EQ.H)NAME=HDG
        IF(NAME.EQ.LST)NAME=LAST
        IF(NAME.EQ.P)NAME=PRINT
        IF(NAME.EQ.S)NAME=SENS
        IF(NAME.EQ.ST)NAME=STEP
        IF(NAME.NE.PRINT)GO TO 700
C
C               PRINT COMMAND
C
        JW=JWORD(2)
        E=9
        IF(IWORD.EQ.1)GO TO 2500
        IF(JW.EQ.C)JW=COL
        IF(JW.EQ.D)JW=DATA
        IF(JW.EQ.DL)JW=DELTA
        IF(JW.EQ.EV)JW=EVENTS
        IF(JW.EQ.G)JW=GOOF
        IF(JW.EQ.H)JW=HDG
        IF(JW.EQ.IX)JW=ITFL
        IF(JW.EQ.LST)JW=LABELS
        IF(JW.EQ.ND)JW=NODATA
        IF(JW.EQ.NDL)JW=NODELT
        IF(JW.EQ.NG)JW=NOGOOF
        IF(JW.EQ.NH)JW=NOHDG
        IF(JW.EQ.NI)JW=NOIT
        IF(JW.EQ.NL)JW=NOLABE
        IF(JW.EQ.NT)JW=NOTABL
        IF(JW.EQ.S)JW=STATUS
        IF(JW.EQ.T)JW=TABLE
        IF(JW.NE.DATA.AND.JW.NE.NODATA)GO TO 300
        PDATA=JW
        GO TO 200
  300   IF(JW.NE.GOOF.AND.JW.NE.NOGOOF)GO TO 310
        PGOOF=JW
        GO TO 200
  310   IF(JW.NE.DELTA.AND.JW.NE.NODELT)GO TO 320
        PDELTA=JW
        GO TO 200
  320   IF(JW.NE.HDG.AND.JW.NE.NOHDG)GO TO 330
        PHDG=JW
        GO TO 200
  330   IF(JW.NE.LABELS.AND.JW.NE.NOLABE)GO TO 340
        PLABEL=JW
```

```
171                 GO TO 200
172          340    IF(JW.NE.ITFL.AND.JW.NE.NOIT)GO TO 400
173                 PITFLG=JW
174                 GO TO 200
175         C
176          400    IF(JW.NE.TABLE.AND.JW.NE.NOTABL)GO TO 500
177         C
178         C                 PRINT COMMAND FOR TABLE
179         C
180                 IF(IWORD.EQ.2)PTABLE=JW
181                 IF(IWORD.EQ.2)GO TO 200
182                 E=10
183                 IW=3
184          405    IF(IWORD-IW)200,2500
185                 IF(ITYPE(IW).NE.NUMBER)GO TO 2500
186                 KW=WORD(IW)+.5
187                 IF(KW.LT.1.OR.KW.GT.10)GO TO 2500
188                 DO 410 J=1,4
189                 I=IW+J
190          410    IF(IOP(I).EQ.COMMA)GO TO 415
191          415    GO TO (2500,425,430,420),J
192          420    IF(IWORD-IW-2) 425,430,2500
193          425    IF(JWORD(IW+1).NE.DELETE.AND.JWORD(IW+1).NE.D)GO TO 2500
194                 CTABLE(KW)=IBLANK
195                 RTABLE(KW)=IBLANK
196                 IW=IW+2
197                 GO TO 455
198          430    CALL VARBLE(JWORD(IW+2),ITAG,I,$2500)
199                 IF(ITAG.NE.1)GO TO 2500
200                 CALL VARBLE(JWORD(IW+1),ITAG,J,$2500)
201                 GO TO (2500,440,435,445),ITAG
202          435    IF(I.EQ.IROW)GO TO 2500
203                 KRD=1
204          440    ICTABL(KW)=J
205                 CTABLE(KW)=CTAG(J)
206                 IF(KRD.EQ.1)CTABLE(KW)=DTAG(J)
207                 GO TO 450
208          445    IF(IBURN(I).EQ.0.OR.J.GT.5)GO TO 2500
209                 ICTABL(KW)=ICOL+IC(J)
210                 CTABLE(KW)=TAG1(J)
211          450    IRTABL(KW)=2*I-1+KRD
212                 RTABLE(KW)=RTAG(I)
213                 KRD=0
214                 IW=IW+3
215          455    DO 460 I=10,1,-1
216          460    IF(CTABLE(I).NE.IBLANK)GO TO 465
217          465    NCOL=I
218                 GO TO 405
219         C
220          500    IF(JW.NE.STATUS)GO TO 600
221         C
222         C                 PRINT COMMAND FOR STATUS
223         C
224                 E=2
225                 IW=3
226                 NW=1
227                 IF(IWORD-3)2500,505,510
```

```
 505   IF(JWORD(3).EQ.AX)JWORD(3)=ALL
       IF(JWORD(3).EQ.P)JWORD(3)=PART
       IF(JWORD(3).NE.ALL.AND.JWORD(3).NE.PART)GO TO 2500
       ISTAT=JWORD(3)
       KSENS=2
       KSTEP=NSTEPS
       GO TO 1600
 510   IF(NW.EQ.11)GO TO 2500
       IF(IWORD-IW)555,2500
       DO 515 J=1,3
       I=IW+J
 515   IF(IOP(I).EQ.COMMA)GO TO 520
 520   GO TO (2500,530,525),J
 525   IF(IWORD-IW.NE.1)GO TO 2500
 530   CALL VARBLE(JWORD(IW+1),ITAG,I,$2500)
       IF(ITAG.NE.1)GO TO 2500
       CALL VARBLE(JWORD(IW),ITAG,J,$2500)
       GO TO (2500,540,535,545),ITAG
 535   IF(I.EQ.IROW)GO TO 2500
       KRD=1
 540   ICSTAT=J
       CSTAT(NW)=CTAG(J)
       IF(KRD.EQ.1)CSTAT(NW)=DTAG(J)
       GO TO 550
 545   IF(IBURN(I).EQ.0.OR.J.GT.5)GO TO 2500
       ICSTAT=ICOL+IC(J)
       CSTAT(NW)=TAG1(J)
 550   IRSTAT=2*I-1+KRD
       RSTAT(NW)=RTAG(I)
       V(NW)=VAR(ICSTAT,IRSTAT)
       KRD=0
       IW=IW+2
       NW=NW+1
       GO TO 510
 555   NW=NW-1
       DO 585 I=1,NW
       IF(BOOL(V(I)).EQ.BLANK)GO TO 560
       IF(ABS(V(I)).GE..05)GO TO 575
       V(I)=BLANK
       GO TO 565
 560   V(I)=UNKNWN
 565   PRINT 570,CSTAT(I),RSTAT(I),V(I)
 570   FORMAT(1X,A6,'(',A6,')='A6)
       GO TO 585
 575   PRINT 580,CSTAT(I),RSTAT(I),V(I)
 580   FORMAT(1X,A6,'(',A6,')='F9.1)
 585   CONTINUE
       GO TO 200
C
C              PRINT COMMAND WITH CHANGES TO PRINT MATRIX
C
 600   IF(JW.NE.COL)GO TO 640
       E=9
       IF(IWORD.LT.3)GO TO 2500
       ITST=0
       DO 620 J=3,IWORD
       IF(JWORD(J).EQ.AX)JWORD(J)=ALL
```

```
285            IF(JWORD(J).EQ.ALL)ITST=ITST+1
286            DO 620 I=1,ICOL
287            IF(JWORD(J).EQ.ALL.AND.IOP(J).NE.MINUS)CPRT(I)=.TRUE.
288            IF(JWORD(J).EQ.ALL.AND.IOP(J).EQ.MINUS)CPRT(I)=.FALSE.
289            IF(JWORD(J).NE.CTAG(I))GO TO 620
290            ITST=ITST+1
291            CPRT(I)=.TRUE.
292            IF(IOP(J).EQ.MINUS)CPRT(I)=.FALSE.
293      620   CONTINUE
294            IF(ITST.EQ.(IWORD-2))GO TO 200
295            E=10
296            GO TO 2500
297      640   E=2
298            IF(JW.NE.EVENTS)GO TO 2500
299            E=9
300            IF(IWORD.LT.3)GO TO 2500
301            ITST=0
302            DO 660 J=3,IWORD
303            IF(JWORD(J).EQ.AX)JWORD(J)=ALL
304            IF(JWORD(J).EQ.ALL)ITST=ITST+1
305            DO 660 I=1,IROW
306            IF(JWORD(J).EQ.ALL.AND.IOP(J).NE.MINUS)RPRT(I)=.TRUE.
307            IF(JWORD(J).EQ.ALL.AND.IOP(J).EQ.MINUS)RPRT(I)=.FALSE.
308            IF(JWORD(J).NE.RTAG(I))GO TO 660
309            ITST=ITST+1
310            RPRT(I)=.TRUE.
311            IF(IOP(J).EQ.MINUS)RPRT(I)=.FALSE.
312      660   CONTINUE
313            IF(ITST.EQ.(IWORD-2))GO TO 200
314            E=10
315            GO TO 2500
316      C
317       700  IF(NAME.NE.SENS.AND.NAME.NE.DELTA)GO TO 800
318      C
319      C              SENS OR DELTA COMMAND
320      C
321            IF(IWORD.EQ.1.OR.IWORD.GT.5)GO TO 720
322            IF(IWORD.NE.2)GO TO 710
323            ISENS=NO
324            IF(JWORD(2).EQ.YES.OR.JWORD(2).EQ.Y)ISENS=JWORD(1)
325            GO TO 200
326       710  CALL VARBLE(JWORD(3),ITAG,J,$720)
327            IF(ITAG.NE.1)GO TO 720
328            GO TO 730
329       720  E=7
330            GO TO 2500
331       730  CALL VARBLE(JWORD(2),ITAG,I,$720)
332            GO TO (720,745,740,750),ITAG
333       740  IF(J.EQ.IROW)GO TO 720
334            KRD=1
335       745  ISNC=I
336            GO TO 760
337       750  E=1
338            IF(I.GT.5)GO TO 2500
339            E=8
340            IF(IBURN(J).EQ.0)GO TO 2500
341            ISNC=ICOL+IC(I)
```

```
      ITAG1=TAG1(I)
  760 IF(IWORD.GT.3)GO TO 770
      FCTR=0.01
      DELT=0.
      GO TO 790
  770 IF(ITYPE(4).NE.NUMBER)GO TO 720
      IF(IWORD.EQ.5)GO TO 780
      FCTR=0.
      DELT=WORD(4)
      GO TO 790
  780 IF(JWORD(5).NE.PERCEN.AND.JWORD(5).NE.PC)GO TO 720
      FCTR=0.01*WORD(4)
      DELT=0.
  790 IF(IWORD.GT.3.AND.IOP(4).EQ.MINUS)FCTR=-FCTR
      IF(IWORD.GT.3.AND.IOP(4).EQ.MINUS)DELT=-DELT
      RSENS=RTAG(J)
      CSENS=CTAG(ISNC)
      IF(KRD.EQ.1)CSENS=DTAG(ISNC)
      IF(ITAG1.NE.IBLANK)CSENS=ITAG1
      IF(CSENS.EQ.TAG1(2).OR.CSENS.EQ.TAG1(3))CSENS=TAG1(1)
      IF(CSENS.EQ.TAG1(5))CSENS=TAG1(4)
      ICSENS=ISNC
      IRSENS=J*2+KRD-1
      ITAG1=IBLANK
      KRD=0
      GO TO 200
C
  800 IF(NAME.NE.STEP)GO TO 900
C
C                 STEP COMMAND
C
      IF(IWORD.EQ.1.OR.IWORD.GT.6)GO TO 820
      IF(IWORD.NE.2)GO TO 810
      ISTEP=NO
      IF(JWORD(2).EQ.YES.OR.JWORD(2).EQ.Y)ISTEP=STEP
      GO TO 200
  810 CALL VARBLE(JWORD(3),ITAG,J,$820)
      IF(ITAG.NE.1)GO TO 820
      GO TO 830
  820 E=11
      GO TO 2500
  830 CALL VARBLE(JWORD(2),ITAG,I,$820)
      GO TO (820,850,840,860),ITAG
  840 IF(J.EQ.IROW)GO TO 820
      KRD=1
  850 ISTC=I
      GO TO 870
  860 E=1
      IF(I.GT.5)GO TO 2500
      E=8
      IF(IBURN(J).EQ.0)GO TO 2500
      ISTC=ICOL+IC(I)
  870 IF(IWORD.GT.3.AND.ITYPE(4).NE.NUMBER)GO TO 820
      IF(IWORD.GT.4.AND.ITYPE(5).NE.NUMBER)GO TO 820
      IF(IWORD.EQ.6.AND.(JWORD(6).NE.PERCEN.AND.JWORD(6).NE.PC))
     . GO TO 820
      STFCTR=0.01
```

```
      STDELT=0.
      NSTEPS=5
      IF(IWORD.EQ.5)STFCTR=0.
      IF(IWORD.EQ.6)STFCTR=0.01*WORD(5)
      IF(IWORD.EQ.5)STDELT=WORD(5)
      IF(IWORD.NE.3)NSTEPS=WORD(4)+0.5
      IF(IWORD.EQ.5.AND.IOP(5).EQ.MINUS)STDELT=-STDELT
      IF(IWORD.EQ.6.AND.IOP(5).EQ.MINUS)STFCTR=-STFCTR
      ICSTEP=ISTC
      IRSTEP=J*2+KRD-1
      KRD=0
      GO TO 200
C
 900  IF(NAME.NE.HDG)GO TO 1000
C
C                 READ HEADING
C
      IF(IWORD.EQ.1)GO TO 910
      E=1
      IF(IWORD.GT.2)GO TO 2500
      IF(JWORD(2).EQ.C)JWORD(2)=CENTER
      IF(ITYPE(2).EQ.NUMBER)GO TO 905
      IF(ITYPE(2).EQ.VARBL )ICTR=JWORD(2)
      GO TO 200
 905  IHDGSV=IHDG
      IHDG=WORD(2)+.5
      IF(IHDG.GE.1.OR.IHDG.LE.5)GO TO 200
      IHDG=IHDGSV
      GO TO 2500
 910  DO 920 NNN=1,IHDG
 920  READ 930,(HEAD(NNN,NN),NN=1,80)
 930  FORMAT(80A1)
      IF(PDATA.NE.DATA)GO TO 200
      DO 940 NNN=1,IHDG
 940  PRINT 950,(HEAD(NNN,NN),NN=1,80)
 950  FORMAT(' ',80A1)
      GO TO 200
C
 1000 IF(NAME.NE.LAST)GO TO 1100
      IF(IWORD.GT.1)GO TO 2500
      GO TO 1500
C
 1100 IF(NAME.NE.STOP.AND.NAME.NE.END.AND.NAME.NE.EV)GO TO 1200
      IF(IWORD.GT.1)GO TO 2500
      PRINT 1110
 1110 FORMAT(' STOP OR END COMMAND---RUN TERMINATED')
      STOP
 1150 PRINT 1160
 1160 FORMAT(' END OF FILE ENCOUNTERED--RUN TERMINATED')
      STOP
C
 1200 IF(NAME.NE.RESTAR.AND.NAME.NE.R)GO TO 1300
      IF(IWORD.GT.1)GO TO 2500
      GO TO 100
C
 1300 IF(NAME.NE.LINEAR.AND.NAME.NE.DISCRE)GO TO 1350
C
```

```
456      C                LINEAR OR NONLINEAR COMMAND
457      C
458              E=4
459              IF(K.EQ.0)GO TO 2500
460              E=8
461              IF(IBURN(K).EQ.0)GO TO 2500
462              LINFLG(K)=NAME.EQ.LINEAR
463              GO TO 200
464      C
465       1350   IF(NAME.NE.ITFL.AND.NAME.NE.IX)GO TO 1400
466      C
467      C                ITERATE COMMAND
468      C
469              IF(JWORD(2).EQ.Y)JWORD(2)=YES
470              ITRATE=JWORD(2)
471              GO TO 200
472      C
473      C                DETERMINE VARIABLE NAME
474      C
475       1400   CALL VARBLE(NAME,ITAG,I,$1410)
476              GO TO (1420,1425,1430,1435),ITAG
477       1410   PRINT 1415,NAME
478       1415   FORMAT(' ''',A6,''' NOT RECOGNIZED')
479              GO TO 200
480       1420   E=1
481              IF(IWORD.NE.1)GO TO 2500
482              K=I
483              GO TO 200
484       1425   I1=I
485              I2=2*K-1
486              GO TO 1440
487       1430   E=6
488              IF(K.EQ.IROW)GO TO 2500
489              I1=I
490              I2=2*K
491              GO TO 1440
492       1435   E=1
493              IF(I.GT.5)GO TO 2500
494              E=8
495              IF(IBURN(K).EQ.0)GO TO 2500
496              I1=ICOL+IC(I)
497              I2=2*K-1
498       1440   E=4
499              IF(K.EQ.0) GO TO 2500
500              IF(JWORD(1).NE.JWORD(2)) GO TO 1455
501              E=3
502              IF(IVAR(I1,I2).EQ.IBLANK)GO TO 2500
503              E=1
504              DO 1445 NN=3,IWORD
505       1445   IF(ITYPE(NN).NE.NUMBER) GO TO 2500
506              DO 1450 NN=3,IWORD
507              IF(IOP(NN).EQ.PLUS )VAR(I1,I2)=VAR    (I1,I2)+WORD(NN)
508              IF(IOP(NN).EQ.MINUS)VAR(I1,I2)=VAR    (I1,I2)-WORD(NN)
509              IF(IOP(NN).EQ.STAR )VAR(I1,I2)=VAR    (I1,I2)*WORD(NN)
510              IF(IOP(NN).EQ.SLASH)VAR(I1,I2)=VAR    (I1,I2)/WORD(NN)
511       1450   CONTINUE
512              GO TO 200
```

```
 1455  E=1
       IF(ITYPE(2).NE.NUMBER.AND.JWORD(2).NE.OLD.AND.JWORD(2).NE.IBLANK
       .OR.IWORD.GT.2)GO TO 2500
       IF(IOP(2).EQ.MINUS)VALUE=-VALUE
       VAR(I1,I2)=VALUE
       IF(JWORD(2).EQ.OLD) VAR(I1,I2)=VARFNL(I1,I2)
       GO TO 200
C
C                 'LAST' CARD---BEGIN COMPUTATION
C
 1500  KSENS=0
       KSTEP=0
       E=5
       IF((ISENS.EQ.SENS.OR.ISENS.EQ.DELTA).AND.IVAR(ICSENS,IRSENS)
       .EQ.IBLANK)GO TO 2500
       IF(ISTEP.EQ.STEP.AND.IVAR(ICSTEP,IRSTEP).EQ.IBLANK)GO TO 2500
C
C                 SAVE CURRENT VALUES AND INITIALIZE
C
 1600  DO 1605 I=1,ICP2
       DO 1605 J=1,IRX2
 1605  VARSAV(I,J)=VAR(I,J)
       DO 1610 I=1,ICOL
 1610  CTAGSV(I)=CTAG(I)
       DO 1620 I=1,IROW
       RTAGSV(I)=RTAG(I)
       IBURNS(I)=IBURN(I)
 1620  LNFLGS(I)=LINFLG(I)
       IF(ISTAT.EQ.ALL.OR.ISTAT.EQ.PART)GO TO 2200
 1630  DO 1640 I=1,ICOL
       DO 1640 J=1,IRM1
 1640  SDONE(I,J)=.FALSE.
       DO 1650 I=1,IRX2
 1650  TDONE(I)=.FALSE.
       DO 1660 I=1,IROW
 1660  RDONE(I)=IBURN(I).EQ.0
       DO 1670 I=1,IROW
       IF(RDONE(I))GO TO 1670
       I1=IBURN(I)
       I2=2*I
       IF(IVAR(I1,I2).NE.IBLANK.AND.VAR(I1,I2).GT.0.)
       VAR(I1,I2)=-VAR(I1,I2)
 1670  CONTINUE
       NBLANK=ICP2*IRX2
C
C                 COMPUTATION LOOP
C
 1700  DO 1750 J=1,IROW
       ERRTAG=RTAG(J)
       L=2*J
       LM1=L-1
       LP1=L+1
       IF(RDONE(J))GO TO 1720
       M=IBURN(J)
       N=0
       DO 1710 I=1,ICM1
       IF(I.EQ.M)GO TO 1710
```

```
      N=N+1
      A(N)=VAR(I,L)
 1710 CONTINUE
      MC=BLANK
      CALL SUMS(MC,A,N,DUMMY,ITFLG,ERRTAG,$1770)
      N=0
      DO 1715 I=1,ICM1
      IF(I.EQ.M)GO TO 1715
      N=N+1
      VAR(I,L)=A(N)
 1715 CONTINUE
      A(1)=VAR(M,L)
      A(2)=MC
      CALL SUMS(VAR(ICOL,L),A,2,DUMMY,ITFLG,ERRTAG,$1770)
      VAR(M,L)=A(1)
      MC=A(2)
      IF(BOOL(MC).NE.BLANK)MC=-MC
      IF(IVAR(M,L).NE.IBLANK)VAR(M,L)=-VAR(M,L)
      CALL ROCKET(VAR(ICP1,LM1),VAR(ICP2,LM1),VAR(ICOL,LM1),VAR(M,L),
     . MC,VAR(ICOL,LP1),LINFLG(J),RDONE(J),ITFLG,ERRTAG,$1770)
      IF(IVAR(M,L).NE.IBLANK)VAR(M,L)=-VAR(M,L)
 1720 DO 1730 I=1,ICM1
 1730 A(I)=VAR(I,LM1)
      IF(.NOT.TDONE(LM1))CALL SUMS(VAR(ICOL,LM1),A,ICM1,TDONE(LM1)
     . ,ITFLG,ERRTAG,$1770)
      DO 1735 I=1,ICM1
 1735 VAR(I,LM1)=A(I)
      IF(J.EQ.IROW)GO TO 1800
      DO 1740 I=1,ICM1
 1740 A(I)=VAR(I,L)
      IF(.NOT.TDONE(L))CALL SUMS(VAR(ICOL,L),A,ICM1,TDONE(L),ITFLG,
     . ERRTAG,$1770)
      DO 1745 I=1,ICM1
 1745 VAR(I,L)=A(I)
      DO 1750 I=1,ICOL
      A(1)=VAR(I,LM1)
      A(2)=VAR(I,L)
      IF(.NOT.SDONE(I,J))CALL SUMS(VAR(I,LP1),A,2,SDONE(I,J),
     . ITFLG,ERRTAG,$1770)
      VAR(I,LM1)=A(1)
 1750 VAR(I,L)=A(2)
      GO TO 1800
 1770 IF(ITFLG.NE.YES)GO TO 1880
      ERRFLG=GOOF
      GO TO 1860
C
C                CHECK FOR VALID SOLUTION
C
 1800 NBLNKS=NBLANK
      NBLANK=0
      DO 1810 I=1,ICP2
      DO 1810 J=1,IRX2
 1810 IF(IVAR(I,J).EQ.IBLANK)NBLANK=NBLANK+1
      IF(NBLANK.EQ.0.AND.ITFLG.EQ.YES)GO TO 1860
      IF(NBLANK.EQ.0)GO TO 1900
      IF(NBLANK.NE.NBLNKS)GO TO 1700
C
```

```
C                NOT ENOUGH DATA
C
 1820  IF (ITRATE .EQ. YES .AND. ITFLG .NE. YES) CALL TEST (IBURN,IRKC,
     . IROCK,JCOL,LINFLG,NROW,VAR,$1840)
       IF(ITRATE.EQ.YES)GO TO 1860
       PRINT 1830
 1830  FORMAT(' NO SOLUTION--DATA INSUFFICIENT (NO ATTEMPT TO ITERATE)')
       GO TO 1880
 1840  PRINT 1850
 1850  FORMAT(' NO SOLUTION--DATA INSUFFICIENT')
       GO TO 2050
 1860  CALL ITERAT(ERRFLG,IBURNS,ICSENS,ICSTEP,ISENS,ISTEP,IRSENS
     .,IRSTEP,ITDUN,ITFLG,KSENS,KSTEP,NBLANK,RDONE,SDONE,TDONE,VAR
     .,VARSAV)
       IT=STAR
       IF(PITFLG.NE.ITFL)IT=IBLANK
       IF(ITFLG.EQ.YES.OR.ITDUN)GO TO 1700
       PRINT 1870
 1870  FORMAT(' NO SOLUTION--DATA INCONSISTENT (ITERATION FAILED)')
 1880  KSENS=2
       KSTEP=NSTEPS
       GO TO 2050
C
C                NBLANK=0---A VALID SOLUTION HAS BEEN OBTAINED
C
 1900  IF((ISENS.EQ.SENS.OR.ISENS.EQ.DELTA).AND.KSENS.GT.0)GO TO 1920
C
C                SAVE RESULTS
C
       DO 1910 I=1,ICP2
       DO 1910 J=1,IRX2
 1910  VARFNL(I,J)=VAR(I,J)
 1920  IF(ISENS.NE.SENS.AND.ISENS.NE.DELTA)GO TO 2050
C
C                CALCULATE SENSITIVITIES
C
       KSENS=KSENS+1
       IF(KSENS.EQ.1)GO TO 130
       DENOM=VAR(ICSENS,IRSENS)-VARFNL(ICSENS,IRSENS)
       E=12
       DO 2000 I=1,ICP2
       DO 2000 J=1,IRX2
       VAR(I,J)=VAR(I,J)-VARFNL(I,J)
       IF(ISENS.EQ.DELTA)GO TO 2000
       IF(ABS(DENOM).LE..00001)GO TO 2500
       VAR(I,J)=VAR(I,J)/DENOM
 2000  CONTINUE
 2050  IF(NBLANK.NE.0.AND.(PGOOF.NE.GOOF.OR.ISENS.EQ.SENS.OR.ISENS.EQ.
     . DELTA.OR.PTABLE.EQ.TABLE))GO TO 130
C
C                PRINT RESULTS
C
       KSTEP=KSTEP+1
       IF(PTABLE.NE.TABLE)GO TO 2200
       CALL PRINT3(CTABLE,IT,ICTABL,IRTABL,KSTEP,NCOL,PTABLE,RTABLE,VAR)
       IF(KSTEP.EQ.NSTEPS)PRINT 2100
 2100  FORMAT(/)
```

```
      GO TO 130
 2200 IF(PLABEL.NE.LABELS)CALL PRINT1(CPRT,CTAG,IBURN,ISTAT,IT,LINFLG
     .,RPRT,RTAG,VAR)
      IF(PLABEL.EQ.LABELS)CALL PRINT2(CHEAD,CPRT,IBURN,ISTAT,IT,LINFLG
     .,RHEAD,RPRT,RSUBHD,VAR)
      GO TO 130
C
C                PRINT ERROR MESSAGES
C
 2500 IF(PDATA.NE.DATA)PRINT 950,COLUMN
      DO 2550 NN=1,12
      NNN=(E-1)*12+NN
 2550 F(NN)=FORM(NNN)
      PRINT F
      IF(E.EQ.12)GO TO 2575
      E=1
      GO TO 200
 2575 KSENS=2
      KSTEP=NSTEPS
      GO TO 130
C
C                INTERNAL SUBROUTINE
C
      SUBROUTINE VARBLE(NAME,ITAG,INDEX,RTN)
C
C                SUBROUTINE VARBLE DETERMINES IF NAME IS ONE OF THE
C                 DIMENSIONED VARIABLES DEFINING THE MATRIX.  IF SO, IT
C                 RETURNS A NUMBER ITAG WHICH IDENTIFIES THE 'TAG' ASSOCIATED
C                 WITH THE VARIABLE AND ITS SUBSCRIPT (INDEX).
C                 IF NOT, IT RETURNS TO STATEMENT NUMBER RTN.
C
C
      ITAG=0
      INDEX=0
      DO 10 LL=1,IROW
      IF(NAME.NE.RTAG(LL))GO TO 10
      ITAG=1
      GO TO 50
   10 CONTINUE
      DO 20 LL=1,ICOL
      IF(NAME.NE.CTAG(LL))GO TO 20
      ITAG=2
      GO TO 50
   20 CONTINUE
      DO 30 LL=1,ICOL
      IF(NAME.NE.DTAG(LL))GO TO 30
      ITAG=3
      GO TO 50
   30 CONTINUE
      DO 40 LL=1,5
      IF(NAME.NE.TAG1(LL))GO TO 40
      ITAG=4
      GO TO 50
   40 CONTINUE
      RETURN 4
   50 INDEX=LL
      RETURN
      END
```

```
      SUBROUTINE ITERAT(ERRFLG,IBURNS,ICSENS,ICSTEP,ISENS,ISTEP,IRSENS
     .,IRSTEP,ITDUN,ITFLG,KSENS,KSTEP,NBLANK,RDONE,SDONE,TDONE,VAR
     .,VARSAV)
C
      COMMON /DIM/ ICM1,ICOL,ICP1,ICP2,IRM1,IROW,IRX2
C
      DIMENSION                 IBURNS(IROW),          VAR  (ICP2,IRX2)
     .,VARSAV(ICP2,IRX2)
C
      INTEGER                   DELTA /'DELTA '/,      ERRFLG
     .,GOOF  /'GOOF  '/,        NO    /'NO    '/,      NOGOOF/'NOGOOF'/
     .,SENS  /'SENS  '/,        STEP  /'STEP  '/,      YES   /'YES   '/
C
      DATA                      BLANK /'      '/
C
      LOGICAL                   ITDUN,                 RDONE(IROW)
     .,SDONE(ICOL,IRM1),        TDONE(IRX2),           X
C
C
      IF(ITFLG.EQ.NO)ISTART=0
      X=ISENS.EQ.SENS.OR.ISENS.EQ.DELTA
      IF(ISTART.GT.0)GO TO 600
C
C                INITIALIZE
C
      CALL INIT
      M=1
      AVG=0.0
      L=0
      DO 100 I=1,ICOL
      DO 100 J=1,IRX2,2
      IF(BOOL(VARSAV(I,J)).EQ.BLANK)GO TO 100
      L=L+1
      AVG=AVG+VARSAV(I,J)
  100 CONTINUE
      IF(L.GT.0)AVG=AVG/FLOAT(L)
      ANS=0.0
      ITFLG=YES
      ITDUN=.FALSE.
      IF(X.AND.KSENS.EQ.1)GO TO 500
      IF(ISTEP.EQ.STEP.AND.KSTEP.GT.0)GO TO 500
      IUNK=1
      JUNK=0
      IKNWN=1
      JKNWN=0
C
C                CHOOSE AN UNKNOWN VARIABLE FOR ITERATION
C
  200 DO 250 I=IUNK,ICP2
      DO 250 J=M,IRX2,2
      IF(I.EQ.IUNK.AND.J.LE.JUNK)GO TO 250
      IF(BOOL(VARSAV(I,J)).NE.BLANK)GO TO 250
      IF(I.GT.ICOL.AND.(MOD(J,2).EQ.0.OR.J.EQ.IRX2))GO TO 250
      IUNK=I
      JUNK=J
      GO TO 300
```

```
  250   CONTINUE
        IF(M.EQ.2)GO TO 750
        M=2
        IUNK=1
        IKNWN=1
        JKNWN=0
        GO TO 200
C
C               FIND AN INITIAL GUESS FOR ITERATION
C
  300   IF(IUNK.LE.ICOL)GO TO 305
        TRY=300.
        GO TO 390
  305   TRY=0.0
        ITRY=0
        IF(JUNK.EQ.1)GO TO 330
        IX=JUNK-1-MOD(JUNK,2)
        DO 310 I=IX,1,-2
  310   IF(BOOL(VARSAV(IUNK,I)).NE.BLANK)GO TO 315
        GO TO 320
  315   ITRY=1
        VI=VARSAV(IUNK,I)
  320   JTRY=0
        IF(JUNK.EQ.IRX2)GO TO 350
        JX=JUNK+1+MOD(JUNK,2)
        DO 330 J=JX,IRX2,2
        IF(BOOL(VARSAV(IUNK,J)).NE.BLANK)GO TO 340
  330   CONTINUE
        GO TO 350
  340   JTRY=J
        VJ=VARSAV(IUNK,J)
  350   IF(ITRY.EQ.0.OR.JTRY.EQ.0)GO TO 355
        TRY=(VI+VJ)/2.
        IF(MOD(JUNK,2).EQ.0)TRY=-VI+VJ
        GO TO 385
  355   IF(ITRY.EQ.0)GO TO 360
        TRY=VI
        IF(MOD(JUNK,2).EQ.0)TRY=-VI/2.
        GO TO 385
  360   IF(JTRY.EQ.0)GO TO 365
        TRY=VJ
        IF(MOD(JUNK,2).EQ.0)TRY=-VJ
        GO TO 385
  365   IF(BOOL(VARSAV(ICOL,JUNK)).EQ.BLANK)GO TO 380
        TRY=VARSAV(ICOL,JUNK)
        DO 370 I=1,ICM1
  370   IF(BOOL(VARSAV(I,JUNK)).NE.BLANK)TRY=TRY-VARSAV(I,JUNK)
        GO TO 385
  380   TRY=AVG
  385   IF(ABS(TRY).GT.1.)GO TO 390
        TRY=1000.
        IF(MOD(JUNK,2).EQ.0)TRY=-1000.
  390   CONTINUE
C
C               CHOOSE A KNOWN VARIABLE FOR COMPARISON
C
        DO 400 I=IKNWN,ICP2
```

```
      DO 400 J=1,IRX2
      IF(I.EQ.IKNWN.AND.J.LE.JKNWN)GO TO 400
      IF(I.EQ.IUNK.AND.J.EQ.JUNK)GO TO 400
      IF(BOOL(VARSAV(I,J)).EQ.BLANK)GO TO 400
      IF(I.GT.ICOL.AND.(MOD(J,2).EQ.0.OR.J.EQ.IRX2))GO TO 400
      IF(X.AND.I.EQ.ICSENS.AND.J.EQ.IRSENS)GO TO 400
      IF(ISTEP.EQ.STEP.AND.I.EQ.ICSTEP.AND.J.EQ.IRSTEP)GO TO 400
      IKNWN=I
      JKNWN=J
      GO TO 500
  400 CONTINUE
      IKNWN=ICP2
      JKNWN=IRX2
      GO TO 700
C
C             VARIABLES HAVE BEEN CHOSEN--RESUME CALCULATIONS
C
  500 ISTART=ISTART+1
      REF=VARSAV(IKNWN,JKNWN)
      VAR(IUNK,JUNK)=TRY
      VAR(IKNWN,JKNWN)=BLANK
      NBLANK=ICP2*IRX2
      RETURN
C
C             THIS IS AT LEAST SECOND ITERATION FOR THESE VARIABLES
C              ---CONTINUE
C
  600 IF(ERRFLG.EQ.GOOF)GO TO 700
      IF(BOOL(VAR(IKNWN,JKNWN)).EQ.BLANK)GO TO 700
      ANSLST=ANS
      ANS=VAR(IKNWN,JKNWN)
      IF(ABS(ANS-REF).LT..01)GO TO 800
      CALL INIT
C
C             HOW MANY ITERATIONS SO FAR?
C
      IF(ISTART.GT.20)GO TO 700
      IF(ISTART.EQ.1)GO TO 640
      IF(ABS(ANS-ANSLST).LT..0001)GO TO 700
C
C             FIND NEW GUESS FOR NEXT ITERATION
C
      TRYNEW=TRY+(TRY-TRYLST)*(REF-ANS)/(ANS-ANSLST)
      GO TO 660
  640 TRYNEW=TRY*1.01
  660 TRYLST=TRY
      TRY=TRYNEW
      IF(ABS(TRY).GT.5000000.)GO TO 700
      GO TO 500
C
C             NO SOLUTION WITH THESE VARIABLES--TRY A NEW SET
C
  700 ISTART=0
      ANS=0.0
      ERRFLG=NOGOOF
      CALL INIT
      IF(.NOT.(IKNWN.EQ.ICP2.AND.JKNWN.EQ.IRX2))GO TO 300
```

```
171              IKNWN=1
172              JKNWN=0
173              IF(.NOT.(IUNK.EQ.ICP2.AND.JUNK.EQ.IRX2))GO TO 200
174        C
175        C                NO SOLUTION--RETURN
176        C
177         750  ISTART=0
178              ITFLG=NO
179              ITDUN=.FALSE.
180              RETURN
181        C
182        C                REFERENCE HAS BEEN MATCHED--RETURN
183        C
184         600  IF(NBLANK.NE.0)GO TO 700
185              ISTART=0
186              ITFLG=NO
187              ITDUN=.TRUE.
188              CALL INIT
189              VAR(IUNK,JUNK)=TRY
190              RETURN
191        C
192        C                INTERNAL SUBROUTINE
193        C
194              SUBROUTINE INIT
195        C
196              DO 10 I=1,ICP2
197              DO 10 J=1,IRX2
198              IF(X.AND.I.EQ.ICSENS.AND.J.EQ.IRSENS)GO TO 10
199              IF(ISTEP.EQ.STEP.AND.I.EQ.ICSTEP.AND.J.EQ.IRSTEP)GO TO 10
200              VAR(I,J)=VARSAV(I,J)
201         10   CONTINUE
202              DO 20 I=1,IROW
203         20   RDONE(I)=IBURNS(I).EQ.0
204              DO 30 I=1,ICOL
205              DO 30 J=1,IRM1
206         30   SDONE(I,J)=.FALSE.
207              DO 40 I=1,IRX2
208         40   TDONE(I)=.FALSE.
209        C
210              RETURN
211        C
212              END
```

PRINT1

```
      SUBROUTINE PRINT1(CPRT,CTAG,IBURN,ISTAT,IT,LINFLG,RPRT,RTAG,VAR)
C
      COMMON /DIM/ ICM1,ICOL,ICP1,ICP2,IRM1,IROW,IRX2
      COMMON /PRT/ C,DELTA,DENOM,HDG,HEAD,ICTR,IHDG,ISENS,PDELTA,PHDG
     . ,SENS,SNSCOL,SNSROW
C
      REAL                    LINEAR
C
      LOGICAL                 CPRT(ICOL),             LINFLG(IROW)
     .,RPRT(IROW)
C
      DIMENSION               CTAG(ICOL),             F(11)
     .,FD(5),                 FMT(4),                 FRMT1(13)
     .,FRMT2(13),             FS(5),                  HEAD(5,80)
     .,IBURN(IROW),           RTAG(IROW),             VAR(ICP2,IRX2)
C
      INTEGER                 ALL    /'ALL    '/,     CENTER/'CENTER'/
     .,DELTA /'DELTA '/,      DV     /'V      '/,     HDG    /'HDG    '/
     .,ISP   /'I     '/,      NO     /'NO     '/,     PART   /'PART   '/
     .,PDELTA,                PHDG,                  SENS   /'SENS   '/
C
      DATA                    BLANK /'       '/,     F(1)   /'(@@X, '/
     .,(F(I),I=7,11)/',A6,''('',A6,''), DELTA='',F9.3)'/
     .,F1    /')     '/,      FD     /''''CHANGES DUE TO CHANGE IN ''''/
     .,FMT(1)/'(2X,  '/,      FMT(2)/'A1,''='''/,     FMT(4)/')      '/
     .,FRMT1(1)/'(/A7,'/,     FRMT1(2)/'1X  '/,       FRMT2(1)/'(2X,'/
     .,FRMT2(2)/'A6, '/,      FS     /''''SENSITIVITIES TO '''',' '/
     .,LINEAR/'LINEAR'/,      UNKNWN/'UNKNWN'/
C
      EQUIVALENCE             (BLANK,IBLANK)
C
C               INITIALIZE
C
  100 ICOLSV=ICOL
      IROWSV=IROW
      IF(ISTAT.EQ.ALL)GO TO 290
      IROWNU=1
      ICOLNU=1
      IV1=ICP2
C
C               ELIMINATE UNWANTED ROWS
C
      DO 280 IR=1,IROW
      IRN=2*IROWNU-1
      IRO=2*IR-1
      IF(.NOT.RPRT(IR))GO TO 280
      DO 200 IC=1,IV1
      DO 200 I=0,1
      IF(I.EQ.1.AND.IR.EQ.IROW)GO TO 200
      VAR(IC,IRN+I)=VAR(IC,IRO+I)
  200 CONTINUE
      RTAG(IROWNU)=RTAG(IR)
      LINFLG(IROWNU)=LINFLG(IR)
      IBURN(IROWNU)=IBURN(IR)
      IROWNU=IROWNU+1
C
```

```
C                 ELIMINATE UNWANTED COLUMNS
C
          ICOLNU=1
          DO 260 IC=1,ICOL
          IF(.NOT.CPRT(IC))GO TO 260
          DO 220 I=0,1
          IF(I.EQ.1.AND.IR.EQ.IROW)GO TO 220
          VAR(ICOLNU,IRN+I)=VAR(IC,IRN+I)
  220     CONTINUE
          IF(IROWNU.GT.2)GO TO 240
          CTAG(ICOLNU)=CTAG(IC)
  240     ICOLNU=ICOLNU+1
  260     CONTINUE
          VAR(ICOLNU,IRN)=VAR(ICOL+1,IRN)
          VAR(ICOLNU+1,IRN)=VAR(ICOL+2,IRN)
  280     CONTINUE
          IROW=IROWNU-1
          ICOL=ICOLNU-1
          IF(IROW.EQ.0.OR.ICOL.EQ.0)GO TO 900
  290     IV1=ICOL+2
          ICP1=ICOL+1
          FRMT1(IV1+1)=F1
          FRMT2(IV1+1)=F1
          NC=8+10*ICOL
          IF(PHDG.EQ.HDG.OR.ISENS.EQ.SENS.OR.ISENS.EQ.DELTA)PRINT 295
  295     FORMAT(/)
          IF(PHDG.NE.HDG)GO TO 360
C
C                 PRINT HEADING
C
          DO 340 J=1,IHDG
          K=1
          K1=1
          K2=79
          IF(ICTR.NE.CENTER)GO TO 340
  300     CALL POS(HEAD,J,K1,K2)
          IF(K1.NE.0)GO TO 310
          K1=1
          K2=1
  310     K=MAX0(1,(NC+K1-K2)/2)
  340     PRINT 350,(BLANK,I=1,K),(HEAD(J,I),I=K1,K2)
  350     FORMAT(120A1)
  360     K=1
          IF(ISENS.NE.SENS.OR.ISTAT.EQ.ALL.OR.ISTAT.EQ.PART)GO TO 380
C
C                 PRINT SENSITIVITY OR DELTA HEADING
C
          IF(NC.GT.47)K=(NC-46)/2
          CALL NX(K,F(1))
          DO 370 I=2,6
  370     F(I)=FS(I-1)
          WRITE(6,F)SNSCOL,SNSROW,DENOM
          GO TO 400
  380     IF(ISENS.NE.DELTA.OR.ISTAT.EQ.ALL.OR.ISTAT.EQ.PART)GO TO 400
          IF(NC.GT.55)K=(NC-54)/2
          CALL NX(K,F(1))
          DO 390 I=2,6
```

```
  390   F(I)=FD(I-1)
        WRITE(6,F)SNSCOL,SNSROW,TENOM
C
C               PRINT COLUMN HEADINGS
C
  400   PRINT 410,IT,(CTAG(I),I=1,ICOL)
  410   FORMAT(//5X,A1,4X,10A10)
C
C               BEGIN LOOP FOR PRINTING MATRIX
C
        DO 800 I=1,IROW
C
C               PRINT EVENT ROW
C
        I2=2*I-1
        DO 500 I1=1,ICOL
        CALL FORM(VAR(I1,I2),FRMT1(I1+2),1)
  500   CONTINUE
        WRITE(6,FRMT1)RTAG(I),(VAR(I1,I2),I1=1,ICOL)
        IF(I.EQ.IROW)GO TO 900
C
C               PRINT DELTA ROW
C
        I2=2*I
        IF(PDELTA.NE.DELTA)GO TO 700
        WORD=BLANK
        IF(LINFLG(I))WORD=LINEAR
        DO 600 I1=1,ICOL
        CALL FORM(VAR(I1,I2),FRMT2(I1+2),1)
  600   CONTINUE
        WRITE(6,FRMT2)WORD,(VAR(I1,I2),I1=1,ICOL)
C
C               PRINT DELTAV AND ISP DATA
C
  700   I2=2*I-1
        IF(IBURN(I).EQ.0)GO TO 800
        DO 750 L=1,2
        NAME=DV
        IF(L.EQ.2)NAME=ISP
        I1=ICOL+L
        CALL FORM(VAR(I1,I2),FMT(5),2)
        WRITE(6,FMT)NAME,VAR(I1,I2)
  750   CONTINUE
C
  800   CONTINUE
C
C               REINITIALIZE
C
  900   ICOL=ICOLSV
        IROW=IROWSV
        ICP1=ICOL+1
        ISTAT=NO
        PRINT 295
C
C               INTERNAL SUBROUTINE
C
        SUBROUTINE NX(K,F)
```

```
C
      DATA                    IZERO /'000000'/
      KA=K/10+IZERO
      KB=MOD(K,10)+IZERO
      FLD(6,6,F)=KA
      FLD(12,6,F)=KB
      RETURN
C
C                INTERNAL SUBROUTINE
C
      SUBROUTINE FORM(V,F,L)
C
      F=',A6'
      IF(BOOL(V).EQ.BLANK)GO TO 20
      IF(ABS(V).GE..00005)GO TO 30
   10 V=BLANK
      IF(L.EQ.1)F=',A10'
      RETURN
   20 V=UNKNWN
      IF(L.EQ.1)F=',A10'
      RETURN
   30 IF(ISTAT.EQ.PART.OR.ISTAT.EQ.ALL)GO TO 40
      IF(ISENS.EQ.SENS)GO TO 70
      IF(ABS(V).LT..0005)GO TO 10
      IF(ISENS.EQ.DELTA)GO TO 60
   40 IF(ABS(V).GE..05)GO TO 50
      IF(ABS(V).LT..005)GO TO 10
      IF(L.EQ.1)GO TO 10
   50 F=',F10.1'
      IF(L.EQ.2)F=',F7.2'
      RETURN
   60 F=',F10.3'
      IF(L.EQ.2)F=',F8.3'
      RETURN
   70 F=',F10.4'
      IF(L.EQ.2)F=',F8.4'
      RETURN
      END
```

```
      SUBROUTINE PRINT2(CHEAD,CPRT,IBURN,ISTAT,IT,LINFLG,RHEAD,RPRT
     .,RSUBHD,VAR)
C
      COMMON /DIM/ ICM1,ICOL,ICP1,ICP2,IRM1,IROW,IRX2
      COMMON /PRT/ C,DELTA,DENOM,HDG,HEAD,ICTR,IHDG,ISENS,PDELTA,PHDG
     . ,SENS,SNSCOL,SNSROW
C
      REAL                    LINEAR
C
      LOGICAL                 CPRT(ICOL),             LINFLG(IROW)
     .,RPRT(IROW)
C
      DIMENSION               CHEAD(2,3,ICOL),        F(11)
     .,FD(5),                                        FMT(7)
     .,FRMT1(13),            FRMT2(13),              FS(5)
     .,HEAD(5,80),           IBURN(IROW),            RHEAD(5,IROW)
     .,RSUBHD(5,3,IROW),     VAR(ICP2,IRX2)
C
      INTEGER                 ALL    /'ALL    '/,     CENTER/'CENTER'/
     .,CHEAD,                 DELTA /'DELTA '/,      HDG    /'HDG    '/
     .,NO     /'NO     '/,    PART  /'PART  '/,      PDELTA
     .,PHDG,                  SENS  /'SENS  '/
C
      DATA                    BLANK /'      '/,      F(1)  /'(QQX, '/
     .,(F(I),I=7,11)/',A6,''('',A6,''), DELTA='',F9.3)'/
     .,F1     /')      '/,    FD     /'''CHANGES DUE TO CHANGE IN '''/
     .,FMT(1)/'(3X,   '/,    FMT(2)/'''ΔV='''/,      FMT(4)/','' ISP'/
     .,FMT(5)/'=''   '/,     FMT(7)/',A2)   '/,      FRMT1(1)/'(/A7,'/
     .,FRMT1(2)/'3A6,A4'/,   FRMT2(1)/'(3X,'/,       FRMT2(2)/'4A6,A2'/
     .,FS     /'''SENSITIVITIES TO '''','  '/,        LINEAR/' L     '/
     .,UNKNWN/'UNKNWN'/
C
      EQUIVALENCE             (BLANK,IBLANK)
C
C               CHOOSE FORMAT FOR NORMAL OR SENSITIVITIES
C
  100 ICOLSV=ICOL
      IROWSV=IROW
      IF(ISTAT.EQ.ALL)GO TO 290
      IROWNU=1
      ICOLNU=1
      IV1=ICP2
C
C               ELIMINATE UNWANTED ROWS
C
      DO 280 IR=1,IROW
      IRN=2*IROWNU-1
      IRO=2*IR-1
      IF(.NOT.RPRT(IR))GO TO 280
      DO 200 IC=1,IV1
      DO 200 I=0,1
      IF(I.EQ.1.AND.IR.EQ.IROW)GO TO 200
      VAR(IC,IRN+I)=VAR(IC,IRO+I)
  200 CONTINUE
      DO 210 I=1,5
      RHEAD(I,IROWNU)=RHEAD(I,IR)
```

```
      DO 210 J=1,3
  210 RSUBHD(I,J,IROWNU)=RSUBHD(I,J,IR)
      LINFLG(IROWNU)=LINFLG(IR)
      IBURN(IROWNU)=IBURN(IR)
      IROWNU=IROWNU+1
C
C                ELIMINATE UNWANTED COLUMNS
C
      ICOLNU=1
      DO 260 IC=1,ICOL
      IF(.NOT.CPRT(IC))GO TO 260
      DO 220 I=0,1
      IF(I.EQ.1.AND.IR.EQ.IROW)GO TO 220
      VAR(ICOLNU,IRN+I)=VAR(IC,IRN+I)
  220 CONTINUE
      IF(IROWNU.GT.2)GO TO 240
      DO 230 J=1,3
      CHEAD(1,J,ICOLNU)=CHEAD(1,J,IC)
  230 CHEAD(2,J,ICOLNU)=CHEAD(2,J,IC)
  240 ICOLNU=ICOLNU+1
  260 CONTINUE
      VAR(ICOLNU,IRN)=VAR(ICOL+1,IRN)
      VAR(ICOLNU+1,IRN)=VAR(ICOL+2,IRN)
  280 CONTINUE
      IROW=IROWNU-1
      ICOL=ICOLNU-1
      IF(IROW.EQ.0.OR.ICOL.EQ.0)GO TO 900
  290 IV1=ICOL+2
      ICP1=ICOL+1
      FRMT1(IV1+1)=F1
      FRMT2(IV1+1)=F1
      NC=30+10*ICOL
      PRINT 295
  295 FORMAT('1')
      IF(PHDG.NE.HDG)GO TO 360
C
C                PRINT HEADING
C
      DO 340 J=1,IHDG
      K=1
      K1=1
      K2=79
      IF(ICTR.NE.CENTER)GO TO 340
  300 CALL POS(HEAD,J,K1,K2)
      IF(K1.NE.0)GO TO 310
      K1=1
      K2=1
  310 K=MAX0(1,(NC+K1-K2)/2)
  340 PRINT 350,(BLANK,I=1,K),(HEAD(J,I),I=K1,K2)
  350 FORMAT(120A1)
  360 K=1
      IF(ISENS.NE.SENS.OR.ISTAT.EQ.ALL.OR.ISTAT.EQ.PART)GO TO 380
C
C                PRINT SENSITIVITY OR DELTA HEADING
C
      IF(NC.GT.47)K=(NC-46)/2
      CALL NX(K,F(1))
```

```
      DO 370 I=2,6
  370 F(I)=FS(I-1)
      WRITE(6,F)SNSCOL,SNSROW,DENOM
      GO TO 400
  380 IF(ISENS.NE.DELTA.OR.ISTAT.EQ.ALL.OR.ISTAT.EQ.PART)GO TO 400
      IF(NC.GT.55)K=(NC-54)/2
      CALL NX(K,F(1))
      DO 390 I=2,6
  390 F(I)=FD(I-1)
      WRITE(6,F)SNSCOL,SNSROW,DENOM
C
C                 PRINT COLUMN HEADINGS
C
  400 PRINT 410
  410 FORMAT(/)
      DO 420 I1=1,3
      DO 420 I=1,ICOL
  420 IF(CHEAD(1,I1,I).NE.IBLANK.OR.CHEAD(2,I1,I).NE.IBLANK)GO TO 430
      GO TO 480
  430 DO 440 I2=3,I1,-1
      DO 440 I=1,ICOL
  440 IF(CHEAD(1,I2,I).NE.IBLANK.OR.CHEAD(2,I2,I).NE.IBLANK)GO TO 450
  450 DO 460 I=I1,I2
      NAME=IBLANK
      IF(I.EQ.I1)NAME=IT
  460 PRINT 470,NAME,(CHEAD(1,I,J),CHEAD(2,I,J),J=1,ICOL)
  470 FORMAT(5X,A1,26X,10(A6,A4))
  480 I2=0
C
C                 BEGIN LOOP FOR PRINTING MATRIX
C
      DO 850 I=1,IROW
C
C                 PRINT EVENT ROW
C
      I2=2*I-1
      DO 500 I1=1,ICOL
      CALL FORM(VAR(I1,I2),FRMT1(I1+2),1)
  500 CONTINUE
      WRITE(6,FRMT1)(RHEAD(I1,I),I1=1,5),(VAR(I1,I2),I1=1,ICOL)
      IF(I.EQ.IROW)GO TO 900
C
C                 DELTA WEIGHT ROWS
C
C
C                 PRINT DELTAV AND ISP DATA
C
      IF(IBURN(I).EQ.0)GO TO 650
      WORD=BLANK
      IF(LINFLG(I))WORD=LINEAR
      DO 600 L=1,2
      I1=ICOL+L
      CALL FORM(VAR(I1,I2),FMT(3*L),2)
  600 CONTINUE
      WRITE(6,FMT)VAR(ICP1,I2),VAR(IV1,I2),WORD
C
C
```

```
 650    IF(PDELTA.NE.DELTA)GO TO 850
C
C               PRINT FIRST SUBHEADING ROW
C
 700    PRINT 710,(RSUBHD(I1,1,I),I1=1,5)
 710    FORMAT(3X,5A6)
C
C               PRINT SECOND SUBHEADING AND DELTA ROW
C
        I2=2*I
        DO 800 I1=1,ICOL
        CALL FORM(VAR(I1,I2),FRMT2(I1+2),1)
 800    CONTINUE
        WRITE(6,FRMT2)(RSUBHD(I1,2,I),I1=1,5),(VAR(I1,I2),I1=1,ICOL)
        PRINT 710,(RSUBHD(I1,3,I),I1=1,5)
C
 850    CONTINUE
C
C               REINITIALIZE
C
 900    ICOL=ICOLSV
        IROW=IROWSV
        ICP1=ICOL+1
        ISTAT=NO
        PRINT 410
C
C               INTERNAL SUBROUTINE
C
        SUBROUTINE NX(K,F)
C
        DATA                    IZERO /'@@@@@@'/
        KA=K/10+IZERO
        KB=MOD(K,10)+IZERO
        FLD(6,6,F)=KA
        FLD(12,6,F)=KB
        RETURN
C
C               INTERNAL SUBROUTINE
C
        SUBROUTINE FORM(V,F,L)
C
        F=',A6'
        IF(BOOL(V).EQ.BLANK)GO TO 20
        IF(ABS(V).GE..00005)GO TO 30
 10     V=BLANK
        IF(L.EQ.1)F=',A10'
        RETURN
 20     V=UNKNWN
        IF(L.EQ.1)F=',A10'
        RETURN
 30     IF(ISTAT.EQ.PART.OR.ISTAT.EQ.ALL)GO TO 40
        IF(ISENS.EQ.SENS)GO TO 70
        IF(ABS(V).LT..0005)GO TO 10
        IF(ISENS.EQ.DELTA)GO TO 60
 40     IF(ABS(V).GE..05)GO TO 50
        IF(ABS(V).LT..005)GO TO 10
        IF(L.EQ.1)GO TO 10
```

```
      50    F=',F10.1'
            IF(L.EQ.2)F=',F7.2'
            RETURN
      60    F=',F10.3'
            IF(L.EQ.2)F=',F8.3'
            RETURN
      70    F=',F10.4'
            IF(L.EQ.2)F=',F8.4'
            RETURN
            END
```

PRINT3

```
      SUBROUTINE PRINT3(C,IT,JC,JR,KSTEP,NCOL,PTABLE,R,VAR)
C
      COMMON /DIM/ ICM1,ICOL,ICP1,ICP2,IRM1,IROW,IRX2
      COMMON /PRT/ CTR,DELTA,DENOM,HDG,HEAD,ICTR,IHDG,ISENS,PDELTA,PHDG
     . ,SENS,SNSCOL,SNSROW
C
      DIMENSION              C(10),                COL(10)
     .,FMT1(21),             FMT2(41),             FMT3(21)
     .,HEAD(5,80),           JC(10),               JR(10)
     .,R(10),                VAR(ICP2,IRX2)
C
      INTEGER                CENTER/'CENTER'/,     DELTA /'DELTA '/
     .,HDG   /'HDG   '/,     PHDG,                 SENS  /'SENS  '/
C
      DATA                   BLANK /'      '/,     F1N   /'F9.1  '/
     .,F1S   /'F9.5  '/,     F2    /'A6    '/,     F3    /',1X,  '/
     .,F4    /',2X,  '/,     F5    /',4X,  '/,     F6    /'''('' '/
     .,F7    /''')'' '/,     F8    /')     '/,     F9    /'(2X,  '/
     .,F10   /'A9    '/,     F11   /''' '' '/,     FMT1(1)/'(1XA2,'/
     .,FMT3(1)/'(1X, '/,     UNKNWN/'UNKNWN'/
C
      EQUIVALENCE            (BLANK,IBLANK)
C
C               SET UP FORMATS
C
      F1=F1N
      IF(ISENS.EQ.SENS.OR.ISENS.EQ.DELTA)F1=F1S
      DO 10 II=1,NCOL
      I=2*II
      FMT1(I)=F2
      FMT1(I+1)=F5
      FMT3(I)=F1
      FMT3(I+1)=F3
      I=4*II-3
      FMT2(I)=F4
      FMT2(I+1)=F6
      IF(BOOL(C(II)).EQ.BLANK)FMT2(I+1)=F11
      FMT2(I+2)=F2
      FMT2(I+3)=F7
      IF(BOOL(C(II)).EQ.BLANK)FMT2(I+3)=F11
   10 CONTINUE
      I=2*NCOL+1
      FMT1(I)=F8
      FMT3(I)=F8
      FMT2(I)=F9
      I=4*NCOL+1
      FMT2(I)=F8
      IF(PHDG.EQ.HDG.AND.KSTEP.EQ.1)PRINT 15
   15 FORMAT(/)
      IF(PHDG.NE.HDG)GO TO 30
      IF(KSTEP.GT.1)GO TO 40
C
C               PRINT HEADING
C
      NC=10*NCOL
      DO 26 J=1,IHDG
```

```
      K=1
      K1=1
      K2=79
      IF(ICTR.NE.CENTER)GO TO 26
      CALL POS(HEAD,J,K1,K2)
      IF(K1.NE.0)GO TO 20
      K1=1
      K2=1
   20 K=MAX0(1,(NC+K1-K2)/2)
   26 PRINT 28,(BLANK,I=1,K),(HEAD(J,I),I=K1,K2)
   28 FORMAT(120A1)
C
C                 PRINT COLUMN HEADINGS
C
   30 IF(KSTEP.GT.1)GO TO 40
      PRINT 15
      PRINT FMT1,IT,(C(I),I=1,NCOL)
      PRINT FMT2,(R(I),I=1,NCOL)
      PRINT 35
   35 FORMAT('     ')
C
C                 SORT DATA TO BE PRINTED
C
   40 DO 48 I=1,NCOL
      IF(BOOL(C(I)).EQ.BLANK)GO TO 42
      IC=JC(I)
      IR=JR(I)
      COL(I)=VAR(IC,IR)
      IF(ABS(COL(I)).GE..05)GO TO 48
      IF(ABS(COL(I)).GE..00005.AND.ISENS.EQ.SENS)GO TO 48
      IF(BOOL(COL(I)).EQ.IBLANK)GO TO 44
   42 COL(I)=BLANK
      GO TO 46
   44 COL(I)=UNKNWN
   46 FMT3(2*I)=F10
   48 CONTINUE
C
C                 PRINT DATA
C
      PRINT FMT3,(COL(I),I=1,NCOL)
      RETURN
      END
```

READ

```
      SUBROUTINE READ(COLUMN,WORD,IOP,ITYPE,PRINT,IWORD,STNMBR)
C
C               SUBROUTINE READ READS A CARD CONTAINING ONE OR MORE WORDS
C                (VARIABLES AND/OR NUMBERS) AND ASSOCIATED OPERATORS.
C
C               COLUMN IS AN ARRAY INTO WHICH THE 80 COLUMNS OF THE CARD ARE READ
C
C               WORD IS AN ARRAY CONTAINING THE VARIABLES AND/OR FLOATING
C                POINT NUMBERS.
C
C               IOP INDICATES THE OPERATOR (+,-,*,/) ASSOCIATED WITH
C                THE WORD.
C
C               ITYPE INDICATES WHETHER WORD IS A VARIABLE OR NUMBER.
C
C               IF PRINT=DATA, THE CARD WILL BE PRINTED OUT AS READ.
C
C               IWORD IS THE NUMBER OF WORDS (VARIABLES AND/OR NUMBERS)
C                ON THE CARD.
C
C               IF AN END-OF-FILE IS ENCOUNTERED, READ RETURNS TO
C                STATEMENT NUMBER 'STNMBR' IN THE CALLING PROGRAM.
C
C               EACH WORD MUST HAVE AN ASSOCIATED OPERATOR, EXCEPT THE
C                FIRST TWO WORDS.  THE FIRST WORD HAS NO OPERATOR.  IF NO
C                OPERATOR IS SPECIFIED FOR THE SECOND WORD, A '+' IS
C                ASSUMED.  IF THERE ARE TWO OR MORE WORDS, THE FIRST MUST
C                BE FOLLOWED BY AN '='   ANY CHARACTER OR NUMBER EXCEPT
C                '=' IS PERMITTED IN THE FIRST WORD, BUT THE FIRST WORD
C                MUST HAVE AT LEAST ONE NON-NUMERICAL CHARACTER.  ANY
C                CHARACTER OR NUMBER EXCEPT '=', '+', '-', '*', '/', ',',
C                OR '.' IS PERMITTED IN SUBSEQUENT WORDS.  ONLY ONE '='
C                IS PERMITTED PER CARD, AND TWO OPERATORS MAY NOT APPEAR
C                IN SUCCESSION.  BLANKS ARE IGNORED, AND IF ANY WORD
C                (EXCEPT A NUMBER) HAS MORE THAN 6 CHARACTERS, ONLY THE
C                FIRST 6 ARE CONSIDERED.
C
C
      DIMENSION              IOP(40),               ITYPE(40)
     .,J(40),                K(40),                 WORD(40)
C
      INTEGER                COLUMN(80),            COMMA /',      '/
     .,DATA  /'DATA  '/,     END   /'END   '/,      EQUAL /'=      '/
     .,IBLANK/'      '/,     MINUS /'-     '/,      NUMBER/'NUMBER'/
     .,PLUS  /'+     '/,     POINT /'.     '/,      PRINT
     .,SLASH /'/     '/,     STAR  /'*     '/,      TEMP(20,40)
     .,VARBLE/'VARBLE'/
C
C               INITIALIZE
C
 1    IEQUAL=0
      IPOINT=0
      IWORD=0
      DO 10 I=1,40
      IOP(I)=IBLANK
      ITYPE(I)=IBLANK
```

```
      J(I)=0
      WORD(I)=BOOL(IBLANK)
      K(I)=-1
      DO 10 L=1,20
      TEMP(L,I)=IBLANK
   10 CONTINUE
C
C               READ A DATA CARD & PRINT IT OUT IF PRINT=DATA
C
   20 READ(5,200,END=90)COLUMN
      IF(PRINT.NE.DATA)GO TO 22
      PRINT 210,COLUMN
   22 CONTINUE
C
C               GO THROUGH THE CARD COLUMN BY COLUMN
C
      DO 70 I=1,80
      IC=COLUMN(I)
      IF(IC.EQ.IBLANK)GO TO 70
      IF(IC.EQ.EQUAL)GO TO 40
      IF(IC.EQ.POINT)GO TO 50
      IF(IC.EQ.COMMA)GO TO 30
      IF(IC.EQ.PLUS.OR.IC.EQ.MINUS.OR.IC.EQ.STAR.OR.IC.EQ.SLASH)GO TO 30
      GO TO 60
C
C               THIS COLUMN IS AN OPERATOR (+,-,*,/) OR A COMMA
C
   30 IEQ=IEQUAL+1
      GO TO(65,34,100,32),IEQ
   32 IWORD=IWORD+1
      IPOINT=0
   34 IEQUAL=2
      IOP(IWORD+1)=IC
      GO TO 70
C
C               THIS COLUMN IS AN EQUAL SIGN
C
   40 IF(IEQUAL.GT.0)GO TO 110
      IF(J(1).EQ.0)GO TO 120
      IEQUAL=1
      IWORD=1
      GO TO 70
C
C               THIS COLUMN IS A DECIMAL POINT
C
   50 IF(IEQUAL.EQ.0)GO TO 65
      IF(IPOINT.EQ.1)GO TO 130
      IEQ=IEQUAL+1
      GO TO(65,52,54,54),IEQ
   52 IOP(IWORD+1)=PLUS
      IEQUAL=3
   54 IPOINT=1
      K(IWORD+1)=J(IWORD+1)
      GO TO 70
C
C               THIS COLUMN IS A NUMBER OR A CHARACTER OF A VARIABLE.
C               SQUEEZE OUT BLANKS, EQUAL SIGNS, DECIMAL POINTS,
```

```
C                 AND OPERATORS
C
 60     IEQ=IEQUAL+1
        GO TO (65,62,64,65),IEQ
 62     IOP(IWORD+1)=PLUS
 64     IEQUAL=3
C
C                 STORE THIS CHARACTER
C
 65     J(IWORD+1)=J(IWORD+1)+1
        JIWORD=J(IWORD+1)
        TEMP(JIWORD,IWORD+1)=IC
 70     CONTINUE
C
C                 ENTIRE CARD HAS BEEN READ - CHECK WHETHER CARD IS BLANK
C
        IF(J(1).EQ.0)GO TO 1
C
C                 CARD IS NOT BLANK - DETERMINE NUMBER OF WORDS
C
        DO 74 I=2,40
        IF(IOP(I).EQ.IBLANK)GO TO 76
 74     CONTINUE
 76     IWORD=I-1
        IF(IEQUAL.EQ.0)IWORD=1
        IF(IWORD.EQ.1.AND.IEQUAL.EQ.1)GO TO 77
        GO TO 78
 77     IWORD=2
        IOP(2)=PLUS
 78     CONTINUE
C
C                 PROCESS SAVED WORDS
C
        DO 80 I=1,IWORD
 80     CALL TYPE(WORD(I),ITYPE(I),J(I))
        IF(ITYPE(1).EQ.NUMBER) GO TO 140
        RETURN
C
C                 AN END-OF-FILE HAS BEEN DETECTED
C
 90     RETURN 7
C
C                 AN ERROR HAS BEEN DETECTED - PRINT ERROR MESSAGE
C
 100    PRINT 105
 105    FORMAT(' ABOVE LINE IGNORED (TWO OPERATORS IN SUCCESSION)')
        GO TO 1
 110    PRINT 115
 115    FORMAT(' ABOVE LINE IGNORED (TWO EQUAL SIGNS)')
        GO TO 1
 120    PRINT 125
 125    FORMAT(' ABOVE LINE IGNORED (FIRST CHARACTER IS AN EQUAL)')
        GO TO 1
 130    PRINT 135
 135    FORMAT(' ABOVE LINE IGNORED (TWO DECIMALS IN ONE NUMBER)')
        GO TO 1
 140    PRINT 145
```

```
  145   FORMAT(' ABOVE LINE IGNORED (FIRST WORD IS A NUMBER)')
        GO TO 1
C
C               FORMAT STATEMENTS
C
  200   FORMAT(80A1)
  210   FORMAT(' '80A1)
C
C
C               INTERNAL SUBROUTINE TYPE - IDENTIFIES WORD AS
C               A NUMBER OR VARIABLE AND RETURNS VALUE OR VARIABLE
C               IN WORD
C
        SUBROUTINE TYPE(WRD,ITYPE,J)
C
        WRD=0.
        ITYPE=IBLANK
        IF(K(I).LT.0)K(I)=J
        DO 10 L=1,J
        NMBR=FLD(0,6,TEMP(L,I))-48
        IF(NMBR.LT.0.OR.NMBR.GT.9.AND.TEMP(L,I).NE.POINT)GO TO 20
   10   WRD=WRD+NMBR*10.**(K(I)-L)
C
C
C               WORD IS A NUMBER
C
        ITYPE=NUMBER
        RETURN
C
C               WORD IS NOT A NUMBER
C
   20   JWRD=0
        IF(J.GT.6)J=6
        IF(J.EQ.0)J=1
        L=1
   30   IF(L.EQ.J)GO TO 40
        JWRD=JWRD+FLD(0,6,TEMP(L,I))*64**(6-L)
        L=L+1
        GO TO 30
   40   JWRD=JWRD+FLD(0,42-6*L,TEMP(L,I))
        ITYPE=VARBLE
        WRD=BOOL(JWRD)
C
        RETURN
C
        END
```

ROCKET

```
      SUBROUTINE ROCKET(DV,ISP,MI,MP,MC,MF,LINEAR,DONE,ITFLG,ERRTAG,$)
C
      REAL ISP,MI,MP,MC,MF
      LOGICAL LINEAR,DONE,LDV,LISP,LMI,LMP,LMC,LMF
      DATA BLANK/' '/,  G/32.174048/,  Z/.0001/
      INTEGER YES/'YES'/
C
      CALL SUM(DUMMY,$1050)
C
      LDV=.NOT.(BOOL(DV).EQ.BLANK)
      LISP=.NOT.(BOOL(ISP).EQ.BLANK)
      LMI=.NOT.(BOOL(MI).EQ.BLANK)
      LMP=.NOT.(BOOL(MP).EQ.BLANK)
      LMC=.NOT.(BOOL(MC).EQ.BLANK)
      LMF=.NOT.(BOOL(MF).EQ.BLANK)
C
      IF(LINEAR.AND.(ABS(MC).GT..01.OR.BOOL(MC).EQ.BLANK))GO TO 500
C
C             CONSUMABLES DROPPED DISCRETELY AFTER BURN
C
      IF(.NOT.LDV.AND.LISP.AND.LMI.AND.LMP)GO TO 100
      IF(LDV.AND..NOT.LISP.AND.LMI.AND.LMP)GO TO 150
      IF(.NOT.(LDV.AND.LISP))RETURN
      IF(.NOT.LMI.AND.LMP)GO TO 200
      IF(LMI.AND..NOT.LMP)GO TO 250
      IF(.NOT.LMI.AND..NOT.LMP.AND.LMC.AND.LMF)GO TO 300
      IF(LMI.AND.LMP.AND.LMC.AND.LMF)GO TO 1000
      RETURN
C
  100 IF(ABS(MI-MP).LT.Z)GO TO 1100
      IF(MI/(MI-MP).LT.Z)GO TO 1100
      DV=ISP*G*ALOG(MI/(MI-MP))
      IF(LMC)DONE=.TRUE.
      RETURN
C
  150 IF(ABS(MI-MP).LT.Z)GO TO 1100
      IF(MI/(MI-MP).LT.Z)GO TO 1100
      IF(ABS(G*ALOG(MI/(MI-MP))).LT.Z)GO TO 1100
      ISP=DV/(G*ALOG(MI/(MI-MP)))
      IF(LMC)DONE=.TRUE.
      RETURN
C
  200 IF(ISP.LT.Z)GO TO 1100
      IF(-DV/(ISP*G).GT.88.)GO TO 1100
      IF(ABS(1.-EXP(-DV/(ISP*G))).LT.Z)GO TO 1100
      MI=MP/(1.-EXP(-DV/(ISP*G)))
      IF(.NOT.(.NOT.LMC.AND..NOT.LMF))GO TO 900
      RETURN
C
  250 IF(ISP.LT.Z)GO TO 1100
      IF(-DV/(ISP*G).GT.88.)GO TO 1100
      MP=MI*(1.-EXP(-DV/(ISP*G)))
      IF(.NOT.(.NOT.LMC.AND..NOT.LMF))GO TO 900
      RETURN
C
  300 IF(ISP.LT.Z)GO TO 1100
```

```
      IF(DV/(ISP*G).GT.88.)GO TO 1100
      MI=(MF+MC)*EXP(DV/(ISP*G))
      GO TO 900
C
C                CONSUMABLES USED LINEARLY DURING BURN
C
  500 IF(LMI.AND.LMP.AND.LMF)GO TO 550
      IF(.NOT.(LDV.AND.LISP))RETURN
      IF(ISP.LT.Z)GO TO 1100
      C=DV/(ISP*G)
      IF(LMP.AND.LMC)GO TO 640
      IF(LMI.AND.LMF)GO TO 660
      IF(LMF.AND.LMC)GO TO 700
      IF(LMP.AND.LMF)GO TO 750
      IF(LMI.AND.LMP)GO TO 800
      IF(LMI.AND.LMC)GO TO 850
      RETURN
C
  550 IF(.NOT.LDV.AND.LISP)GO TO 600
      IF(LDV.AND..NOT.LISP)GO TO 620
      IF(LDV.AND.LISP)GO TO 1000
      RETURN
C
  600 IF(ABS(MF).LT.Z)GO TO 1100
      IF(MI/MF.LT.Z)GO TO 1100
      IF(ABS((MI-MF)*ALOG(MI/MF)).LT.Z)GO TO 1100
      DV=ISP*G*MP/(MI-MF)*ALOG(MI/MF)
      DONE=.TRUE.
      RETURN
C
  620 IF(ABS(MF).LT.Z)GO TO 1100
      IF(MI/MF.LT.Z)GO TO 1100
      IF(ABS(G*MP*ALOG(MI/MF)).LT.Z)GO TO 1100
      ISP=DV*(MI-MF)/(G*MP*ALOG(MI/MF))
      DONE=.TRUE.
      RETURN
C
  640 IF(ABS(MP).LT.Z)GO TO 1100
      IF(C*(MP+MC)/MP.GT.88.)GO TO 1100
      IF(ABS(EXP(C*(MP+MC)/MP)-1.).LT.Z)GO TO 1100
      MF=(MP+MC)/(EXP(C*(MP+MC)/MP)-1.)
      GO TO 900
C
  660 IF(ABS(MF).LT.Z)GO TO 1100
      IF(MI/MF.LT.Z)GO TO 1100
      IF(ABS(ALOG(MI/MF)).LT.Z)GO TO 1100
      MP=C*(MI-MF)/ALOG(MI/MF)
      GO TO 900
C
  700 MP=(MF+MC/2.)*(EXP(C)-1.)
      CALL INIT(MF,1.,$1100)
      DO 710 K=1,20
      IF(ABS(MP).LT.Z)GO TO 1100
      IF(C*(MP+MC)/MP.GT.88.)GO TO 1100
      X=EXP(C*(MP+MC)/MP)
      FMP=MF*X-MP-MC-MF
      DFMP=-1.-C*MF*MC*X/MP**2
```

```
      DLTMP=FMP/DFMP
      IF(ABS(DLTMP).LT..001)GO TO 900
  710 MP=MP-DLTMP
      GO TO 1150
C
  750 MI=MF+2.*(MP/C-MF)
      IF(MF*MI.LT.0.)MI=MP/(2.*C)
      DO 760 K=1,20
      IF(ABS(MP).LT.Z)GO TO 1100
      IF(C*(MI-MF)/MP.GT.88.)GO TO 1100
      X=EXP(C*(MI-MF)/MP)
      FMI=-MI+MF*X
      DFMI=-1.+C*MF*X/MP
      DLTMI=FMI/DFMI
      IF(ABS(DLTMI).LT..001)GO TO 900
  760 MI=MI-DLTMI
      GO TO 1150
C
  800 MF=MI+2.*(MP/C-MI)
      IF(MF*MI.LT.0.)MF=MP/(2.*C)
      DO 810 K=1,20
      IF(ABS(MP).LT.Z)GO TO 1100
      IF(-C*(MI-MF)/MP.GT.88.)GO TO 1100
      X=EXP(-C*(MI-MF)/MP)
      FMF=MI*X-MF
      DFMF=C*MI*X/MP-1.
      DLTMF=FMF/DFMF
      IF(ABS(DLTMF).LT..001)GO TO 900
  810 MF=MF-DLTMF
      GO TO 1150
C
  850 MP=MI*(1.-EXP(-C))
      CALL INIT(MI,-1.,$1100)
      DO 860 K=1,20
      IF(ABS(MP).LT.Z)GO TO 1100
      IF(-C*(MP+MC)/MP.GT.88.)GO TO 1100
      X=EXP(-C*(MP+MC)/MP)
      FMP=MI*(1.-X)-MC-MP
      DFMP=-1.-C*MI*MC*X/MP**2
      DLTMP=FMP/DFMP
      IF(ABS(DLTMP).LT..001)GO TO 900
  860 MP=MP-DLTMP
      GO TO 1150
C
  900 CALL SUM(DONE,$1050)
      RETURN
C
C               ALL VARIABLES HAVE VALUES--CHECK FOR CONSISTENCY
C
 1000 IF(LINEAR)GO TO 1010
      IF(ABS(MF+MC).LT.Z)GO TO 1100
      IF(MI/(MF+MC).LT.Z)GO TO 1100
      TST=ISP*G*ALOG(MI/(MF+MC))
      GO TO 1020
 1010 IF(ABS(MF).LT.Z)GO TO 1100
      IF(MI/MF.LT.Z)GO TO 1100
      IF(ABS((MI-MF)*ALOG(MI/MF)).LT.Z)GO TO 1100
```

```
      TST=ISP*G*MP/(MI-MF)*ALOG(MI/MF)
 1020 CONTINUE
      IF(ABS(DV-TST).GT..01)GO TO 1100
      DONE=.TRUE.
      RETURN
C
C               DISCREPANCY DISCOVERED--PRINT ERROR MESSAGE
C
 1050 IF(ITFLG.EQ.YES)RETURN 11
      PRINT 1080,DV,ISP,MI,MP,MC,MF
 1080 FORMAT(' DV=',F10.2,' ISP=',F10.2/
     . ' MI=',F10.2,' MP=',F10.2,' MC=',F10.2,' MF=',F10.2)
      RETURN 11
C
 1100 IF(ITFLG.EQ.YES)RETURN 11
      PRINT 1110,ERRTAG
 1110 FORMAT(' ***DISCREPANCY ENCOUNTERED IN ROCKET, EVENT ',A6)
      PRINT 1120,DV,TST,ISP,MI,MP,MC,MF
 1120 FORMAT(' DV=',F10.2,' DVTEST=',F10.2,' ISP=',F10.2/
     . ' MI=',F10.2,' MP=',F10.2,' MC=',F10.2,' MF=',F10.2)
      RETURN 11
C
C               NEWTON-RAPHSON ITERATION FAILED TO CONVERGE
C
 1150 IF(ITFLG.EQ.YES)RETURN 11
      PRINT 1160,ERRTAG
 1160 FORMAT(' ***NEWTON-RAPHSON FAILED, EVENT ',A6)
      GO TO 1050
C
C               INTERNAL SUBROUTINE
C
      SUBROUTINE SUM(DUN,$)
C
      DIMENSION A(3)
      LOGICAL DUN
C
      A(1)=MF
      A(2)=MP
      A(3)=MC
      CALL SUMS(MI,A,3,DUN,ITFLG,ERRTAG,$10)
      MF=A(1)
      MP=A(2)
      MC=A(3)
      RETURN
 10   RETURN 2
C
C               INTERNAL SUBROUTINE
C
      SUBROUTINE INIT(W,X,$)
C
      IF((X*W+MC)*SIGN(1.,X*W).LE.0.)MP=-MC
      IF(C*MC*W.GT.0.)RETURN
      S=-C*W/MC
      MP=-MC*(.5+X*C/4.)
      IF(S.LT.1..AND.ABS(MC).LT.ABS(C*MC/2.))MP=-C*MC
      DO 20 J=1,20
      STST=-EXP(X*C*(MP+MC)/MP)*C*W*MC/(MP*MP)
```

```
      IF(S.LT.1.)GO TO 10
      IF(STST.LT.1./S)RETURN
      MP=2.*MP
      GO TO 20
   10 IF(STST.GT.1./S)RETURN
      IF(J.EQ.20.AND.STST.GT.1.)RETURN
      IF(X*C.LT.0.)MP=.5*MP
      IF(X*C.GE.0.)MP=MP/2.-X*C*MC/4.
   20 CONTINUE
      RETURN 3
C
      END
```

SUMS

```
      SUBROUTINE SUMS(SUM,A,N,DONE,ITFLG,ERRTAG,$)
C
C             SUBROUTINE SUMS CHECKS SUM AND THE ARRAY A, DIMENSIONED N.
C              IF SUM AND ALL THE A'S HAVE VALUES, SUMS CHECKS THE VALIDITY
C              OF THE SUM. IF A DISCREPANCY IS ENCOUNTERED, AN ERROR MESSAGE
C              IS PRINTED AND SUMS RETURNS TO $.  IF SUM AND THE A'S ARE
C              CONSISTENT, SUMS RETURNS WITH 'DONE'=TRUE.
C
C              IF EITHER SUM OR ONLY ONE OF THE A'S IS UNKNOWN, SUMS
C              CALCULATES THE UNKNOWN VARIABLE AND RETURNS TO THE
C              CALLING PROGRAM WITH 'DONE'=TRUE.
C
C              IF THERE ARE > 1 UNKNOWN, SUMS RETURNS TO THE CALLING PROGRAM.
C
C              THE LOGICAL VARIABLE 'DONE' IS SET TO TRUE IF A VALID
C              SOLUTION IS FOUND.
C
      LOGICAL DONE
      INTEGER ERRTAG, YES/'YES'/
      DATA IBLANK/' '/
      DIMENSION A(N)
C
C             COUNT THE NUMBER OF UNKNOWNS
C              (NONE IF M=1, ONE IF M=0, >1 IF M<0)
C
      L=0
      M=1
      DO 10 J=1,N
      IF(BOOL(A(J)).NE.IBLANK)GO TO 10
      L=J
      M=M-1
   10 CONTINUE
      IF(BOOL(SUM).EQ.IBLANK)M=M-1
      IF(M)90,,50
C
C             ONLY ONE BLANK---FIND SINGLE UNKNWN
C
      IF(L.NE.0)GO TO 30
C
C             SUM IS THE ONLY UNKNOWN
C
      SUM=0.
      DO 20 J=1,N
   20 SUM=SUM+A(J)
      GO TO 80
C
C             A(L) IS THE ONLY UNKNOWN
C
   30 A(L)=0.
      TEMP=SUM
      DO 40 J=1,N
   40 TEMP=TEMP-A(J)
      A(L)=TEMP
      GO TO 80
C
C             NO BLANKS---CHECK VALIDITY OF SUM
```

```
C
 50     TST=0.
        DO 60 J=1,N
 60     TST=TST+A(J)
        IF(ABS(SUM-TST).LT..05)GO TO 80
C
C               DISCREPANCY ENCOUNTERED
C
        IF(ITFLG.EQ.YES)RETURN 7
        PRINT 70,ERRTAG,SUM,TST,(A(J),J=1,N)
 70     FORMAT(' ***DISCREPANCY ENCOUNTERED IN SUMS, EVENT ',A6/
     . ' SUM='F10.2,' SUMTST='F10.2,' A='4F10.2/6X,6F10.2)
        RETURN 7
C
C               A VALID SOLUTION HAS BEEN FOUND
C
 80     DONE=.TRUE.
 90     RETURN
        END
```

TEST

```
      SUBROUTINE TEST(IBURN,IRKC,IROCK,JCOL,LINFLG,NROW,VAR,$)
C
      COMMON/DIM/ ICM1,ICOL,ICP1,ICP2,IRM1,IROW,IRX2
C
      DIMENSION                IBURN(IROW),          IRKC(IROW)
     .,IROCK(IROW),            JCOL(IROW,ICOL),      NROW(IRX2)
     .,VAR(ICP2,IRX2)
C
      DATA BLANK/'
C
      LOGICAL LINFLG(IROW)
C
C                INITIALIZE
C
      MF=ICP2
      MT=ICOL
      KF=IROW-1
      NF=IRX2
      IFLAG=0
      K=1
      M=1
      N=1
      MATRIX=0
      DO 1 I=1,NF
    1 NROW(I)=0
      DO 2 I=1,KF
      IROCK(I)=0
      DO 3 J=1,MT
    3 JCOL (I,J)=0
    2 CONTINUE
C
C                SET LIMITS FOR EQUATIONS
C
      IRC=ICOL-1
      ICC=2
      DO 4 K=1,KF
      IF (IBURN(K) .EQ. 0) GO TO 4
      IF (LINFLG(K)) IRKC(K)=ICOL+1
      IF (.NOT. LINFLG(K)) IRKC(K)=3
    4 CONTINUE
      NRKT=0
      DO 6 K=1,KF
    6 IF (IBURN(K) .NE. 0) NRKT=NRKT+1
      MC=(IRX2*ICP2-2*(IRX2-NRKT))-(NRKT+IRX2+ICOL*KF-KF)
C
C                TEST EACH KNOWN VARIABLE FOR INDEPENDENCE
C
   10 IF (M .GT. MT .AND. (N .NE. 2*K-1 .OR. IBURN(K) .EQ. 0)) GO TO 40
      IF (BOOL(VAR(M,N)) .NE. BLANK) GO TO 20
   40 IF (M .EQ. MF) GO TO 30
      M=M+1
      GO TO 10
   30 IF (N .EQ. NF) GO TO 120
      N=N+1
      M=1
      IF (N .GT. (2*K) .AND. N .LT. NF) K=K+1
```

```
      GO TO 10
C
C                TEST SUM ACROSS ROW
C
   20 IF (M .GT. MT) GO TO 45
      IF (NROW(N) .EQ. IRC) GO TO 42
      NROW(N)=NROW(N)+1
      GO TO 45
   42 IFLAG=1
C
C                TEST ROCKET EQUATION
C
   45 IF (IBURN(K) .EQ. 0) GO TO 50
      IF ((M .GE. MT .AND. N .NE. (2*K-1)) .OR. (M .LT. MT .AND. N .NE.
     1 (2*K))) GO TO 50
      IF (.NOT. LINFLG(K) .AND. M .LT. MT .AND. M .NE. IBURN(K)) GO TO
     1 50
      IF (IROCK(K) .EQ. IRKC(K)) GO TO 47
      IROCK(K)=IROCK(K)+1
      GO TO 50
   47 IFLAG=1
C
C                TEST SUM DOWN COLUMN
C
   50 IF (M .GT. MT) GO TO 80
      IF (JCOL(K,M) .EQ. ICC) GO TO 60
      JCOL(K,M)=JCOL(K,M)+1
      GO TO 70
   60 IFLAG=1
   70 IF (K.EQ. 1 .OR. N .NE. (2*K-1)) GO TO 80
      IF (JCOL(K-1,M) .EQ. ICC) GO TO 90
      JCOL(K-1,M)=JCOL(K-1,M)+1
      L=K-1
      GO TO 80
   90 IFLAG=1
C
C                TOTAL INDEPENDENT VARIABLES KNOWN
C
   80 IF (IFLAG .NE. 1 ) MATRIX=MATRIX+1
  110 IFLAG=0
      GO TO 40
  120 IF (MATRIX .LT. MC) RETURN 8
      RETURN
      END
```

APPENDIX D

DESCRIPTION OF COMMANDS USED DURING EXECUTION OF MAIN PROGRAM

DELTA

See SENS commands.

DISCRE(TE) †

See LINEAR command.

END

The END (or STØP) command terminates execution of the program, after printing the message "STØP or END COMMAND--RUN TERMINATED." If a STØP or END command is not used, the next system control card (beginning with "@") will stop execution of the program.

HDG

The HDG commands are used to provide information about the heading to be printed. The HDG commands do not affect the column headings.

HDG=CENTER or
HDG=NØCENT(ER)

Determines whether the heading will be centered. For HDG=CENTER, the heading will be centered. For HDG=NØCENT, the heading will begin in the second column. If not specified, CENTER is assumed.

HDG=NUMBER

Specifies the number of lines in the heading. NUMBER may be 1, 2, 3, 4, or 5. If not specified, NUMBER is assumed to be 1.

---

† Throughout this memorandum, those characters enclosed in parentheses in a command may be omitted. Thus, the command DISCRE(TE) may be used as either DISCRETE or DISCRE.

```
HDG
LINE 1
LINE 2
  .
  .
  .
LINE N
```

Used to input the desired heading. The next N lines (N=1, 2, 3, 4, or 5 as described under HDG=N) after the HDG card will be read in as the N lines of the heading. This heading will be saved until a new heading is provided. If N is subsequently changed, the heading data already saved is not deleted, but the number of lines printed will be determined by the new N.

ITERAT

The ITERAT command is used to inhibit the use of subroutine ITERAT. This command allows the user to preclude the possibility of an unusually long run time which might result from some cases that require attempts to iterate with many different combinations of known and unknown elements of the matrix.

ITERAT=YES or
ITERAT=NØ

For ITERAT=YES iteration is permitted. For ITERAT=NØ iteration is not permitted. If not specified, YES is assumed.

LAST

The LAST command tells the program to resume computations to evaluate the unknowns in the matrix, in accordance with previously provided data and/or commands. After the results are printed, the initial conditions just prior to receipt of the LAST command are restored and the program is ready to accept new data and/or commands.

LINEAR

The LINEAR command, when used after an event is specified, causes the calculations to be based on the assumption that any weight change prior to the next event (excluding the propellant required) occurs linearly during the burn. The DISCRE(TE) command

causes the calculations to be based on the assumption that this weight change occurs discretely after the burn. These designations remain in effect until changed by a new LINEAR or DISCRE command. When results are printed, the word LINEAR or the letter L is used to indicate that the weight change for this burn was treated linearly. No such word is printed if the weight change was treated discretely. If neither of these commands is used during execution of the program, the type of burn specified when the program was precompiled is assumed.

PRINT

The PRINT commands are used to control the data that are printed and the format in which the data are printed.

PRINT=CØLUMN(S)±ALL±C1±C2...or
PRINT=EVENTS±ALL±E1±E2

Specifies the columns (or events) to be used or omitted when the output data are printed. A "+" means the column (or event) is to be used, and a "-" means the column (or event) is to be omitted. "±ALL" means all of the columns (or events) are to be used or omitted. If one or more columns (or events) are omitted, the remaining columns (or events) are "squeezed" in to fill up the vacant spaces.

PRINT=DATA or
PRINT=NØDATA

Determines whether the input data (card images or lines typed in at a terminal) will be printed. For PRINT=DATA, all input data will be printed. For PRINT=NØDATA, the input data will not be printed. The latter option allows one to avoid waiting for a lengthy stream of data to be printed when using the ADD statement from a terminal. If not specified, DATA is assumed.

PRINT=DELTA or
PRINT=NØDELT(A)

Determines whether the rows of the output matrix representing the weight changes between two events will be printed. For PRINT=DELTA, these rows will be printed. For PRINT=NØDELT, they will not be printed. If not specified, DELTA is assumed.

PRINT=GØØF or
PRINT=NØGØØF

If the given data are inconsistent, an error message with a summary of pertinent data is printed. However, this summary is sometimes not sufficient to determine the changes necessary. For PRINT=GØØF, the matrix, as calculated up to the point of the inconsistency, will be printed. For PRINT=NØGØØF, the matrix will not be printed. If not specified, NØGØØF is assumed.

PRINT=HDG or
PRINT=NØHDG

Determines whether the heading (which must be specified by the user - see HDG command) will be printed. For PRINT=HDG, the heading will be printed. For PRINT=NØHDG, the heading will not be printed. Heading data previously specified by HDG commands will not be destroyed by a PRINT=NØHDG command. If not specified, NØHDG is assumed.

PRINT=ITERAT(E) or
PRINT=NOITER(ATE)

Used as a debugging tool. For PRINT=ITERAT, the symbol "*" will be printed at the beginning of the column headings line if subroutine ITERAT was called during the calculation. For PRINT=NØITER, the "*" will not be printed. This command is not effective when the tabular format is used. If not specified, NØITER is is assumed.

PRINT=LABELS or
PRINT=NØLABE(LS)

Determines whether complete or abbreviated column and event labels will be printed. For PRINT=LABELS, the complete labels will be printed. For PRINT=NØLABE, the abbreviated labels will be printed. Two of the advantages of using PRINT=NØLABE are that less time is taken to print the results, and the abbreviated labels are identical to the column and event names used to input new data. When using a remote terminal, the number of columns to be printed should be limited to five if PRINT=LABELS or seven if PRINT=NØLABE. If not specified, NØLABE is assumed.

PRINT=STATUS,ALL

Causes the current status of all elements in the matrix to be printed.

PRINT=STATUS,PART

Causes the current status of all elements in an abbreviated form of the matrix to be printed. The columns and events included in this abbreviated matrix are determined by PRINT=EVENTS and PRINT=CØLUMN commands described elsewhere.

PRINT=STATUS,C1/E1,C2/E2....

Causes the current status of the specified elements (column C1 at event E1, column C2 at event E2, etc.) to be printed. If the name of the specified column is preceded by "D" (for delta), the element representing the change between the specified event and the succeeding event will be printed.

PRINT=TABLE or
PRINT=NØTABL(E)

Determines whether data will be printed in the normal format or in a tabular form. The elements included in the table are determined by another form of the PRINT=TABLE command described below. For PRINT=TABLE, the tabular form will be printed. For PRINT=NØTABL, the normal format will be used. The former option is convenient when using the STEP command. If not specified, NØTABL is assumed.

PRINT=TABLE, N1/C1/E1,N2/C2/E2....

Specifies the elements that comprise the columns to be printed when PRINT=TABLE. Column C1 at event E1 will be printed in column N1, column C2 at event E2 will be printed in column N2, etc. Up to 10 columns may be specified, in any order. (Only 7 columns should be used when operating from a remote terminal.) If NX/CX/EX is replaced by NX/DELETE, column NX will not be printed. The element to be used for a given column of the table may be changed with a new NX/CX/DX entry. (It is not necessary to delete the column before respecifying it.) If the name of a specified column is preceded by "D" (for delta), the element representing the change between the specified event and the succeeding event will be printed.

RESTAR(T)

The RESTAR(T) command resets all elements of the matrix to the values initially built into the program during precompilation. Only the elements of the matrix and the types of burns (linear or discrete) are affected; other data and/or commands are not changed (i.e., data provided by HDG, STEP, PRINT, etc. are unaffected).

SENS

The SENS commands are used to calculate the sensitivity of each element in the matrix to variations of a single specified element, referred to as the perturbing element or variable. When the SENS mode is used, the program first calculates the unknown elements of the matrix in the normal manner. The element that was designated as the perturbing element is then changed by the amount specified in a SENS command, and the matrix is re-evaluated. The difference between these two matrices is then calculated, and the resulting matrix is normalized by dividing all elements by the change to the perturbing element. The result is a set of sensitivities, or partial derivatives, of each element with respect to the perturbing variable.

The DELTA commands are used exactly like the SENS commands. The only difference is that the results are not normalized. To use them simply replace SENS with DELTA in the statements described below.

The element designated as the perturbing variable must not be an unknown. It is not advisable to run the program (i.e., to use a LAST card) with SENS=YES until after a perturbing element has been designated. When the results are printed in the normal format, an appropriate heading is automatically provided, indicating the perturbing variable and the amount by which it was perturbed.

The same storage locations are used for identifying the perturbing variable for both the SENS and the DELTA commands. Therefore, once a perturbing variable and the amount of the perturbation are identified for one of these commands, it will be maintained (until changed) for use by either of the two commands.

SENS=YES or
SENS=NØ

Determines whether the program will be run in the SENS mode. SENS=YES places the program in the SENS mode, and SENS=NØ removes it from the SENS mode.

The sensitivity matrix is not calculated until a LAST command is received. A SENS=NØ command does not destroy a previously designated perturbing variable. If not specified, NØ is assumed.

SENS=C/E±Δ*PERCEN

Specifies the perturbing variable and the amount of the perturbation. The perturbing variable is the element defined by column C, event E. A column name preceded by "D" (for delta) specifies the change between the specified event and the succeeding event. Δ specifies the magnitude of the perturbation. If "*PERCEN" is present, the perturbation will be ±Δ% of the perturbing variable. If "*PERCEN" is not present, the perturbation will be ±Δ. If "±Δ*PERCEN" is not present, a perturbation of +1% is assumed. The data contained in this command will remain in effect until changed by another SENS command.

STEP

The STEP commands are used to solve the matrix a number of times (N) in succession, with a designated variable being incremented by a specified constant each time. The sequence is initiated by a single LAST command. If the normal printing format is used, the entire (or abbreviated) matrix will be printed N times. If the tabular format is used, the heading and column headings will be printed only once, and N lines of data will be printed. The element designated as the perturbing variable must not be an unknown. It is not advisable to run the program (i.e., to use a LAST card) with STEP=YES until after a perturbing element has been designated.

.STEP=YES or
STEP=NØ

Determines whether the program will be run in the STEP mode. STEP=YES places the program in the STEP mode, and STEP=NØ removes it from the STEP mode. A STEP=NØ command does not destroy a previous designation of a stepping variable. If not specified, NØ is assumed.

STEP=C/E/N±Δ*PERCEN

Specifies the variable to be incremented, the number of increments, and the size of the increment. The variable to be incremented is defined by column C, event E. A column name preceded by "D" (for delta) specifies the change between the specified event and the succeeding event. N specifies the number of increments (including the first calculation, with an

increment of zero). Δ specifies the magnitude of the increment. If "*PERCEN" is present, the increment will be ±Δ% of the variable to be stepped. If "*PERCEN" is not present, the increment will be ±Δ. If "±Δ*PERCEN" is not present, an increment of +1% is assumed. If "/N±Δ*PERCEN" is not present, 5 increments of 1% are assumed. The data contained in this command will remain in effect until changed by another STEP command.

STØP

The STØP (or END) command terminates execution of the program, after printing the message "STØP or END COMMAND---RUN TERMINATED." If a STØP or END command is not used, the next system control card (beginning with "@") will stop execution of the program.

APPENDIX E

SUMMARY OF COMMANDS AND DATA CARDS

I. PRECOMPILATION

A. Column Specification Cards

(The forms shown in each group are equivalent)

```
CNAME*ONE/TWO/THREE/*
CNAME*ONE/TWO/THREE*

CNAME*ONE/TWO//*
CNAME*ONE/TWO/*
CNAME*ONE/TWO*

CNAME*ONE//THREE/*
CNAME*ONE//THREE*

CNAME*ONE///*
CNAME*ONE//*
CNAME*ONE/*
CNAME*ONE*

CNAME*/TWO//*
CNAME*/TWO/*
CNAME*/TWO*

CNAME*//THREE/*
CNAME*//THREE*

CNAME*///*
CNAME**
CNAME
```

B. Event Specification Cards

```
EVENT ENAME*HEADING*CNAME

EVENT ENAME*HEADING*

EVENT ENAME**
EVENT ENAME
```

C. Input Data Cards

```
NAME=NUMBER

NAME
```

D. Other Cards

DELETE

DISCRE(TE)

END

LAST

LINEAR

STØP

SUBHEADING

II. EXECUTION OF MAIN PROGRAM

A. Event Specification Cards

ENAME

B. Input Data Cards

NAME=NUMBER

NAME=ØLD

NAME=NAME op NUMBER op NUMBER ....

NAME

C. Commands

General Form	Abbreviated Form
DELTA=YES	D=Y
DELTA=NØ	D=N
DELTA=CØLUMN/EVENT ± Δ*PERCEN	D=CØLUMN/EVENT ± Δ*%
DISCRE(TE)	---
END	E
HDG=CENTER	H=C
HDG=NØCENT(ER)	H=NC
HDG=NUMBER	H=NUMBER
HDG LINE 1 LINE 2 . . . LINE N	H LINE 1 LINE 2 . . . LINE N

General Form	Abbreviated Form
ITERAT=YES	I=Y
ITERAT=NØ	I=N
LAST	L
LINEAR	---
PRINT=CØLUMN(S)±ALL±C1±C2...	P=C±A±C1±C2...
PRINT=EVENTS±ALL±E1±E2...	P=E±A±E1±E2...
PRINT=DATA	P=D
PRINT=NØDATA	P=ND
PRINT=DELTA	P=DL
PRINT=NØDELT(A)	P=NDL
PRINT=GØØF	P=G
PRINT=NØGØØF	P=NG
PRINT=HDG	P=H
PRINT=NØHDG	P=NH
PRINT=ITERAT(E)	P=I
PRINT=NØITER(ATE)	P=NI
PRINT=LABELS	P=L
PRINT=NØLABE(LS)	P=NL
PRINT=STATUS,ALL	P=S,A
PRINT=STATUS,PART	P=S,P
PRINT=STATUS,C1/E1,C2/E2...	P=S,C1/E1,C2/E2...
PRINT=TABLE	P=T
PRINT= NØTABL(E)	P=NT
PRINT=TABLE,N1/C1/E1,N2/C2/E2...	P=T,N1/C1/E1,N2/C2/E2..
PRINT=TABLE,N1/DELETE...	P=T,N1/D...
RESTAR(T)	R
SENS=YES	S=Y
SENS=NØ	S=N
SENS=CØLUMN/EVENT±Δ*PERCEN	S=CØLUMN/EVENT±Δ*%
STEP=YES	ST=Y
STEP=NØ	ST=N
STEP=CØLUMN/EVENT/N±Δ*PERCEN	ST=CØLUMN/EVENT/N±Δ*%
STØP	---